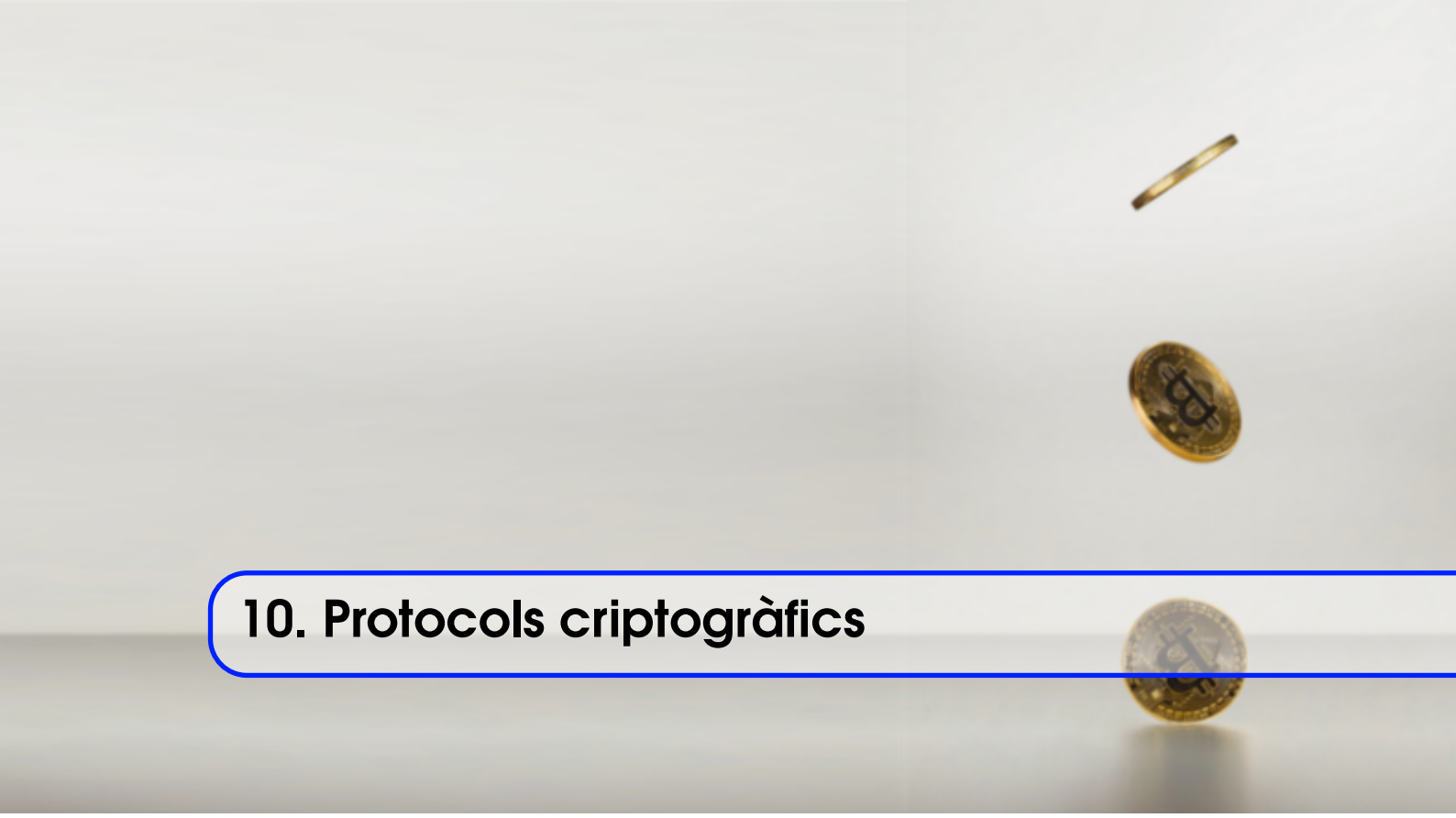


10. Protocols criptogràfics

Més enllà dels mecanismes per xifrar i desxifrar missatges el cert és que la criptografia permet construccions més elaborades que continuen tenint el mateix objectiu que els criptosistemes: protegir la informació. Així, ens podem trobar diferents situacions on calguin protocols que ens garanteixin un seguit de propietats de seguretat que els criptosistemes per si sols no poden proporcionar. És en aquest punt on intervenen els protocols criptogràfics, protocols entre dos o més usuaris que utilitzen mecanismes criptogràfics per protegir la informació.

En aquest capítol estudiarem diversos protocols criptogràfics cada un d'ells amb un propòsit diferent. Llevat de l'esquema de compartició de secrets, els protocols descrits en aquest capítol són protocols en els que hi intervenen dos usuaris i no es contempla l'existència de cap tercera part de confiança. Així, les operacions es realitzen, sovint de forma conjunta, entre els dos usuaris per aconseguir l'objectiu del protocol. La suposició que es fa en tot moment és que els usuaris poden actuar de forma deshonesta de manera que és important que els propis protocols incorporin els mecanismes de seguretat necessaris per tal que, en cas que una part actuï de forma maliciosa, l'altra part no se'n vegi afectada o, com a mínim, pugui detectar l'engany.

10.1 El protocol de tres passos de Shamir

El protocol de tres passos de Shamir, va ser proposat per A. Shamir tot i que no el va publicar mai. El protocol permet establir una comunicació secreta entre dues parts sense cap intercanvi previ de claus. La base del protocol és una funció de xifratge commutativa respecte a les claus. És a dir, serà el mateix xifrar un missatge m amb una clau k_1 i el resultat tornar-lo a xifrar amb una clau k_2, que xifrar-lo primer amb la clau k_2 i el resultat xifrar-lo amb la k_1, és a dir:

$$E_{k_1}(E_{k_2}(m)) = E_{k_2}(E_{k_1}(m))$$

Passem a descriure el **protocol de tres passos de Shamir** en el qual l'Alice vol fer arribar el missatge m al Bob. Per fer-ho, l'Alice disposarà d'una clau per xifrar, k_A^e i una clau per desxifar k_A^d i el Bob també tindrà una clau per xifrar k_B^e i una per desxifrar k_B^d. Denotarem per $E_{k_A^e}(m)$ l'acció de xifrar el missatge m amb la clau de xifrat k_A^e de l'Alice. Igualment, denotarem per $D_{k_A^d}(c)$ el desxifrat del missatge c amb la clau de desxifrat k_A^d de l'Alice.

En l'esquema de la Taula 10.1 es poden veure els diferents passos del protocol i la informació que s'intercanvien els usuaris que hi participen.

Taula 10.1: Esquema de 3 passos de Shamir

Pas	Alice		Bob
1.	Calcula $c_1 = E_{k_A^e}(m)$	$\xrightarrow{c_1}$	
2.		$\xleftarrow{c_2}$	Calcula $c_2 = E_{k_B^e}(c_1) = E_{k_B^e}(E_{k_A^e}(m))$
3.	Calcula $c_3 = D_{k_A^d}(E_{k_B^e}(E_{k_A^e}(m))) =$ $= D_{k_A^d}(E_{k_A^e}(E_{k_B^e}(m))) = E_{k_B^e}(m)$	$\xrightarrow{c_3}$	
4.			Calcula $m = D_{k_B^d}(c_3) = D_{k_B^d}(E_{k_B^e}(m))$

Com podem veure, al final del protocol l'Alice ha fet arribar a Bob el missatge m de forma segura ja que en cap dels missatges que s'han intercanviat en cada un dels tres passos el missatge m no ha viatjat en clar. Així, un atacant que estigui analitzant les comunicacions entre A i B no podrà extreure cap informació de m. Noteu, a més, que en cap moment s'ha produït un intercanvi de claus. L'Alice només coneix k_A^e i k_A^d i en Bob només k_B^e i k_B^d.

En els següents apartats veurem alguns criptosistemes que tenen la propietat de commutativitat de claus i quins resultats presenten quan s'utilitzen com a esquema de xifrat en el protocol de tres passos de Shamir.

10.1.1 El xifrat de Vernam i el protocol de tres passos de Shamir

Un dels criptosistemes que hem presentat en aquest llibre és el criptosistema de Vernam, que seria el criptosistema més segur que existeix ja que, utilitzant una bona clau, ens aporta seguretat incondicional. Si recordem, el mecanisme tant de xifrat com de desxifrat d'aquest criptosistema és molt simple. Donat un missatge m expressat en bits, i una clau k de la mateixa mida que el missatge també expressada en bits, la funció de xifrat consisteix en fer una XOR entre el missatge i la clau, és a dir $E_k(m) = m \oplus k = c$. D'altra banda, per desxifrar el missatge c simplement haurem de fer de nou una XOR amb la mateixa clau k amb la que hem xifrat $D_k(c) = c \oplus k = m$.

