Reti di Feistel

Come mostrato nell'Esempio sulla crittografia con SPN, l'uso dell'operatore binario XOR (⊕) è molto utilizzato:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

In realtà gli schemi SPN utilizzati nella cifratura simmetrica moderna non usano solo lo XOR come funzione per «mescolare» la chiave a ogni round, ma funzioni più complesse, la cui utilizzazione fu proposta dallo studioso *Horst Feistel* (1915-1990) (Reti di Feistel).

Gli standard di cifratura simmetrica più utilizzati in ambito informatico sono il DES (*Data Encryption Standard*, con chiave a 56 bit), RC2 e RC4 (*Rivest cipher 2 e 4*, con chiave a 128 bit), l'IDEA (*International Data Encryption Algorithm*, con chiave a 128 bit), il 3DES (*Triple DES*, con chiave a 168 bit), Camellia (con chiave a 256 bit), l'AES (*Advanced Encryption Standard*, con chiave a 256 bit).

Tra questi solo IDEA, Camellia e AES sono tuttora (2013) inviolati.

La crittografia a chiave simmetrica ha due caratteristiche che vanno ricordate:

a) **è veloce**: crittografare e decriptare sono operazioni che non necessitano di onere computazionale. Le SP-box sono vettori indicizzati e lo XOR è un'operazione molto semplice;
b) **la chiave segreta deve essere condivisa**: questo è un problema. Lo scambio delle chiavi spesso deve avvenire proprio tramite la stessa rete dalla quale ci si vuole proteggere con la crittografia simmetrica: se la violazione avviene sullo scambio delle chiavi e venisse intercettata (per esempio, con un *keylogger*), anche la crittografia simmetrica più efficiente viene compromessa.

2 Crittografia a chiave asimmetrica

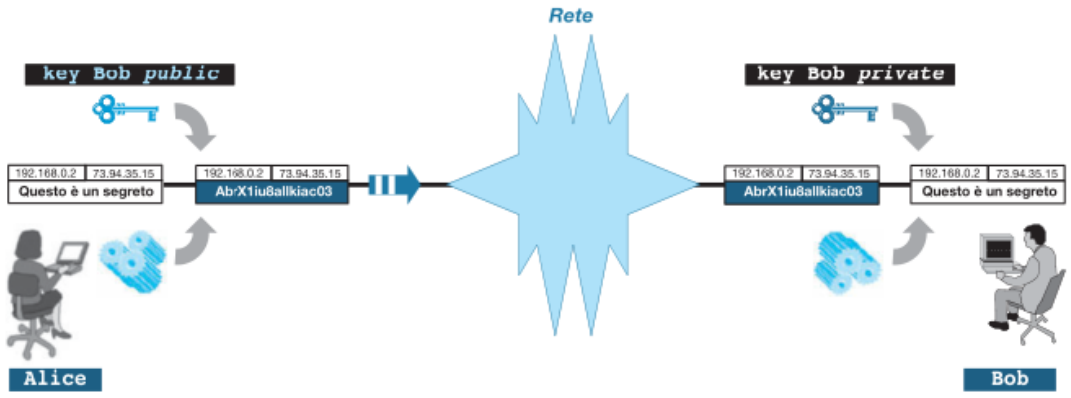
La crittografia asimmetrica nasce fondamentalmente per risolvere alla radice il problema dello scambio della chiave: con questa tecnica non c'è scambio di chiave segreta.

Gli utenti che usano la crittografia asimmetrica sono tutti dotati di due chiavi: la chiave privata e la chiave pubblica.

Se Alice vuole mandare un messaggio cifrato a Bob, usa la chiave pubblica di Bob per cifrare il messaggio, quindi lo invia.

Quando Bob riceve il messaggio, egli può decrifrarlo solo con la sua chiave privata. Nessun altro è in grado di decifrare il messaggio, neppure chi conosce la chiave pubblica di Bob: solo con la chiave privata di Bob si decifra il messaggio.

cifratura asimmetrica



In altri termini ogni utente mantiene segreta la propria chiave privata, e non ha necessità di scambiarla. Le chiavi pubbliche, invece, devono essere scambiate senza problemi, dato che non servono per decifrare, ma solo per cifrare.

Nel caso della crittografia a chiave asimmetrica gli algoritmi, anch'essi noti, si basano sulla caratteristica di alcune operazioni matematiche di essere *difficilmente invertibili*.

