Decrypting:

Chiave (privata):

p(x) = sinx^2-tanx/3

A: // Utilizziamo la tabella ASCII per ottenere i rispettivi valori per ogni simbolo della stringa chiave.
// equivalente a: "4/4 AEOGC IMFNHQBDPL" // la stringa comincia sempre con il numero di righe/colonne, seguito da uno spazio.

{65,69,79,71},
{67,32,73,77},
{70,78,72,81},
{66,68,80,76}

Procedimento:

AAt:

{20268,17797,21371,20698},
{17797,16771,18679,18290},
{21371,18679,22729,21840},
{20698,18290,21840,21156}

/* Eliminiamo gli eventuali duplicati nella matrice, incrementando, di una unità, i numeri coinvolti tante volte quante necessario (ovvero fin quando non otteniamo un numero non presente nella matrice) */

AAt (senza duplicati):

{20268,17798,21372,20699},
{17797,16771,18680,18291},
{21371,18679,22729,21841},
{20698,18290,21840,21156}

det(AAt) = 133614
norma infinito di A = 84620

Inv(AAt): // i numeri ottenuti andranno a ridursi a, minimo, tre cifre decimali (si smette di inserire le cifre decimali solo quando queste sono ALMENO 3, di cui ALMENO una significativa)

{0.199,0.025,0.024,-0.241},
{0.026,0.005,0.005,-0.034},
{0.026,0.005,0.01,-0.04},
{-0.243,-0.034,-0.038,0.306}

```
    Inv(AAt) (senza duplicati):

    {0.199,0.025,0.024,-0.241},
    {1.026,1.005,2.005,0.966},
    {0.026,0.005,0.01,-0.04},
    {-0.243,-0.034,-0.038,0.306}

Riceviamo il vettore:

[601496.0, 678970.343, -3368.64, 555642.373, 645469.464, 455219.799, -232347.797, -167980.952, 12570.797, 162153.469,
619691.55, 500089.875, -349641.234, 520705.019, 581677.457, 487027.382, -48096.711, -243615.481, 90258.14, -565.335, -
8983.695, 156634.846, -73487.498, 559146.204, 88180.874, -268880.674, -22948.585, -180940.468, 476475.303, -273798.12,
465448.591, -278242.994, -74531.63, 217394.18, 289548.244, 386855.376, 50530.518, 168889.214, 78812.839, 10378.96, -
287011.879, -129593.876, -215135.339, -219648.068, 605839.897, -229244.612, 237581.092, 100501.395, -181748.011, 69.
159, 974.045, -201710.717, -75216.326, 603502.959, -10337.928]

Ricaviamo il numero di punti np = 55.
Sappiamo che inizialmente si aveva:

    idet = det(AAt) in 55 = 133614 in 55 = 19
    istart = norm(AAt) in 55 = 84620 in 55 = 30 // Nel caso sfortunato in cui idet e istart risultassero uguali (i),
    manterremmo nella posizione trovata 'i' idet e modificheremmo istart in 'i+1'
    ijump = norm(AAt)*det(AAt) in 55 = 11306416680 in 55 = 20 // La stessa identica cosa vale per ijump, il quale andrà
    però a confrontarsi sia con idet che con istart, incrementandosi ogni volta di uno, se necessario.

Adesso eseguiamo gli stessi calcoli del mittente per ottenere, come prima cosa, i 3 parametri di settaggio, cioè idet,
istart e ijump.

    yi --> (i*tval(i mod k)) mod np

    y(idet)   --> idet*tval(idet mod k) mod np = 19*tval(3) mod 55      = 19*21372 mod 55 = 3
    y(istart) --> istart*tval(istart mod k) mod np = 30*tval(14) mod 55 = 30*18290 mod 55 = 20
    y(ijump)  --> ijump*tval(ijump mod k) mod np   = 20*tval(4) mod 55  = 20*20699 mod 55 = 50

    Adesso sappiamo che dalla posizione 3,20,50 possiamo ottenere, rispettivamente, la lunghezza del messaggio reale,
    il suo indice di partenza e il parametro jump.

Innanzitutto dobbiamo rimuovere la modifica indotta dalla divisione con il polinomio, e per farlo è sufficiente
calcolare:

    oldy(i) = y(i)*p(i) // dove p(i) indica il risultato del polinomio in funzione di i, ridotto al minimo numero di
    cifre decimali (ovvero con la stessa logica di prima: minimo tre cifre decimali, di cui almeno una significativa)

    oldy(3)  = -3368.64*(sin(3^2)-tan(3)/3)  =  -3368.64*0.139 = -468.2409 -> -468.241
    oldy(20) = -565.336*0.521  = -294.540056 -> -294.54
    oldy(50) = 69.159*(-0.739) = -51.108501  -> -51.109
```

Così facendo, abbiamo ritrovato il valore antecedente la divisione con il polinomio. Ora, tentiamo di ottenere i valori originali, così facendo:

Il mittente ha eseguito (E' IMPORTANTE FARE ATTENZIONE AGLI INDICI IN QUANTO QUESTA OPERAZIONE E' STATA ESEGUITA PRIMA DEL RIORDINAMENTO, dunque, ad esempio, la 'i' da considerare per il ritrovamento della lunghezza del messaggio è l'originario 19, come 20 è quella per ritrovare l'indice di partenza del messaggio):

oldy(i) = vi*ival((i*tval(i mod k)) mod k)*tval(i mod k)/aval((aval(i mod j)*i) mod j) // // dove k è uguale al numero di elementi della matrice AAt e j il numero di elementi della matrice A (in questo specifico caso k=j=16)