Si ens hi fixem, aquest criptosistema presenta commutativitat de claus, ja que si tenim dues claus k_1 i k_2 es compleix que:

$$E_{k_1}(E_{k_2}(m)) = (m \oplus k_2) \oplus k_1 = (m \oplus k_1) \oplus k_2 = E_{k_2}(E_{k_1}(m))$$

ja que l'operació XOR és commutativa.

Així, si utilitzem el criptosistema de Vernam per al protocol de tres passos de Shamir entre A i B tenim que les claus de xifrar i desxifrar per a cada usuari són la mateixa, és a dir, $k_A^e = k_A^d = k_A$ i $k_B^e = k_B^d = k_B$ i en els tres intercanvis d'informació del protocol es generaran els missatges mostrats en l'esquema de la Taula 10.2.

Taula 10.2: Esquema de tres passos de Shamir amb el xifrat de Vernam

Pas	Alice		Bob
1.	Calcula $c_1 = m \oplus k_A$	$\xrightarrow{c_1}$	
2.		$\xleftarrow{c_2}$	Calcula $c_2 = c_1 \oplus k_B$
3.	Calcula $c_3 = c_2 \oplus k_A = m \oplus k_B$	$\xrightarrow{c_3}$	
4.			Calcula $m = c_3 \oplus k_B$

Tot i que aparentment hem aconseguit desenvolupar el protocol correctament utilitzant un dels criptosistemes més segurs que hi ha, el problema està en que un atacant que pugui veure la comunicació en té prou en prendre nota dels tres missatges xifrats que s'intercanvien l'Alice i en Bob, ja que un cop intercepta c_1, c_2 i c_3 per obtenir el missatge xifrat m només cal que faci una suma XOR dels tres:

$$c_1 \oplus c_2 \oplus c_3 = (m \oplus k_A) \oplus (m \oplus k_A \oplus k_B) \oplus (m \oplus k_B) = m$$

Per tant, podem concloure que alhora d'utilitzar un criptosistema per al protocol de tres passos de Shamir no en tindrem prou en assegurar-nos que compleixi la commutativitat de les claus sinó que caldrà anar en compte sobre la relació que tenen els missatges una vegada han estat xifrats.

Aquest fet ens fa veure que, més enllà d'aquest exemple concret, en la creació de protocols criptogràfics és important no només que cada una de les eines criptogràfiques que s'utilitza sigui segura sinó que a més, la seva combinació ho continuï essent, fet que com hem vist, no sempre succeeix.

10.1.2 El criptosistema d'exponenciació

Un altre esquema amb commutativitat de claus el va proposar el mateix A. Shamir. Aquest sistema es basa amb l'exponenciació i la seva seguretat recau en la dificultat del càlcul del logaritme discret. És un criptosistema semblant amb l'RSA però no s'ha de confondre amb l'RSA ja que en aquest cas les dues claus que s'utilitzen, una per xifrar i l'altra per dexifrar, són dues claus secretes que únicament estan en possessió d'un sol usuari.

En primer lloc es tria un paràmetre per a l'intercanvi, un primer p gran. Totes les operacions es realitzaran al cos $\mathbb{Z}_p$. L'Alice genera les seves claus de la següent manera. Tria com a clau de xifrat k_A^e un valor aleatori i com a clau de desxifrat calcula el valor k_A^d tal que $k_A^e \cdot k_A^d = 1 \pmod{p-1}$. La funció de xifrat per a un missatge m serà $E_{k_A^e}(m) = m^{k_A^e} \pmod{p}$. La funció de desxifrat d'un missatge c serà $D_{k_A^d}(c) = c^{k_A^d} \pmod{p}$. De la mateixa manera, el Bob generarà les seves claus k_B^e i k_B^d i utilitzarà les mateixes funcions de xifrat i desxifrat. Amb aquestes condicions el protocol queda descrit en l'esquema de la Taula 10.3.

Taula 10.3: Esquema de tres passos de Shamir amb el xifrat d'exponenciació

Pas	Alice		Bob
1.	Calcula $c_1 = m^{k_A^e} \pmod{p}$	$\xrightarrow{c_1}$	
2.		$\xleftarrow{c_2}$	Calcula $c_2 = (c_1)^{k_B^e} \bmod p$
3.	Calcula $c_3 = (c_2)^{k_A^d} \bmod p = m^{k_B^e} \bmod p$	$\xrightarrow{c_3}$	
4.			Calcula $m = (c_3)^{k_B^d} \bmod p$

Fixeu-vos que, en aquest cas, un atacant que intercepti els tres missatges de la comunicació, c_1, c_2 i c_3 no podrà obtenir cap informació sobre el missatge transmès ja que les claus per xifrar només les coneixen A i B.

Exemple 10.1 Exemple de protocol de tres passos de Shamir amb el criptosistema d'exponenciació. En aquest exemple suposarem que els dos usuaris treballen amb el paràmetre $p = 131$. A més, l'usuari A disposarà de la clau de xifrat $k_A^e = 21$ i de la clau de desxifrat $k_A^d = (k_A^e)^{-1} \pmod{p-1} = 31$. D'altra banda, l'usuari B també tindrà el seu parell de claus. La de xifrat serà $k_B^e = 27$ i la de desxifrat $k_B^d = (k_B^e)^{-1} \pmod{p-1} = 53$.

Amb aquests paràmetres, l'usuari A vol enviar de forma secreta el missatge $m = 15$ a B i per fer-ho els passos del protocol seran els següents:

Pas	Alice		Bob
1.	$c_1 = 15^{21} \pmod{131} = 125$	$\xrightarrow{125}$	
2.		$\xleftarrow{27}$	$c_2 = (125)^{27} \bmod 131 = 27$
3.	$c_3 = (27)^{31} \bmod 131 = 129$	$\xrightarrow{129}$	
4.			$m = (129)^{53} \bmod 131 = 15$

Exercici 10.1 Reproduïu el protocol de tres passos de Shamir per tal que A enviï el missatge $m = 20$ a B utilitzant el criptosistema d'exponenciació on la clau de xifrat d'A val $k_A^e = 19$ la clau de desxifrat d'A val $k_A^d = 79$ i les corresponents claus de xifrat i desxifrat de B valen $k_B^e = 13$ i $k_B^d = 77$ respectivament. Suposarem, també, que $p = 101$.

10.2 Esquemes de compartició de secrets

Quan volem emmagatzemar un secret cal tenir en compte que hi ha situacions en les que el secret no pot ser guardat de forma centralitzada perquè hi ha el perill que aquesta centralització esdevingui un punt feble en la seguretat. En aquestes situacions el concepte de centralització pot tenir diferents vessants. Per exemple, imaginem-nos que tenim el codi d'obertura d'una caixa forta però no volem que estigui custodiat per una sola persona perquè té el perill que aquesta persona pugui marxar amb tots els diners. Voldríem poder distribuir aquest codi de manera que més d'una persona fos necessària per a l'obertura de la caixa forta.

Una altra situació, potser més quotidiana, és l'emmagatzemament de contrasenyes. Si emmagatzemem la contrasenya en un únic lloc, si aquest lloc sofrís algun incident perdríem la clau. Podríem solucionar aquest problema guardant la mateixa clau en diferents llocs, però això implicaria una reducció de la seguretat ja que les probabilitats que algú la trobi són més grans. Al igual que el que hem fet amb la caixa forta, podríem repartir el valor de la clau en diferents fragments. Fixeu-vos que en aquest cas, la possibilitat de poder recuperar la clau només amb alguns fragments (i no necessàriament amb tots) és important ja que si els necessitem tots per recuperar-la, tornem a estar en el punt de partida: si un dels llocs on hi ha un dels fragments de la clau sofrís algun incident no podríem recuperar la clau i també l'hauríem perduda.

Per resoldre aquests tipus de situacions tenim els esquemes de compartició de secrets. Aquests esquemes van ser proposats de forma independent l'any 1979 per Adi Shamir i George Blakley.

> Un **esquema de compartició de secrets llindar** (m,n) (en anglès (m,n)-*threshold secret sharing scheme*), és un esquema que permet distribuir un secret en n fragments diferents de manera que si s'ajunten m o més fragments es pot recuperar el secret, però no és possible obtenir cap informació del secret si es disposen de menys d'm fragments.

Si assumim l'escenari en el qual volem repartir un secret S entre diferents usuaris, un esquema de compartició de secrets llindar (m,n) està format per n usuaris, $u_1, \cdots, u_n$. Cada usuari té el seu corresponent fragment s_i del secret S. A més cal que es compleixin les següents propietats:

1. Per a tot $i = 1, \cdots, n$, l'usuari u_i només coneix el seu fragment s_i.
2. El secret S es pot obtenir a partir d'm valors diferents s_i per a qualsevol $i \in \{1, \cdots, n\}$.
3. Donats $m-1$ valors diferents, s_i, no es pot obtenir cap informació d'S.