Gran parte delle operazioni matematiche comunemente conosciute sono facilmente invertibili: dati due addendi si calcola facilmente la loro somma, così come dato un solo addendo e una somma, si trova altrettanto facilmente il secondo addendo. In alcuni casi però non è così: un verso dell'operazione è facile; il verso opposto è estremamente difficile.

Per esempio, la fattorizzazione di un numero in numeri primi è un'operazione che si inverte con estrema difficoltà.

Se si scelgono due numeri primi molto grandi (**p** e **q**) e si moltiplicano tra loro, si ottiene un numero anch'esso molto grande ($\mathbf{m} = \mathbf{p} \cdot \mathbf{q}$) di cui si conosce, evidentemente, la fattorizzazione (**p** e **q**).

Quando **m** è veramente molto grande, per esempio un numero con trecento cifre, si può renderlo pubblico senza temere che vengano scoperti **p** e **q**: chiunque tentasse di fattorizzare **m** impiegherebbe molti anni con un buon calcolatore prima di scovare che $\mathbf{m} = \mathbf{p} \cdot \mathbf{q}$.

«Craccare» carte magnetiche

Nel gergo informatico il termine «craccare» indica l'operazione per cui una comunicazione segreta viene violata da un intruso (*forzatura*).

Alcune carte a banda ottica e/o magnetica contengono la chiave o i dati di login per accedere a sistemi protetti. Strisciando la carta su un dispositivo lettore collegato al PC, i dati di accesso contenuti nella carta vengono inviati al software che li attende per attivare il sistema protetto.

Sotto Windows, in alcuni casi, si può tenere aperto, anche iconizzato, **Blocco Note** mentre si striscia la carta: magicamente i dati segreti della carta compariranno nel documento di Blocco Note.

ESEMPIO

Crittografia con fattorizzazione (RSA)

Calcolare la chiave pubblica e la chiave privata di un utente affinché sia possibile cifrare in modo sicuro il codice Ascii di zero (48) con lo schema asimmetrico RSA.

Prima di tutto si individuano due numeri primi **p** e **q** a piacere, per esempio 13 e 7.

Quindi se ne calcola il prodotto $\mathbf{n} = \mathbf{p} \cdot \mathbf{q} = 13 \cdot 7 = 91$.

Ora si calcola un particolare valore $\Phi(\mathbf{n}) = (\mathbf{p} - 1) \cdot (\mathbf{q} - 1)$

$$\Phi(\mathbf{91}) = (13 - 1) \cdot (7 - 1) = 12 \cdot 6 = 72$$

Serve ora un numero **e** con la proprietà che $\Phi(\mathbf{n})$ ed **e** siano primi tra loro (senza fattori in comune).

Se si sceglie 5, il numero 72 e il numero 5 non hanno fattori comuni, quindi $\mathbf{e} = 5$.

I due numeri $\mathbf{e} = 5$ ed $\mathbf{n} = 91$ sono la **chiave pubblica**.

Infine il passaggio più noioso: trovare un numero **d** per cui $\mathbf{e} \cdot \mathbf{d} = 1 + \mathbf{k} \cdot \Phi(\mathbf{n})$, con un **k** a piacere.

Dopo qualche prova, calcolando l'espressione con qualche valore di **k** (per esempio, $\mathbf{k} = 1$, $\mathbf{k} = 2$, ecc.), risulta che per $\mathbf{k} = 2$ il numero **d** vale 29.

Infatti: $\mathbf{d} = (1 + \mathbf{k} \cdot 72)/5$; se $\mathbf{k} = 2$, allora $\mathbf{d} = (1 + 2 \cdot 72)/5 = (1 + 144)/5 = 145/5 = 29$.

Il numero $\mathbf{d} = 29$ sarà la **chiave privata**.

L'algoritmo per cifrare e decifrare è semplice e può essere reso noto.

Per cifrare un byte **b** in un byte **c** si calcola: $\mathbf{c} = \mathbf{b}^{(\text{public key0})} \bmod (\text{public key}_1)$

Per decifrare un byte **c** e ottenere **b**, si calcola: $\mathbf{b} = \mathbf{c}^{(\text{private key})} \bmod (\text{public key}_1)$

Ora si può cifrare il codice Ascii di zero (48) con la chiave pubblica:

$\mathbf{c} = \mathbf{b}^{(\text{public key0})} \bmod (\text{public key}_1) = 48^5 \bmod 91 = 254803968 \bmod 91 = \mathbf{55}$

Quindi si può decifrare il messaggio (55) con la chiave privata per riottenere il codice Ascii di zero:

$\mathbf{b} = \mathbf{c}^{(\text{private key})} \bmod (\text{public key}_1) = 55^{29} \bmod 91 = 2{,}954731317556447488096424760O939e + 50 \bmod 91 = \mathbf{48}$

In definitiva:

testo in chiaro (*plaintext*): **48**

testo cifrato (*ciphertext*): **55**

Il resto

L'operatore **mod** restituisce il resto della divisione tra due numeri interi.

