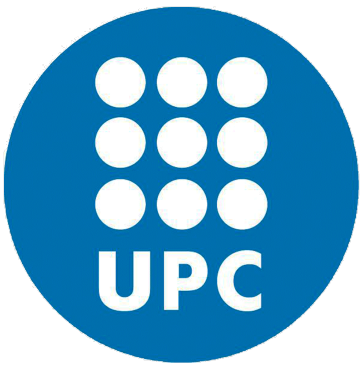
Criptografía

Temps d'execució amb TCR i sense

Juli Sahun Montejano
Víctor Pla Sanchis



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Comparació de temps entre firmes amb TCR i sense

Hem executat 100 firmes de missatges diferents tant amb el mètode del teorema xinès del residu per calcular el missatge encriptat com sense. Els resultats de l'execució es mostren a les següents figures.

METODE \ BITS	TCR	NO TCR
512	0.002s	0.007s
1024	0.012s	0.043s
2048	0.088s	0.413s
4096	0.679s	3.043s

Figura 1: Taula sobre el temps d'execució de 100 firmes diferents amb TCR i sense.

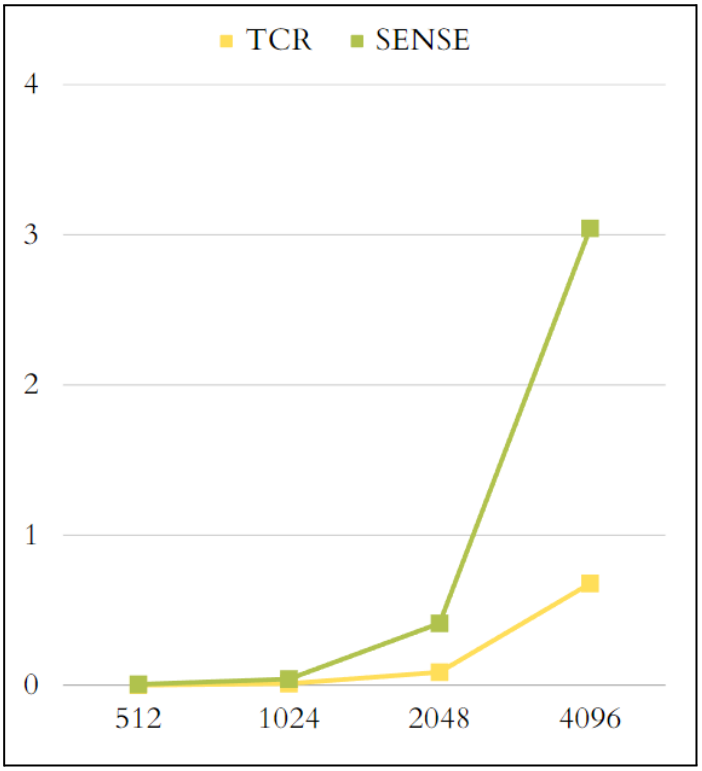


Figura 2: Temps d'execució de 100 firmes diferents amb TCR i sense. En groc és veu la progressió del temps amb el mètode TCR (TCR), en verd sense (SENSE).

Podem observar en la figura 2 com amb dos mètodes es comporten de manera exponencial, això és degut a que augmentem el número de bits de manera també exponencial (sempre com a potències de 2). Tot i això, els mètodes tant amb TCR com sense TCR tenen un cost logarítmic O(logn) en les seves operacions.
És molt notable la diferència entre les dues metodologies, de fet, obtenim un *speedup* del 125,9% de mitjana utilitzant el teorema xinès del residu.

Si ens fixem en les diferències que existeixen entre els dos mètodes podem trobar el perquè d'aquesta diferència tan notable. Si ho volem fer de la manera lenta fem la següent operació:

En codi:

```
    def sign_slow(self, message):
    return pow(message, self.privateExponent, self.modulus)
```

Matemàticament:

Sigui a el misatge a firmar, d l'exponent privat de l'usuari i m el modul en el que treballem.

$$a^d \mod m$$

Per altre banda, amb el teorema xinès del residu tenim:

En codi:

```
    def sign(self, message):
a = pow(message, self.privateExponentModulusPhiP, self.primeP)
b = pow(message, self.privateExponentModulusPhiQ, self.primeQ)
qq = self.inverseQModulusP * self.primeQ
pp = 1 - qq
return pp*b + qq*a
```

Matemàticament:

Sigui m el misatge a firmar, m' el misatge firmat, (p, q) la parella de primers escollits, $(\phi(p), \phi(q))$ la parella de phi's dels primers.

$$a = m^{\phi(p)} \mod p$$
$$b = m^{\phi(q)} \mod q$$
$$t = (q^{-1} \mod p) \cdot q$$
$$m' = t \cdot b + (1 - t) \cdot a$$

Tal com veiem, sembla que inicialment el TCR genera més operacions que fent-ho directament i que així en el fons trigui més, tot i això, hem de fixar-nos en el costós que és cada pas de les dues metodologies.
És evident que el cost del mètode TCR no recau en el càlcul de la variable *t* ni de la variable *m'* ja que són multiplicacions i sumes de números de mida de l'orde del missatge a encriptar, i per tant, on hauríem de fixar-nos és en el càlcul de *a* i *b*.
És també fàcil veure que tant *ϕ(p)* com *ϕ(q)* són números molt més petits que *d*, i és per això doncs que evaluar exponents en el TCR és més ràpid que directament amb l'exponent privat, ho podem verificar a la figura 3.

```
primeP:  1232
primeQ:  1233
phi:  2465
privateExponent:  2464
modulus:  2465
privateExponentModulusPhiP:  1232
privateExponentModulusPhiQ:  1233
inverseQModulusP:  1231
```

Figura 3: Mida en dígits de cada variable de la classe *RSA_key* amb primers de 4096 bits

Finalment, podem veure com TCR utilitza totes aquelles variables que tenen un nombre de dígits més petits, i en canvi, sense TCR s'utilitzen aquelles variables amb nombre de dígits més grans.

TCR:
- primeP
- primeQ
- privateExponentModulusQ o bé privateExponentModulusP
- inverseQModulusP
-

Sense TCR:
- modulus
- privateExponent