10.2.1 Esquema de compartició de secrets polinòmic

Un esquema per compartir secrets llindar (m,n) força utilitzat és el proposat per A. Shamir basat en la interpolació polinòmica.

Suposem que volem compartir el secret S utilitzant un esquema de llindar (m,n). Això vol dir que hem de crear n fragments i que en tenim prou en tenir-ne m per reconstruir-lo, però que menys d'aquesta quantitat no ens serà suficient. En aquest tipus d'esquema hi haurà un superusuari, el gestor, que serà l'encarregat de, partint del secret S, generar els n fragments. Com a paràmetres públics tindrem un nombre primer p tal que $p > n$ i $p > S$.

Per construir els fragments, el gestor construeix un polinomi $a(x)$ de grau $m-1$ amb coeficients a_i a $\mathbb{Z}_p$, és a dir $a(x) \in \mathbb{Z}_p[x]$. Aquest polinomi tindrà com a coeficients valors aleatoris, llevat del terme independent que

serà exactament el valor secret S, és a dir, podem expressar el polinomi de la següent manera:

$$a(x) = S + a_1 x + a_2 x^2 + \cdots + a_{m-1} x^{m-1} \pmod{p}$$

La generació dels fragments es realitzarà de la següent manera. El gestor tria n valors aleatoris de $\mathbb{Z}_p$, $\{x_1, \cdots, x_n\}$, i per a cada valor en calcula la seva avaluació pel polinomi, és a dir, $a(x_i) = S + a_1 x_i + a_2 x_i^2 + \cdots + a_{m-1} x_i^{m-1} \pmod{p}$.

El polinomi $a(x)$ es manté en secret i només el coneix el gestor, però es pot eliminar una vegada s'han generats els fragments.

Cada participant rep com a fragment del secret el parell $\{x_i, a(x_i)\}$, és a dir, un valor x_i i la seva avaluació en el polinomi, $a(x_i)$.

Podrem recuperar el secret si tenim m fragments plantejant el següent sistema d'equacions:

$$\begin{aligned}
a(x_1) &= S + a_1 x_1 + a_2 x_1^2 + \cdots + a_{m-1} x_1^{m-1} \pmod{p} \\
a(x_2) &= S + a_1 x_2 + a_2 x_2^2 + \cdots + a_{m-1} x_2^{m-1} \pmod{p} \\
&\vdots \quad \vdots \\
a(x_m) &= S + a_1 x_m + a_2 x_m^2 + \cdots + a_{m-1} x_m^{m-1} \pmod{p}
\end{aligned}$$

Si ens fixem, en aquest sistema hi tenim m incògnites corresponents als m coeficients dels polinomis $S, a_1, a_2, \cdots, a_{m-1}$ i també hi ha m equacions, per la qual cosa al resoldre'l obtindrem el valors de les incògnites i en particular la que ens interessa, que és el valor secret S. A més, aquest sistema sempre tindrà solució i serà única perquè hi intervé el determinant de Vandermonde.

Exemple 10.2 Exemple de protocol de compartició de secrets llindar (3,5)

Suposem que tenim cinc usuaris u_1, u_2, u_3, u_4, u_5 que volen repartir-se el valor secret $S = 673$. Per fer-ho utilitzaran l'esquema de compartició de secrets polinòmic de Shamir i treballaran amb el primer $p = 1931$.

Passem a descriure els dos processos d'un esquema de compartició de secrets: la generació dels fragments i la recuperació del secret.

Generació dels fragments:

Donat que amb 3 usuaris n'hi haurà prou per recuperar el secret, el gestor construirà un polinomi de grau 2 amb coeficients a $\mathbb{Z}_{1931}$ on el terme independent sigui el secret $S = 673$. Així, el gestor triarà dos valors aleatoris per crear el polinomi, per exemple 436 i 806 i construirà el polinomi $a(x) = 673 + 806x + 436x^2$. Amb aquest polinomi, procedirà a construir els fragments de cada usuari avaluant el polinomi en una component x per a cada participant. Si suposem que u_1 té la component $x = 1$, u_2 la component $x_2 = 2$ i així per a cada usuari, tindrem les següents avaluacions:

$a(1) = 673 + 806 \cdot 1 + 436 \cdot 1^2 = 1915 \pmod{1931}$
$a(2) = 673 + 806 \cdot 2 + 436 \cdot 2^2 = 167 \pmod{1931}$
$a(3) = 673 + 806 \cdot 3 + 436 \cdot 3^2 = 1222 \pmod{1931}$
$a(4) = 673 + 806 \cdot 4 + 436 \cdot 4^2 = 1218 \pmod{1931}$
$a(5) = 673 + 806 \cdot 5 + 436 \cdot 5^2 = 155 \pmod{1931}$

Per tant l'usuari u_1 rebrà el fragment $[1, 1915]$, l'usuari u_2 el fragment $[2, 167]$, l'usuari u_3 el fragment $[3, 1222]$, l'usuari u_4 el fragment $[4, 1218]$ i l'usuari u_5 el fragment $[5, 155]$.

Recuperació del secret:

Suposem ara que tres dels cinc usuaris es reuneixen per recuperar el secret. Suposem que són els usuaris u_1, u_4 i u_5 (però haguéssim pogut triar qualssevol tres altres). Els fragments d'aquests usuaris són $[1, 1915]$,

$[4,1218]$ i $[5,155]$ respectivament. Com que aquests valors són punts del polinomi utilitzat per generar els fragments, podem plantejar el següent sistema d'equacions:

$$\begin{aligned} S+a_1\cdot 1+a_2\cdot 1^2 &= 1915 \pmod{1931} \\ S+a_1\cdot 4+a_2\cdot 4^2 &= 1218 \pmod{1931} \\ S+a_1\cdot 5+a_2\cdot 5^2 &= 155 \pmod{1931} \end{aligned}$$

Com que només ens interessa resoldre el sistema per la variable S, que és el secret, podem aplicar el mètode de Kramer i obtenim:

$$\frac{\begin{vmatrix} 1915 & 1 & 1 \\ 1218 & 4 & 16 \\ 155 & 5 & 25 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 1 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \end{vmatrix}} = \frac{352}{12} = 352\cdot 161 = 673 \pmod{1931}$$

Exercici 10.2 Utilitzeu un esquema de compartició de secrets de Shamir per generar els fragments d'un sistema $(3,5)$-llindar per compartir el nombre secret 11. Preneu com a primer $p = 13$.

Exercici 10.3 En un esquema de compartició de secrets polinòmic de Shamir amb llindar $(3,6)$ els participants reben els següents fragments $(58,137),(11,48),(50,99),(80,50),(104,33),(39,114)$. Tenint en compte que treballen a $\mathbb{Z}_{149}$ recupereu el secret.

Exercici 10.4 En un esquema de compartició de secrets polinòmic de Shamir amb llindar $m = 3$, construït sobre $\mathbb{Z}_{13}$, l'usuari A té l'avaluació del polinomi per a $x = 1$, l'usuari B, $x = 2$ i l'usuari C, $x = 3$. Els tres usuaris es reuneixen per poder trobar la clau del sistema. Tots tres usuaris fan trampa; els usuaris A i C li sumen 2 a l'avaluació del polinomi en el seu punt però la clau que recuperen és la correcta. Quina és la trampa que ha fet l'usuari B?

10.2.2 Problemàtiques dels esquemes de compartició de secrets

A la pràctica, els esquemes de compartició de secrets tenen un seguit de restriccions que fan que el seu ús requereixi construccions molt més complexes que les que hem presentat aquí.

El primer punt a tenir en compte en un esquema de compartició de secrets és la confiança que es diposita en el gestor del sistema. Fixeu-vos que el gestor és el que s'encarrega de generar el polinomi que permetrà crear els fragments de cada participant i, per fer-ho, necessita el valor del secret. Per tant, cal que el gestor sigui una tercera part de confiança o bé que aquest procés es realitzi amb les garanties de seguretat necessàries.

D'altra banda, també ens podríem preguntar què passaria si un dels participants donés un valor aleatori en comptes del seu fragment. El cert és que el secret no es recuperaria i encara més, no sabríem qui ha estat el culpable. I encara pitjor, l'atacant podria utilitzar el secret recuperat erròniament, el seu fragment fals i el seu fragment correcte per recuperar el secret real sense l'ajut de la resta de participants mentre que la resta de participants continuarien sense poder recuperar el secret.