Per esempio, se A = 10 e B = 3:

A mod B = 10 mod 3 = 1

perché «10 diviso 3 fa 3 con il resto di 1».

L'operatore è sempre definito nelle librerie di tutti i linguaggi di programmazione.

Nei linguaggi C/C++/Java è indicato con il simbolo della percentuale (%):

```
int M;
M = 10 % 3;  // M vale 1
```

In altri termini un utente (per esempio Bob) può consegnare, senza proteggerlo, il numero **m** ad Alice. Il numero **m** contiene il segreto (**p** e **q**), ma nessuno riesce a ottenerlo. Alice usa **m** per crittografare e Bob, l'unico che conosce il segreto, può decifrare.

Il numero **m** sarebbe la **chiave pubblica** di Bob; i numeri **p** e **q** sono la **chiave privata** di Bob.

ATTENZIONE. Affinché anche Alice possa ricevere messaggi cifrati da Bob, è necessario che possegga anch'essa una coppia di chiavi (una pubblica, da consegnare a Bob, e una privata, da non consegnare a nessuno). Rispetto alla cifratura con chiave privata ora Alice e Bob hanno bisogno di 4 chiavi per scambiarsi messaggi segreti, invece di una sola. La crittografia simmetrica evita lo scambio delle chiavi con il costo di dover usare 4 chiavi invece di una.

Il modo di crittografia asimmetrica utilizzato nell'esempio alla pagina precedente è noto come **RSA** (dalle iniziali dei suoi autori *Ronald Rivest*, *Adi Shamir* e *Leonard Adleman*, 1977).

Gli standard di cifratura asimmetrica più utilizzati in ambito informatico sono il già citato **RSA**, **PGP** (*Pretty Good Privacy*) e **OpenPGP**.

Lo standard RSA è tuttora (2013) inviolato.

Come si può notare, stabilire una coppia di chiavi asimmetriche (privata e pubblica) può costituire un ostacolo dal punto di vista computazionale: più i due numeri **p** e **q** sono grandi, maggiore è la sicurezza, ma anche maggiore è il tempo impiegato per il calcolo della chiave pubblica **m**. Questo tuttavia non è necessariamente un problema: dopotutto l'assegnazione delle chiavi a un utente viene eseguita una sola volta per tutte.

La questione rimane aperta nel momento in cui si cifra e decifra un intero documento con chiavi asimmetriche. Per esempio, calcolare potenze con esponenti molto grandi è un'operazione computazionalmente molto onerosa. Questo è un problema, dato che cifrare e decifrare sono operazioni che devono essere ripetute molte volte per ogni messaggio.

La crittografia a chiave asimmetrica ha alcune caratteristiche che vanno ricordate:

- **è lenta**: cifrare e decifrare numerosi blocchi di bit implica tempi inaccettabili. Viene quindi usata per cifrare messaggi brevi, come per esempio la chiave segreta da utilizzare per una cifratura simmetrica. La chiave simmetrica protetta dalla cifratura asimmetrica è detta **chiave di sessione**;
- implica un **meccanismo di distribuzione delle chiavi pubbliche** dei destinatari, in modo che si possano cifrare i messaggi per il destinatario desiderato;
- ha la proprietà che, cifrando un messaggio con la propria chiave privata, il messaggio può essere decifrato solo con la propria chiave pubblica: questo è ideale per **autenticare il mittente di un messaggio**.

Lo zaino

Lo studioso *Ralph Merkle* nel 1978 propose di utilizzare, come funzione *difficilmente invertibile* per un algoritmo a chiave asimmetrica, quella dedotta dal ***problema dello zaino***.

Promise quindi 100$ a chi avrebbe violato il suo algoritmo. *Adi Shamir*, la «A» di RSA, lo forzò immediatamente e intascò la ricompensa.