Ne consegue che noi, per ottenere vi, eseguiremo: // arrotondando alle prime 3 cifre decimali

vi = oldy(i)/(ival((iorig*tval(iorig mod k)) mod k)*tval(iorig mod k)/aval((aval(iorig mod j)*iorig) mod j))

Applicandolo a i=3,i=20,i=50 otteniamo:

```
v3  = -468.241/(ival(19*tval(19 mod 16)) mod 16)*tval(19 mod 16)/aval((aval(19 mod 16)*19) mod 16)
    = -468.241/(-0.241*21372/66) = 5.99999883509... -> 6  // il messaggio contiene 6 caratteri
v20 = -294.54/(-0.04*18290/77)   = 30.9999726626... -> 31 // l'indice di partenza è x=31
v50 = -51.109/(-0.04*20699/81)   = 5.00003502584... -> 5  // jump = 5
```

Adesso che abbiamo queste tre informazioni, deduciamo che il messaggio, prima del riordinamento, partiva da 35 (visto che 31 non è multiplo di jump=5 e la prima posizione libera x che rispettava tale vincolo era 35), ed è lungo 6 caratteri.
Quindi le x significative, sempre prima del riordinamento, erano 35 - 40 - 45 - 50 - 55 - 5.
Ora che abbiamo questa informazione possiamo comprendere come sono state riordinate le postazioni.
Seguendo l'ordine di cui sopra, svolgiamo:

- Posizioni riordinate occupate a nostra conoscenza: 3,20,50.
y35 -> (35*tval(35 in 16)) in 55 = 35*21372 in 55 = 20, il quale è già occupato, dunque avanzo di una postazione fin quando non ne trovo una libera. Trovo 21.
- Posizioni riordinate occupate a nostra conoscenza: 3,20,50,21.
y40 -> (40*tval(40 in 16)) in 55 = 40*18291 in 55 = 30
- Posizioni riordinate occupate a nostra conoscenza: 3,20,50,21,30.
y45 -> (45*tval(45 in 16)) in 55 = 45*20698 in 55 = 40
- Posizioni riordinate occupate a nostra conoscenza: 3,20,50,21,30,40.
y50 -> (50*tval(50 in 16)) in 55 = 50*17798 in 55 = 55
- Posizioni riordinate occupate a nostra conoscenza: 3,20,50,21,30,40,55.
y55 -> (55*tval(55 in 16)) in 55 = 55*18680 in 55 = 55, il quale è già occupato, dunque avanzo di una posizione fin quando non ne trovo una libera. Trovo 1.
- Posizioni riordinate occupate a nostra conoscenza: 3,20,50,21,30,40,55,1.
y5 -> (5*tval(5 in 16)) in 55   = 5*17797 in 55 = 50, il quale è già occupato, dunque avanzo di una posizione fin quando non ne trovo una libera. Trovo 51.

Adesso conosciamo esattamente la posizione assunta dai valori significativi. Ci comportiamo come prima, con i 3 valori di settaggio, ed estraiamo le informazioni contenute nei valori rimasti:

Innanzitutto eliminiamo la modifica indotta dal polinomio, eseguendo ancora una volta oldy(i) = y(i)*p(i):

```
oldy(21) = -8983.695*0.860 = -7725.9777 -> -7725.978
oldy(30) = -273798.125*(-0.088) = 24094.235
oldy(40) = 10378.96*0.062 = 643.49552 -> 643.496
oldy(55) = -10337.928*0.098 = -1013.116944 -> -1013.117
oldy(1)  = 601496*0.012  = 7217.952
oldy(51) = 974.046*0.576 = 561.050496 -> 561.050
```

Ora, ricordandoci i seguenti legami tra posizioni originali e posizioni ridisposte, rispettivamente:

```
-------------
iorig <-> i
-------------
35 <-> 21
40 <-> 30
45 <-> 40
50 <-> 55
55 <-> 1
5  <-> 51
-------------
```

eseguiamo:

```
vi = oldy(i)/(ival((iorig*tval(iorig mod k)) mod k)*tval(iorig mod k)/aval((aval(iorig mod j)*iorig) mod j))

v21 = -7725.978/(ival(35*tval(35 mod 16)) mod 16)*tval(35 mod 16)/aval((aval(35 mod 16)*35) mod 16)
    = -7725.978/(ival((35*tval(3)) mod 16)*tval(3)/aval((aval(3)*35) mod 16))
    = -7725.978/(ival((35*21372) mod 16)*21372/aval((79*35) mod 16))
    = -7725.978/((-0.241)*21372/66 = 99 -> ASCII_decode(99) = 'c'
v30 = 24094.235/() = 105 -> ASCII_decode(105) = 'i'
v40 =              = 97  -> ASCII_decode(97)  = 'a'
v55 =              = 111 -> ASCII_decode(111) = 'o'
v1  =              = 31  -> ASCII_decode(32)  = ' '
v51 =              = 97  -> ASCII_decode(97)  = 'a'
```

Abbiamo dunque decifrato il messaggio, il quale è: "ciao a"