Per tal de resoldre aquests problemes hi ha els esquemes de compartició de secrets verificables, esquemes més elaborats que utilitzen mecanismes de compromís de bit que veurem més endavant.

Fixeu-vos, però, que totes aquestes problemàtiques no ens afecten quan només volem utilitzar l'esquema de compartició de secrets per emmagatzemar una contrasenya de forma distribuïda i segura ja que en aquest cas, tant el gestor com els usuaris que proporcionaran els fragments són tots el mateix.

10.3 Esquemes de compromís de bit

Hi ha situacions quotidianes en les que estem acostumats a fer servir alguns mecanismes molt simples que funcionen sense cap dificultat d'execució. Un d'aquests casos és el de 'tirar una moneda a l'aire' per, per exemple, decidir quin dels dos jugadors d'una partida d'escacs tindrà les fitxes blanques. Ara bé, quan les dues parts que duen a terme aquest petit protocol no es troben físicament al mateix lloc, la simplicitat de tirar una moneda a l'aire no ens serveix si hi ha certa desconfiança entre els dos participants.

Si analitzem el procés de tirar una moneda a l'aire veiem que, normalment, un dels dos usuaris tria cara o creu i l'altre, una vegada s'ha decidit qui guanyarà segons el revers de la moneda, tira la moneda a l'aire. En aquest simple esquema, l'usuari que tria cara o creu ho fa de forma pública, de manera que després (quan cau la moneda) no pot dir que ha triat una altra cosa. I l'usuari que tira la moneda no pot fer trampa (assumint que la moneda no està trucada!) perquè tira la moneda davant de l'altre usuari i els dos veuen el resultat que en surt, de manera que qui tira la moneda no pot canviar-ne el resultat.

Per emular aquest protocol de forma remota (o digital) es fa servir un esquema de compromís de bit.

> Un **esquema de compromís de bit** (en anglès, *bit commitment*) és una tècnica per la qual un usuari A es compromet, davant d'un usuari B, a un valor m per mitjà d'un valor $C(m)$, que serà el compromís. Aquest compromís ha de tenir les següents propietats:
>
> 1. Donat el compromís $C(m)$, B no pot obtenir informació del valor compromès m.
> 2. A ha de poder obrir el compromís $C(m)$ mostrant el valor compromès m.
> 3. A no pot obrir el compromís $C(m)$ mostrant un valor diferent al valor m compromès inicialment.

Amb un esquema de compromís de bit com el que acabem de descriure, el protocol de tirar una moneda a l'aire es pot definir amb els següents passos.

1. L'usuari A tria cara o creu i codifica la seva tria en el missatge m. Posteriorment, calcula el compromís d'm, $C(m)$, i l'envia a B.
2. B genera aleatòriament un bit, on 1 correspondrà al valor cara i 0 correspondrà a creu. B enviarà a A el valor aleatori generat.
3. A obrirà el compromís $C(m)$ mostrant a B quin valor (cara o creu) havia triat, de manera que es veurà qui ha guanyat en el protocol de tirar una moneda a l'aire.

Fixeu-vos que en el pas 2 del protocol, l'usuari A ja ha triat cara o creu però l'usuari B, tot i tenir el compromís $C(m)$, no pot saber quin valor ha triat (gràcies a la primera propietat de l'esquema de compromís de bit). En el pas 2, tot i que l'usuari B no generés el bit de forma aleatòria (per intentar alterar el protocol) el fet que no coneix si A ha triat cara o creu fa que la tria d'aquest valor aleatori sigui intrascendent. D'altra banda, en el pas 3, A ja sap quin valor ha obtingut B i per tant B no pot desdir-se'n. A més, A obre el seu compromís i, tot i conèixer el valor obtingut per B, no pot obrir-lo mostrant un altre valor diferent al que s'ha compromès, gràcies a la tercera propietat de l'esquema de compromís de bit.

Els protocols de compromís de bit es descriuen per mitjà de dues fases: fase de generació del compromís i fase d'obertura del compromís i en els següents apartats veurem dues tècniques diferents que implementen un esquema de compromís de bit.

10.3.1 Compromís de bit utilitzant funcions hash

Una de les tècniques més utilitzades per implementar un esquema de compromís de bit és mitjançant una funció hash, funcions que hem definit en el Capítol 5. Una funció hash h és una funció que parteix d'una informació de mida qualsevol x i en retorna un resum de mida fixa $h(x)$, un valor petit d'alguns centenars de bits. Perquè aquesta funció hash sigui considerada criptogràficament segura cal que compleixi tres propietats essencials. En primer lloc, donat un valor y tal que $h(x) = y$, no és possible trobar la seva antiimatge, x, és a dir, la funció hash no es pot invertir. D'altra banda, donats els valors x i y tals que $h(x) = y$ no és possible trobar un valor $x' \neq x$ tal que $h(x) = h(x') = y$. Finalment, tampoc és possible trobar dos valors x_1 i x_2 tals que $x_1 \neq x_2$ i que $h(x_1) = h(x_2)$. Amb una funció amb aquestes propietats podem definir un compromís de bit de la següent manera.

Sigui m el missatge al qual l'usuari es vol comprometre, en la **fase de generació del compromís** l'usuari A selecciona un valor aleatori r i calcula $C(m) = h(r \parallel m)$ on h és una funció hash criptogràfica.

En la **fase d'obertura del compromís** $C(m)$, l'usuari A revela els valors r i m. A partir d'aquests valors, l'usuari B pot calcular $h(r \parallel m)$ i comprovar que efectivament coincideix amb el valor $C(m)$ al qual A s'havia compromès.

Comprovem que aquest esquema compleix amb les tres propietats d'un esquema de compromís de bit.

1. B no pot obtenir el valor compromès m a partir el compromís $C(m)$ ja que $h(\cdot)$ és una funció hash criptogràfica i per tant no es pot invertir. Fixeu-vos que el valor aleatori r s'utilitza en cas que el missatge m se seleccioni d'un conjunt petit de missatges, per tal d'evitar que B pugui calcular totes les imatges de la funció hash per a tots els possibles valors diferents d'm i descobrir-ne el valor compromès.
2. A pot obrir el compromís $C(m)$ fent públics els valors r i m.
3. A no pot obrir el compromís, $C(m)$, obtenint un valor $m' \neq m$ perquè això voldria dir que A pot trobar $(r \parallel m) \neq (r' \parallel m')$ tal que $h(r \parallel m) = h(r' \parallel m')$ i això no és possible per les propietats que hem enumerat de la funció hash criptogràfica que s'utilitza.

10.3.2 Compromís de Pedersen

Un altre algorisme de compromís de bit és el Compromís de Pedersen, presentat per Torben Pryds Pedersen l'any 1991 com a part d'un esquema de compartició de secrets verificable. Les dues fases d'aquest tipus de compromís de bit es descriuen a continuació.

Sigui m el missatge al qual l'usuari es vol comprometre, en la **fase de generació del compromís** l'usuari A selecciona un grup multiplicatiu $\mathbb{G}$ d'ordre q i tria aleatòriament dos generadors d'aquest grup, g i h, tal que no es conegui el logaritme discret d'h en base g. Aquests valors seran valors públics de l'esquema. Aleshores A calcula el valor del compromís com $C(m) = g^m \cdot h^r \pmod{q}$, on r és un valor aleatori.

En la **fase d'obertura del compromís** $C(m)$, l'usuari A revela els valors r i m. A partir d'aquests valors, l'usuari B pot calcular $g^m \cdot h^r \pmod{q}$ i comprovar que efectivament coincideix amb el valor $C(m)$ al qual A s'havia compromès.

En aquest cas també es pot verificar que aquest esquema compleix amb les tres propietats d'un esquema de compromís de bit.

1. B no pot obtenir cap informació del valor compromès m a partir del compromís $C(m)$ ja que donat el missatge m, com que r és triat de forma uniformement aleatoria en el conjunt $\mathbb{G}$, el resultat del compromís $C(m) = g^m \cdot h^r \pmod{q}$ també és un valor uniformement distribuït en $\mathbb{G}$ i per tant B no en pot obtenir cap informació.
2. A pot obrir el compromís $C(m)$ fent públics els valors r i m.
3. A no pot obrir el compromís, $C(m)$, obtenint un valor $m' \neq m$ perquè això voldria dir que A pot trobar $(r,m) \neq (r',m')$ tal que $g^m \cdot h^r = g^{m'} \cdot h^{r'} \pmod{q}$. Però això no és possible perquè aleshores es podria calcular el logaritme discret d'h en base g com $\frac{m-m'}{r'-r}$ i això no pot succeir perquè el càlcul

del logaritme discret és un problema amb una complexitat massa elevada.