Non contento, Merkle rinforzò il suo algoritmo e promise 1000$ a chiunque lo avrebbe forzato. *Ron Rivest*, la «R» di RSA, lo forzò immediatamente e incassò la ricompensa. (*Andrew Tanenbaum, Reti di computer, 1997*)

ESEMPIO

Calcolo delle chiavi RSA (Linux)

Utilizzando l'applicazione *openssl* si può generare in un file una coppia di chiavi pubblica e privata con RSA:

```
linux:~# openssl genrsa -des3 -out itisRSA.key 2048
Generating RSA private key, 2048 bit long modulus
...................+++
...................+++
e is 65537 (0x10001)
Enter pass phrase for itisRSA.key: (password)
Verifying - Enter pass phrase for itisRSA.key: (password)
linux:~#
```

La creazione non è immediata nemmeno su una macchina ad alte prestazioni: le chiavi sono a 2048 bit.

Il file risultato viene cifrato con **3DES** e una chiave derivata dalla password inserita dall'utente.

Ora si può visualizzare il contenuto del file:

```
linux:~# cat itisRSA.key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,598A999A83DBC95A

IjmHjsD07Xf09XUbYbmPzhqM7SGWu8Cc1v6Km4RBR0o6D1PqFcDFJxOaGbAcAmQK
BBluOzqCB0yBnxCK2ICdnY3WUGN1veqXZQNWGhNP/hHIwyrReuHkzqZHerJxNE9x
290qMDfGPGtZg9CEAHnw3FEr/h0pYdTL/OjShweCKBmypvyj1II4GKSqCzhcl+zq
PPER0Z3vgqrdTYhRH94ixt+agPV+lAvtLASBPWZ+0rFV5LbMXds+sFg6qr1sbOUH
a2aXuGRBGgwyaKaP4cBT86GVd0F1Qt+g2YOSimzmvP+m81liB2hQvnroYHjRroJf
bMp8i+fBOTQja5F5RCFE2g4A6TygVW7qgED6E4KoCkRnrMEHHReTRD5X6k6YHbaG
HL5Earb/jrVOudNPDXuDdQV4V4ebXlk1YvD4Dz7XN7iWiS9eyjUnHvPp0iU4x25I
UzvINgyYvvKKRJNfo/n4qYt0oMfOSztAIQMB+grgAThDN/bDSNLzbUNt4vHAruKt
YfdR0BBfWogNIanBNxdU2GOAsxyMtliCjRnU8LmCFfZ5N3x35uFMJFJvzqsB015R
9RaOVWe2oxSWY1wbo7HM9OHDBldHTonXm9pSPobYeE0AwAyXmbV/KeQqYrw11HvQ
UVPQWWUhWqxqxdPOC4Gj1x4nKk6vnnnG/vaPBDOSd90J+MYVpqfmenbt8eWvs0TL
59b5JBUFJ882IocH/TrSnPKBkkk+UPbtZNFePvLq9tH6opyRm30b2Lrg5jv91vAB
wqveQndhCBC6t2CsgHYVp8neWak3yc9/pGyxf1Vg5+DpT4ao1X6BaJsMjKL8zhfW
nspuLy8MIi5J9yE+GFj0AhvuUVyOGSKIEWdZiYnlMuvwo0GVNorDidXkPuhGhkce
RXkyy/tiIjXeE+7qZW6zZRUSA6NVoMHbXqrQ+OowWm/mIdwMgRmJQbLpGEgpo5+Q
i4kHDbP1BC6hEFSj1pabs35V3wNF5dO0dIFg9DlTKUKUzKQGZKZUf9aKpDOVFcmM
Z6GE4QQZ8jkkROHXkAX/ZoszAkNXfkjz2mU7P+a4XAQ6e0szON9mwsNyq/AuLx98
Zm2ZtUZBWldKdKqtIDg9hX+QPuvaWKT21ojvC7Qq2AqjXuwYHwbfFSdD9X5fpkJc
qPdGRp4crjai4gknORRsf293TcEbZShriTCCI7+1TlS/2NkyKzM/v5CWhpi6xnuo
wwdWTi4ulq4aAEIIG+VfSlyJEB5vhNHrNLv493M/AKopJc/22d53FRV3sfRaSLm5
JZ/dA5P4cfIEpbqetxde74FLVGd2BfooIxjxHug68nzdJGP794BAFFAfOxllyyGv
8NW0yY9B/iaN95Qm+D47/9spCwGJCtQhYRW0j1DzJdEuGDHwDlRYuwr9iXq2bl0Y
6uZ413sVlxAnBLvouYf4JxRMiR+Pl2ZyOr2fIlAeeAvZXkvlmJb1UhD96vqVWTAP
lnR9D9CbdhlsZFwgl+Mwo/1OSE9dfyhw5DyrD7/B8df2nffASngniaGv+LBXLF3Y
EYF4akBMjB3NR3crMCvZHhDoXmDrBCqQdz0bI3ubeWoFiTWDxzMRug==
-----END RSA PRIVATE KEY-----
linux:~#
```