Exemple 10.3 Exemple d'esquema de compromís de bit de Pedersen

Suposem que l'usuari A vol comprometre's al valor $m = 11$. Per fer-ho, treballarà en el grup multiplicatiu $\mathbb{G} = \mathbb{Z}_{103}$ i triarà els valors $g = 6$ i $h = 88$ que són generadors de $\mathbb{Z}_{103}$. Aleshores, per a la generació del compromís, A tria com a valor aleatori $r = 17 \in \mathbb{Z}_{103}$ i calcula el compromís com $C(m) = 6^{11} \cdot 88^{17} \pmod{103} = 34$.

En la fase d'obertura, A envia a B els valors $(m,r) = (11,17)$ i B comprova que efectivament $6^{11} = 53 \pmod{103}$, que $88^{17} = 57 \pmod{103}$ i que per tant el seu producte és efectivament el valor $34 = 53 \cdot 57 \pmod{103}$.

Una de les característiques interessants que presenta el compromís de Pedersen és la seva propietat homomòrfica. Aquesta propietat ens permet generar el compromís de la suma de dos valors sense conèixer els valors i només a partir dels compromisos de cada un d'ells. Així, si tenim dos missatges m_1 i m_2 i els seus respectius compromisos de Pedersen, $C(m_1)$ i $C(m_2)$, tenim que el compromís del valor suma $m = m_1 + m_2$ serà igual al producte dels seus compromisos $C(m) = C(m_1) \cdot C(m_2)$.

En efecte, si escrivim la formulació per a cada un dels compromisos:

$$C(m_1) = g^{m_1} \cdot h^{r_1} \pmod{q} \quad \text{i} \quad C(m_2) = g^{m_2} \cdot h^{r_2} \pmod{q}$$

i en fem el producte, tenim

$$C(m_1) \cdot C(m_2) = (g^{m_1} \cdot h^{r_1}) \cdot (g^{m_2} \cdot h^{r_2}) \pmod{q} = g^{m_1+m_2} \cdot h^{r_1+r_2} \pmod{q} = C(m_1 + m_2)$$

Exercici 10.5 En un esquema de compromís de Pedersen s'utilitzen com a valors de l'esquema $\mathbb{G} = \mathbb{Z}_{113}$ i els generadors $g = 27$ i $h = 94$. L'usuari A s'ha compromès al valor $m_1 = 29$ amb el compromís $C(m_1) = 24$ i a més del missatge, el valor necessari per a obrir el compromís és $r_1 = 90$. Comproveu que efectivament, amb els valors $m_1 = 29$ i $r_1 = 90$ es pot obrir el compromís $C(m_1) = 24$. L'usuari A també s'ha compromès al valor $m_2 = 20$ per mitjà del compromís $C(m_2) = 91$. Calculeu el compromís per al valor $m_1 + m_2$, és a dir $C(m_1 + m_2)$. Podeu obrir el valor d'aquest compromís $C(m_1 + m_2)$?

10.3.3 Aplicacions dels esquemes de compromís de bit

Els esquemes de compromís de bit tenen múltiples aplicacions en protocols criptogràfics on hi ha una desconfiança mútua entre els usuaris que hi participen. Una de les aplicacions és en l'esquema de llançament d'una moneda, que ja hem detallat a l'inici d'aquest apartat.

Una altra aplicació d'aquests esquemes és en l'àmbit dels esquemes de compartició de secrets. Com ja hem comentat en l'apartat corresponent, quan en un esquema de compartició de secrets els usuaris mostren els seus fragments, en cas que alguns usuaris no proporcionin el fragment correcte, la resta d'usuaris no sap si el fragment proporcionat és correcte o no i poden recuperar un secret incorrecte. A més, en els esquemes de compartició de secrets, el gestor que reparteix el secret cal que sigui honest. Els esquemes de compartició de secrets verificables solucionen aquests problemes fent que el gestor que reparteix els secrets també reparteixi un compromís per a cada coeficient del polinomi que fragmenta el secret. D'aquesta manera, tot i no conèixer el polinomi, gràcies a les propietats homomòrfiques del compromís es pot comprovar si un fragment és o no correcte.

Els esquemes de compromís de bit també s'utilitzen en proves de coneixement nul. Com veurem més endavant, les proves de coneixement nul es basen en processos iteratius. Per tal de paral·lelitzar aquests processos sense que en la primera ronda es mostrin tots els valors, es pot utilitzar un esquema de compromís de bit per tal que una de les parts del protocol pugui seleccionar certs valors per endavant però sense

necessitat de revelar-los, de manera que a posteriori, quan s'hagin d'utilitzar en el protocol, es puguin obrir els compromisos per revelar-ne el valor.

10.4 Signatures cegues

Un altre dels protocols interessants en criptografia són les signatures cegues, un protocol que s'utilitza per signar digitalment missatges de forma especial.

> En un **protocol de signatura cega** (en anglès *blind signature*) l'usuari A aconsegueix la signatura d'un missatge m per part de l'usuari B sense que B sàpiga quin missatge ha signat.

El concepte de signatura cega el va proposar David Chaum l'any 1982 per al seu ús en esquemes de pagament anònim.

Per imaginar-se com funciona un protocol de signatura cega és interessant utilitzar una analogia en termes de papers i signatures manuscrites. La idea és que l'usuari A té el document que ha de signar B i en comptes de proporcionar-li directament (fet que faria que B en pogués veure el contingut), A el posa dins d'un sobre. La peculiaritat d'aquest sobre és que està fet de paper carbó, és a dir, si escrivim alguna cosa fora del sobre es calcarà a l'interior. En particular, si B fa una signatura manuscrita fora del sobre, quan posteriorment traiem el document de dins del sobre tindrem el document signat per les propietats de calca del sobre de paper carbó. A més, B no haurà pogut veure el contingut del document que ha signat.

Com amb altres protocols criptogràfics, no és òbvia quina utilitat pot tenir que un usuari pugui signar un document sense saber el que signa. Tot i això, al llarg d'aquest apartat veurem en quines situacions tenen aplicabilitat les signatures cegues.

10.4.1 Signatura cega amb RSA

Donat que un protocol de signatura cega pretén la signatura digital d'un missatge, aquest protocol sempre inclourà un esquema de signatura digital en concret. A continuació veurem un protocol de signatura cega basat en RSA. Aquest mateix protocol és el que va idear D. Chaum quan va proposar el concepte de signatura cega.

Denotarem per m el missatge que A vol tenir signat per B. B signarà digitalment els seus missatges amb un esquema RSA. Per fer-ho utilitzarà la seva clau privada d. La clau pública corresponent a aquesta clau privada la denotarem per (e,n). El protocol es desenvoluparà en els següents passos:

1. A tria un valor aleatori r a $\mathbb{Z}_n$ tal que $mcd(r,n) = 1$ i el xifra amb la clau pública de B, és a dir, calcula $t = r^e \pmod{n}$. El valor t és el valor que utilitzarà per tapar el missatge m que B ha de signar. Per fer-ho, A calcularà $m' = m \cdot t \pmod{n}$ i enviarà el valor m' a B.
2. En rebre m', B simplement realitzarà la signatura sobre aquest valor de forma estàndard, utilitzant la seva clau privada d. Així obtindrà $s' = (m')^d \pmod{n}$ i enviarà el valor s' a A.
3. A destaparà la signatura feta per B simplement dividint la signatura que ha rebut de B, s', pel valor aleatori r generat en el primer pas, $s = \frac{s'}{r}$.

En la Taula 10.4 es mostra el protocol de forma esquemàtica.

Fixeu-vos que el valor s destapat per A en el pas 3 efectivament correspon a la signatura del missatge original m. Això és així perquè:

$$s = \frac{s'}{r} = \frac{(m')^d}{r} = \frac{(m \cdot t)^d}{r} = \frac{m^d \cdot t^d}{r} = \frac{m^d \cdot (r^e)^d}{r} = \frac{m^d \cdot r}{r} = m^d \pmod{n}$$

Taula 10.4: Protocol de signatura cega

Pas	Alice		Bob
1.	Tria $r \in_R \mathbb{Z}_n$ t.q $mcd(r,n) = 1$ Calcula $t = r^e \bmod n$ Tapat : calcula $m' = m \cdot t \bmod n$	$\xrightarrow{m'}$	
2.		$\xleftarrow{s'}$	Signa el valor m' calculant: $s' = (m')^d \bmod n$
3.	Obté la signatura de m calculant $s = \frac{s'}{r}$ (destapat)		

Exemple 10.4 Exemple de protocol de signatura cega amb RSA

Suposem que l'usuari A vol que l'usuari B li signi el missatge $m = 15$. L'usuari B utilitza per a realitzar signatures digitals el criptosistema RSA. La clau pública de B és $(e,n) = (19,551)$ i la corresponent clau privada $d = 451$. Amb aquests paràmetres, el protocol de signatura cega entre A i B serà el següent:

Pas	Alice		Bob
1.	Tria $25 \in_R \mathbb{Z}_{551}$ t.q. $mcd(15,551) = 1$ Calcula $t = 25^{19} = 310 \bmod 551$ Tapat : calcula $m' = 15 \cdot 310 = 242 \bmod 551$	$\xrightarrow{m'=242}$	
2.		$\xleftarrow{s'=14}$	Signa el valor $m' = 242$ calculant: $s' = 242^{451} = 14 \bmod 551$
3.	Obté la signatura de m calculant $s = \frac{14}{25} = 14 \cdot 529 = 243 \pmod{551}$		

Fixeu-vos que el valor $s = 243$ és efectivament la signatura del missatge original $m = 15$ ja que $s = 15^{451} = 243 \bmod 551$

Exercici 10.6 En un sistema d'autentificació anònima, l'usuari A té accés a un recurs S. Per poder-hi accedir, l'autoritat de certificació CA li generarà una credencial que consistirà en la signatura d'un missatge m que contindrà una clau pública generada per l'usuari A i l'identificador del recurs S. Per tal que la credencial sigui anònima, la CA realitzarà una signatura cega de manera que no tindrà manera de saber quina és la clau pública que certifica i per tant quan A accedeixi al recurs la CA no podrà saber-ho. Ara bé, per assegurar-se que A no accedeix a un recurs diferent, la signatura cega la realitzaran amb un protocol de triar i remenar. Així, A prepararà 5 missatges diferents m_i tals que $m_i = (Pk_i||S)$, on PK_i serà una clau pública de la qual A en coneix la corresponent clau privada i el símbol $||$ denota la concatenació. Expliciteu tots els missatges que s'intercanviaran A i la CA en aquest protocol. Suposeu que treballen a $\mathbb{Z}_{899}$ i que els criptosistemes de clau pública que fem servir són l'RSA. Suposeu que el valor $S = 5$ i que el parell de claus (pública i privada) de la CA són $PK_{CA} = 19$,$SK_{CA} = 619$. Per simplificar, no cal indicar les corresponents claus privades de les 5 clau públiques triades.

10.4.2 Aplicacions de les signatures cegues

Hi ha múltiples escenaris on les signatures cegues són interessants d'utilitzar i la majoria d'ells tenen a veure amb la protecció de l'anonimat. Vegem com es poden fer servir en el següent escenari per tenir identificadors

anònims.

Suposeu un sistema amb una autoritat central que té identificats als seus usuaris. Per tal de permetre utilitzar els recursos del sistema de forma anònima, els usuaris poden obtenir uns pseudònims per part de l'autoritat central. Aquests pseudònims estan signats digitalment per l'autoritat central una vegada ha comprovat que l'usuari té suficients privilegis per a utilitzar els corresponents recursos. La signatura de l'autoritat central sobre el pseudònim ha de permetre la seva validació per una tercera part quan l'usuari vol utilitzar el pseudònim davant d'algun dels recursos del sistema on es vol autenticar.

Amb aquest escenari, si l'autoritat central signa els pseudònims dels usuaris de forma estàndard, els usuaris obtindran anonimat davant dels tercers amb qui s'autentifiquin utilitzant el pseudònim. Ara bé, no aconseguiran anonimat davant de l'autoritat central ja que l'autoritat central, quan signa el pseudònim, sap la identitat real de l'usuari i, per tant, la correspondència entre la identitat real i el pseudònim, trencant així l'anonimat.

Una opció per resoldre aquest problema és que l'autoritat central signi el pseudònim però utilitzant un protocol de signatura cega. D'aquesta manera, l'autoritat central donaria validesa al pseudònim però no sabria a qui correspon el pseudònim.

10.4.3 Protecció contra abusos en les signatures cegues

Malgrat que les signatures cegues són interessants d'utilitzar en alguns escenaris, el cert és que la possibilitat que un usuari signi un valor sense saber exactament el que signa pot comportar també alguns problemes de seguretat. Per exemple, com ja hem estudiat anteriorment, la realització d'una signatura digital és equivalent al desxifrat d'un missatge. Per tant, un usuari A que hagués interceptat un missatge xifrat c dirigit a B, podria utilitzar un protocol de signatura cega per tapar c, fer-lo signar per B i d'aquesta manera obtenir el missatge desxifrat. D'altra banda, en escenaris més complexos, el contingut del que signa B pot ser rellevant i A pot voler-lo modificar per treure'n profit. Per exemple, imaginem-nos el cas descrit en l'apartat anterior en el que l'usuari A vol obtenir un pseudònim per autenticar-se. B només li proporcionarà el pseudònim en funció dels privilegis que tingui A en el sistema. A més, el pseudònim ha d'incloure aquesta informació per tal que A el pugui fer servir. Un possible atac d'A seria presentar un pseudònim amb unes atribucions diferents de les que el sistema li permet. Si B ha de realitzar una signatura cega, no podrà verificar aquestes condicions i podria arribar a signar condicions no desitjades.

Per evitar aquest tipus d'accions hi ha diferents estratègiques. La primera és utilitzar una clau específica per a les signatures cegues. És a dir, una clau pública que incorporés la pròpia semàntica de l'autorització. Per exemple, qualsevol pseudònim signat amb la clau pública que tingués el valor concret $PK_{CA}^{S_1,S_2}$ només serviria per autenticar-se davant dels recursos S_1 i S_2. Per a autentica-se davant del recurs S_3, per exemple, caldria tenir el pseudònim signat amb la clau pública $PK_{CA}^{S_3}$. A més, aquestes claus públiques de signatures cegues només es farien servir en aquest context i mai s'utilitzarien per xifrar missatges, de manera que l'atac per al desxifrat no seria possible.

Tot i que aquesta protecció que associa una semàntica a una clau és factible, a la pràctica pot comportar la gestió d'un volum de claus molt gran. Per evitar-ho una altra opció és utilitzar el procediment de "remenar i triar" per assegurar que B no signa res fraudulent. El procés funciona tal i com es mostra en la Figura 10.1.

L'usuari A, en comptes d'enviar un únic valor tapat m' a B, calcula múltiples valors tapats $m'_1, m'_2, \cdots, m'_n$. És important que cada valor s'hagi tapat amb un element diferent, és a dir, per a cada m'_i tindrem un valor t_i diferent, seguint la nomenclatura que hem utilitzat en l'esquema de signatura cega. Cada un d'aquests valors tapats $m'_1, m'_2, \cdots, m'_n$ conté certa informació que B ha de poder validar abans de signar i una altra informació diferent per a cada un dels valors m'_i. Per exemple, en el cas dels pseudònims per a l'autenticació, la part que ha de poder validar B és la part que indica a quins recursos permet accedir el pseudònim. Aquesta part ha de ser la mateixa per a tots els valors. La part que és diferent per a cada valor m'_i és la que indicarà el pseudònim que A farà servir.

Una vegada A ha enviat els n valors tapats $m'_1, m'_2, \cdots, m'_n$ a B, B demana a A que destapi $n-1$ valors, és a dir, A proporcionarà els corresponents t_i per a $n-1$ valors que B haurà triat aleatòriament. Una vegada

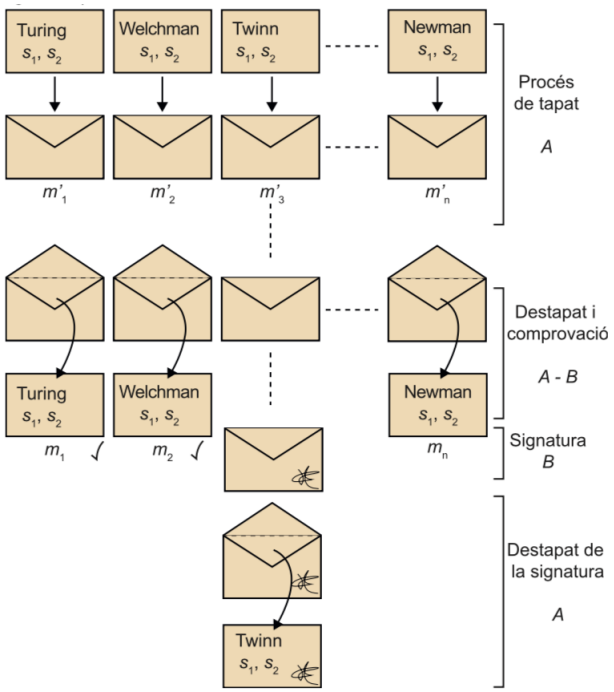


Figura 10.1: Esquema del mecanisme de remenar i triar

destapats, B podrà comprovar que la part que ha de validar coincideix en tots i cada un dels $n-1$ valors que ha destapat. Si és així, assumirà que el valor que li resta per destapar (el qual no pot destapar perquè A manté el corresponent t_i per fer-ho) també compleix les condicions estipulades. Per tant pot procedir a signar de forma cega aquest valor.