ESEMPIO

Più interessante visualizzare il contenuto tramite *openssl*:

```
linux:~# openssl rsa -in itisRSA.key -noout -text
Enter pass phrase for itisRSA.key: (password)
Private-Key: (2048 bit)
modulus:
    00:c7:62:3b:8c:8c:4a:5d:7f:08:1d:51:96:e9:1b:
    3e:92:ab:a8:97:4f:de:c9:a0:42:c3:61:bf:72:48:
    9e:2d:78:ea:f0:3b:ba:0e:e9:02:2f:9c:14:07:9f:
    fd:37:a0:a2:22:e2:c7:b8:a7:ec:eb:6b:e3:81:da:
    17:0f:dd:e9:90:6e:aa:4a:e0:8e:4c:f0:b1:2b:2a:
    41:0e:65:e0:b4:c0:29:e2:61:86:8b:09:ea:00:15:
    ad:38:5a:8f:92:83:28:28:67:ec:69:47:3e:98:b5:
    a8:6f:ef:ae:3e:bb:81:80:9d:83:c2:89:a4:77:c7:
    17:eb:01:1d:69:36:20:33:86:69:8f:9c:f0:dc:cf:
    c2:38:e7:27:86:28:85:9d:36:86:e1:2c:77:ba:97:
    e6:a4:a8:8c:0f:8e:2e:d0:45:d6:5f:a3:53:bd:c2:
    10:19:80:d3:33:8a:0e:2a:4c:3c:98:74:cb:c7:48:
    10:a9:09:0d:44:e3:79:47:d9:2a:08:38:eb:7e:4f:
    f1:58:96:c8:2f:8e:70:6e:37:10:02:7e:f9:82:16:
    c0:7e:a2:9f:07:76:4e:65:27:c6:4b:1a:12:1a:e5:
    49:ef:ee:e6:fc:7d:4b:cd:22:64:ac:ac:a0:d6:31:
    a1:c8:18:01:ad:9e:ef:c9:4e:06:c6:96:85:d4:90:
    0a:e1
publicExponent: 65537 (0x10001)
privateExponent:
    63:c8:17:81:29:1c:76:5a:02:97:99:a3:6a:99:85:
    e1:25:23:44:46:66:7a:85:47:a4:3c:20:f1:72:c2:
    26:83:a3:20:02:e4:04:5e:3c:07:d3:96:7a:92:68:
    c9:14:0c:d0:64:aa:0b:11:8f:11:ea:76:7b:1f:c7:
    f6:da:d9:ee:bc:53:61:11:ac:65:78:f7:51:60:de:
    19:f4:86:56:2e:ed:47:2c:03:87:45:b8:e3:bd:f5:
    68:84:79:e1:9a:dd:d8:0a:da:57:7d:9e:28:12:91:
    6f:23:86:12:43:08:76:73:5d:e3:57:bb:05:6e:8f:
    db:be:3d:17:d0:4c:a1:3b:ba:1d:21:19:30:cb:7c:
    14:a0:dc:17:4f:83:a2:99:2a:c0:e8:3d:a4:db:76:
    bc:d4:34:70:5e:21:02:32:cb:ae:d7:ec:43:af:46:
    e2:f9:4f:e0:a9:b5:dd:d6:e0:26:8f:0c:97:2f:cc:
    21:0b:70:2c:8c:8d:bd:b2:78:44:1b:d3:97:5b:65:
    21:e6:4e:6d:f0:93:a6:7d:6e:f4:be:0a:16:5e:09:
    92:70:24:95:4a:ca:97:e2:36:eb:71:a6:ae:0f:2a:
    79:25:75:8e:b3:49:23:26:d9:10:e4:12:36:d8:82:
    81:d0:72:a2:66:dc:0f:70:ca:e2:29:02:65:33:32:
    a1
prime1:
    00:f8:cd:68:a5:1d:91:f9:d8:57:f4:21:4c:bb:de:
    87:65:11:3b:49:40:78:28:9f:92:ee:b1:99:6a:ac:
    54:16:d0:c7:21:66:02:68:8f:d4:c5:86:46:1e:f3:
```

ESEMPIO

```
    a2:a6:64:73:87:75:1a:67:98:e4:50:62:0c:b7:de:
    e5:47:c4:4b:9b:5f:08:bd:af:1e:71:0d:11:44:5f:
    f3:0b:90:2e:b1:bb:16:0d:34:19:db:ea:2e:27:96:
    c3:a4:e8:c2:0f:73:fd:0a:11:3c:71:6e:bc:a3:19:
    41:bd:30:c5:de:f8:38:45:fd:27:3a:76:cc:65:e5:
    1f:08:63:31:e0:12:94:43:8d
prime2:
    00:cd:26:d6:49:24:10:cd:2b:35:d2:e7:22:0b:63:
    12:ff:b3:c9:ca:9b:55:be:2c:76:80:1f:aa:3a:db:
    77:20:88:da:64:8c:c4:25:57:af:5f:32:35:99:83:
    a6:0f:0c:d0:0d:8e:8a:bc:9d:e0:62:78:0e:53:ce:
    23:bf:1f:01:c7:ec:d5:0d:6f:d6:f8:4c:39:60:c3:
    c7:4e:c8:8a:14:92:30:d4:21:e2:db:f4:96:f0:91:
    c0:ba:13:3d:68:a3:95:56:3c:d4:88:29:12:91:d4:
    5d:11:e3:7c:34:a1:3e:24:f7:24:82:31:4c:d8:4d:
    34:ac:68:b3:9e:23:59:c5:a5
exponent1:
    00:96:a3:c7:b8:31:2f:31:16:cc:2a:03:ff:71:c0:
    4a:39:e7:34:fe:25:0a:9b:8e:02:68:83:1f:60:76:
    f6:72:d9:f5:b7:43:0c:32:42:e3:90:b4:bb:c0:01:
    c3:78:fb:58:f7:aa:ef:51:ca:40:72:6a:eb:48:68:
    ac:69:c7:6f:ff:a2:8a:a8:4e:5f:20:13:c9:60:9c:
    b7:8b:48:c0:fc:db:49:7e:b5:0c:f3:19:d6:d8:21:
    70:53:68:9a:16:c1:23:73:f4:fb:a3:b2:68:84:57:
    c6:75:c6:12:07:ee:42:24:1e:22:a2:43:4b:7e:66:
    3b:63:d8:ab:59:ff:e5:c5
exponent2:
    00:a3:75:80:63:c2:a2:c8:76:d7:69:f5:d3:c0:72:
    ee:5e:62:e8:33:d0:d4:de:b4:1a:af:37:8b:b1:5d:
    d0:6b:51:df:81:22:4f:de:d9:20:d8:9e:ee:ea:24:
    65:19:b4:c1:c9:2b:7c:0b:91:57:89:dd:d2:bc:9f:
    91:07:e5:32:cc:13:3e:26:78:a8:36:2a:b5:c5:0d:
    f9:2e:22:c7:32:60:d1:1b:14:ec:e7:08:d9:83:50:
    fe:d8:c4:1f:b7:d2:2f:59:09:1a:e6:6a:a3:6b:22:
    64:0d:ae:cd:f6:39:4b:84:b4:8e:98:55:a3:be:ec:
    b5:3d:72:27:3b:a7:3b:0e:29
coefficient:
    4c:08:15:e6:d6:9b:0b:42:a9:10:3c:1a:78:9b:9b:
    74:99:8c:b3:c1:6a:c4:d3:ea:af:d5:2a:ae:8b:78:
    a6:10:62:94:b1:7f:87:98:f2:a6:77:bc:f4:3c:13:
    10:3e:ce:94:bd:64:9d:18:7c:cd:5e:41:52:04:60:
    d9:ac:60:c7:a4:5f:5b:f8:53:19:81:a5:f9:17:f1:
    67:88:a9:c1:21:2e:d9:7f:3b:f7:e5:12:56:20:42:
    7f:0c:d1:23:95:78:a9:e4:d9:cb:dd:fb:7f:b1:e1:
    b4:59:8b:20:64:73:e4:02:c9:01:dc:ee:64:a1:ae:
    de:47:20:36:e1:a6:5b:3a
linux:~#
```