Fixeu-vos que cada un dels valors que ha destapat A pot contenir un pseudònim diferent, de manera que B no sap quin pseudònim hi haurà en el valor que ha signat. D'altra banda, la probabilitat que A pugui enganyar a B aconseguint que signi algun contingut que no vulgui es pot fer tant petita com es vulgui ja que el seu valor és de $\frac{1}{n}$.

10.5 Signatures d'anell

Un altre dels protocols relacionats amb les signatures digitals són les signatures d'anell, que van ser formalitzades per Ron Rivest, Adi Shamir i Yael Tauman al 2001.

En un **protocol de signatura d'anell** (en anglès *ring signature*) un usuari u_s que pertany a un grup d'usuaris $\mathscr{R} = \{u_1, \cdots, u_r\}$ (amb $s \in [1, r]$) signa un missatge m, de manera que un validador pot comprovar que la signatura ha estat realitzada per algun membre del grup $\mathscr{R}$ però, alhora, és computacionalment impossible saber quin usuari individual del grup ha realitzat la signatura.

La particularitat dels protocols de signatura en anell és que no necessiten cap mena de coordinador entre els membres del grup, ni tampoc cap procediment d'inicialització dels grups. A partir d'un conjunt d'usuaris cadascun dels quals té un parell de claus pública-privada, qualsevol usuari pot seleccionar un conjunt de possibles signants $\mathscr{R}$ (entre els quals hi és el propi usuari) i crear una signatura, sense necessitar l'ajuda o aprovació dels altres signants, ni la col·laboració de cap tercera part de confiança.

Aquest tipus d'esquemes poden ser útils, per exemple, per a permetre filtrar informació sense revel·lar la identitat de qui ha fet la filtració, però oferint garanties sobre la font. Així, un metge d'un hospital pot voler donar la seva opinió sobre la gestió d'una crisi sanitària a un periodista sense revel·lar la seva identitat (per por a represàlies), però assegurant al periodista que és un metge col·legiat. Així, el metge podria crear una signatura d'anell sobre el missatge on dona la seva opinió, seleccionant com a possibles signants un conjunt de metges col·legiats a la seva elecció. D'aquesta manera, el periodista podria verificar la signatura, comprovant que és vàlida per al grup de signants i que tots ells són metges col·legiats, i per tant podria estar segur que qui ha signat el missatge és efectivament un metge col·legiat. Alhora, la identitat individual del metge que ha signat el missatge quedaria oculta, i el periodista només sabria qui és amb probabilitat $1/r$, amb r el número de possibles signants.

10.5.1 Les signatures d'anell basades en RSA

En aquesta secció presentarem un dels esquemes de signatures d'anell fent servir claus RSA. Els usuaris del sistema disposaran doncs d'un parell de claus pública-privada del criptosistema RSA. Per tal de realitzar una signatura d'anell, un usuari seleccionarà un conjunt de claus públiques (que formaran l'anell, el grup de possibles signants del missatge entre els quals hi serà el propi usuari) i generarà una signatura fent servir les claus públiques dels altres membres de l'anell i la seva clau privada. Aquesta signatura podrà ser després validada per un receptor, coneixent el missatge original i les claus públiques dels usuaris de l'anell.

L'esquema fa servir quatre primitives criptogràfiques bàsiques:

- L'RSA.
- Un criptosistema de clau simètrica.
- Una funció hash.
- Una funció de combinació.

RSA

Farem servir $\mathscr{R} = \{u_1, \cdots, u_r\}$ per descriure al conjunt de possibles signants de l'anell, on $u_s \in \mathscr{R}$ és l'usuari que generarà la signatura. Cada usuari $u_i \in \mathscr{R}$ disposa d'un parell de claus RSA: una de pública $PK_i = (n_i, e_i)$ que és de domini públic, i una de privada $SK_i = (n_i, d_i)$ que només el propi usuari coneix.

Com ja hem vist en el Capítol 6, l'RSA es basa en el fet que la funció $f_i(x) = x^{e_i} \pmod{n_i}$ és computacionalment impossible d'invertir si no es coneix la factorització del mòdul n_i (el que permet calcular l'exponent privat d_i). Per tant, només els usuaris que són coneixedors de la clau privada SK_i, poden calcular $f_i^{-1}(y) = y^{d_i} \pmod{n_i}$.

Criptosistema de clau simètrica i funció hash

El protocol fa servir també un criptosistema de clau simètrica. Denotarem amb $E_k(m)$ la funció de xifrat del missatge m amb la clau k, i amb $E_k^{-1}(y)$ la funció de desxifrat. Els missatges a xifrar i desxifrar seran cadenes de b bits, i la mida de la clau del criptosistema simètric serà l.

Addicionalment, el protocol fa servir una funció hash h, que pot rebre entrades de qualsevol mida i retornarà cadenes d'l bits (que es faran servir com a clau del criptosistema de clau simètrica).

Funció de combinació

Per últim, el protocol fa servir una funció de combinació amb clau, que incorpora totes les primitives anteriors i que és la base del protocol de signatura d'anell. La funció de combinació rep una clau k d'l bits, un valor d'inicialització v de b bits, i un número arbitrari d'entrades y_i també de b bits, i retorna una cadena de b bits:

$$C_{k,v}(y_1, y_2, \cdots, y_r) = E_k(y_r \oplus E_k(y_{r-1} \oplus E_k(y_{r-2} \oplus E_k(\cdots \oplus E_k(y_1 \oplus v)\cdots)))) = z$$

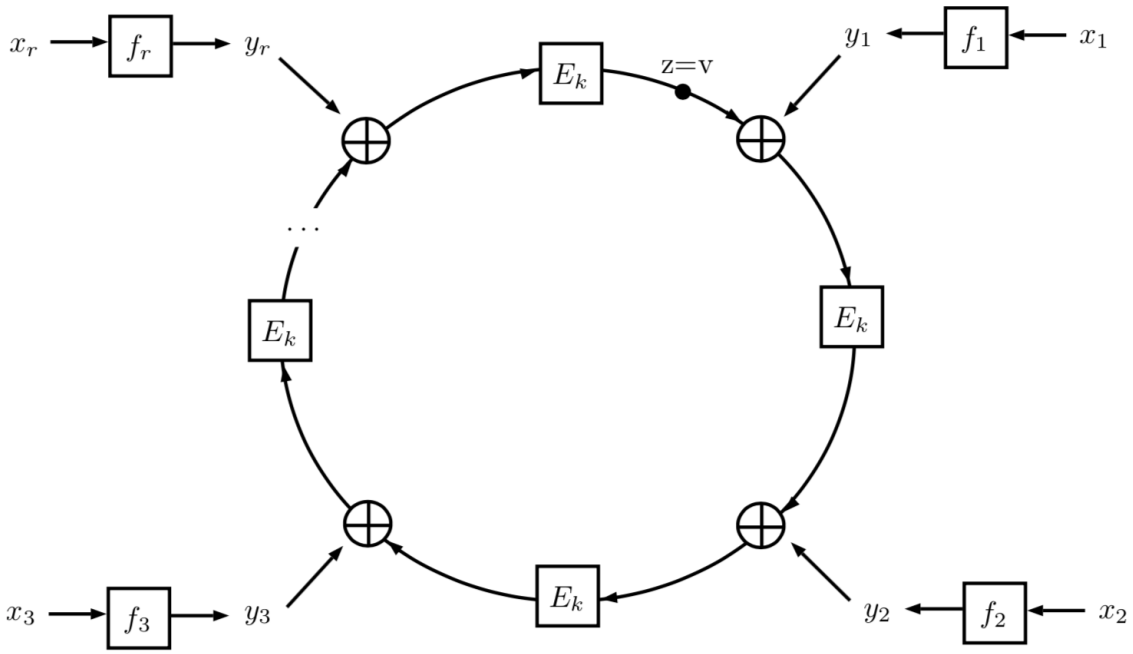


Figura 10.2: Esquema de la signatura d'anell amb RSA

Tot i que a primer cop d'ull la funció de combinació pugui semblar complexa, fixeu-vos que no fa res més que calcular, reiteradament, una xor entre dos valor, xifrant-ne després el resultat amb el criptosistema de clau simètrica.

El punt clau de l'esquema recau en com s'aplica la funció de combinació en la creació de les signatures d'anell. Doncs bé, d'una banda, la clau k que es fa servir en el criptosistema simètric correspon al hash del missatge a signar, és a dir, $k = h(m)$. D'altra banda, la funció s'aplica a la seqüència d'entrada $(y_1, y_2, \cdots, y_r)$, amb $y_i = f_i(x_i)$ per al conjunt de signants de l'anell. Per últim, es força que la sortida z de la funció de combinació hagi de ser igual al valor d'inicialització v, és a dir:

$$C_{k,v}(y_1, y_2, \cdots, y_r) = v$$

Aquest últim punt fa que sigui necessari conèixer com a mínim una de les funcións f_i^{-1} per tal de poder calcular tots els y_i de la seqüència d'entrada, de manera que podrem assegurar que només els membres de l'anell poden generar signatures vàlides per a aquell anell.

Addicionalment, aquest punt també és el que dona el nom de signatures d'anell a la construcció que estem presentant: al forçar que el valor de sortida de la funció de combinació hagi de ser igual al valor d'inicialització, es crea una estructura de dependències circular entre els valors, que té forma d'anell. La Figura 10.2 mostra com es combinen les diferents primitives criptogràfiques en la funció de combinació.

El protocol de signatura d'anell basat en RSA

Una vegada presentades les diferents primitives criptogràfiques que intervenen en el protocol de signatura d'anell basat en RSA, veiem ara com s'executa el protocol.

En primer lloc, descriurem a grans trets en què consisteix la creació i validació d'una signatura amb el protocol de signatura en anell.

En el procés de realització de la signatura, l'usuari que la genera calcularà els y_i corresponents a tots els altres possibles signants fent servir les seves claus públiques i calculant $y_i = f_i(x_i)$ per a un conjunt x_i de valors aleatoris. Tenint en compte la construcció de la funció de combinació, per a un missatge i valor

d'inicialització concrets, això determinarà el valor y_s corresponent al signant. Coneixent la funció f_s^{-1}, el signant podrà invertir el valor y_s i trobar el valor x_s tal que $f_s(x_s) = y_s$. La signatura serà aleshores el conjunt de tots els valors x_i (incloent el del signant) amb les corresponents claus públiques, i el valor d'inicialització. Fixeu-vos que, a diferència d'una signatura convencional on no cal incloure la clau pública del signant, en aquest cas cal incloure totes les claus públiques utilitzades perquè qui posteriorment validi la signatura no té coneixement ni de qui realment ha realitzat la signatura ni de quines claus ha utilitzat en l'anell.

En el procés de validació, el validador podrà comprovar que la signatura és vàlida, ja que podrà recrear tots els y_i a partir dels x_i i les claus públiques, i comprovar després que efectivament l'equació de la funció de combinació es compleix per al valor v rebut. A més, el verificador no podrà saber quin usuari ha signat el missatge, ja que per a tots els usuaris de l'anell de possibles signants, en rep exactament la mateixa informació (el parell x_i, y_i).

En segon lloc, prosseguim a presentar el detall de l'execució del protocol.

El procés de realització de signatura s'inicia quan l'usuari que vol realitzar la signatura del missatge m, selecciona un conjunt d'usuaris, dels quals en coneix la clau pública, per formar part de l'anell de possibles signants $\mathscr{R}$. És a dir, el signant obté r claus públiques $\{PK_i = (n_i, e_i), i \in \{1, \cdots r\}\}$. Amb aquesta informació i els seu propi parell de claus (SK_s, PK_s) (fem servir l'índex s per referir-nos al signant) executa els següents passos:

1. El signant calcula els següent valors:
   - $k = h(m)$
   - b tal que $2^b > n_i$ per a $1 \leq i \leq r$
   - $v \in_R \{0,1\}^b$
   - $x_i \in_R \{0,1\}^b$ per a $1 \leq i \leq r$, per $i \neq s$
   - $y_i = f_i(x_i)$ per a $1 \leq i \leq r$, per $i \neq s$
2. Troba el valor y_s que soluciona l'equació: $C_{k,v}(y_1, y_2, \cdots, y_r) = v$
3. Calcula: $x_s = f_s^{-1}(y_s)$

Per tant, el valor de la signatura del missatge m serà

$$\sigma = \{PK_1, \cdots, PK_r, v, x_1, \cdots, x_r\}$$

Fixeu-vos que al Pas 1, l'usuari signant calcula un conjunt de valors necessaris per a la resolució de l'equació que descriu la funció de combinació. D'una banda, calcula la clau del criptosistema simètric k a partir del hash del missatge a signar. També tria un valor b tal que 2^b sigui major que tots els mòduls de les claus públiques dels usuaris de l'anell. Després, selecciona un valor d'inicialització v aleatori de b bits, així com $r-1$ valors aleatoris x_i (també de b bits). Finalment, per a cada valor x_i, calcula l'y_i corresponent fent servir la clau pública de cadascun dels altres usuaris de l'anell de possibles signants.

A continuació, al Pas 2, el signant resol l'equació plantejada per la funció de combinació, fent servir els valors calculats al pas anterior, per tal de trobar el valor y_s.

Una vegada calculat el valor y_s, al Pas 3 el signant troba el valor x_s tal que $f_s(x_s) = y_s$. Aquesta operació la pot fer ja que el signant coneix el valor de la clau privada, SK_s i, per tant, pot invertir la funció i calcular $f_s^{-1}(y_s) = x_s$. Noteu que aquest procés només el pot fer per al valor y_s, però no per cap altra valor y_i (amb $i \neq s$).

Finalment, al Pas 4 el signant genera la signatura σ, que no és res més que el conjunt de tots els valors x_i (un dels quals haurà calculat a partir de la seva clau privada, i els altres correspondran als valors aleatoris obtinguts al Pas 1), el conjunt de totes les claus públiques dels usuaris de l'anell (entre les quals hi haurà la del signant, que serà indistingible de les altres), i el valor d'inicialització v.

Per tal de validar la signatura σ, el verificador necessita tant el valor de la signatura $\sigma = \{PK_1, \cdots, PK_r, v, x_1, \cdots, x_r\}$ com el missatge m sobre el que s'ha realitzat la signatura. Amb aquestes dades, el verificador realitza els següents passos:

1. Calcula:

- $y_i = f_i(x_i)$ per a $1 \leq i \leq r$
- $k = h(m)$

2. Verifica que es compleixi la igualtat $C_{k,v}(y_1, y_2, \cdots, y_r) = v$

Fixeu-vos que al Pas 1 de la validació, el verificador procedirà a recalcular tots els y_i, aplicant les funcions f_i (que coneix ja que ha rebut també les claus públiques dels possibles signants) a cadascun dels valors x_i rebuts. També calcularà la clau k a partir del hash del missatge rebut.

Per acabar la validació, al Pas 2 el verificador comprovarà que el resultat de la funció de combinació per als valors y_i calculats, la clau k corresponent al missatge i el valor d'inicialització v rebut és també el valor v. Fixeu-vos que, efectivament, el verificador no pot saber quin dels usuaris ha signat el missatge, ja que no pot distinguir de cap manera l'usuari que ha signat de la resta d'usuaris de l'anell de possibles signants.

Exemple 10.5 Exemple de signatura d'anell basada en RSA

Per tal d'executar el protocol de signatura d'anell basada en RSA, caldrà primer triar una funció hash h i una funció de xifrat simètric E. Amb l'objectiu de simplificar al màxim l'exemple i centrar-nos en el càlcul de la signatura d'anell, seleccionem dues funcions senzilles per a aquestes dues primitives, que no oferiran la seguretat desitjada però que ens permetran exemplificar el protocol. Així, d'una banda, farem servir la funció identitat com a funció hash, de manera que $h(x) = x$ per a qualsevol valor d'entrada x. D'altra banda, farem servir una xor entre el missatge i la clau com a funció de xifrat simètric, de manera que $E_k(x) = x \oplus k$.

Suposarem també que el conjunt d'usuaris $\mathscr{U}$ amb claus públiques RSA conegudes que formaran part de l'anell serà $\mathscr{R} = \{u_1, u_2, u_3, u_4\}$ i les claus de cada un:
$PK_1 = (28907, 18541)$
$PK_2 = (41917, 22491)$
$PK_3 = (39407, 26077)$
$PK_4 = (32743, 17539)$
Per a aquest exemple, suposarem que l'usuari que fa la signatura és $s = 3$. La seva corresponent clau privada és $SK_3 = (39407, 27013)$. Suposarem també que el missatge sobre el qual vol realitzar la signatura és $m = 16962$.

El procés de signatura tindrà els següents passos:

1. Calcula:
   - $k = h(16962) = 16962$
   - $b = 16$: ja que $2^{16} = 65536 > n_i$ per a $1 \leq i \leq 4$
   - $v = 29424 \in_R \{0,1\}^{16}$
   - $x = \{25816, 11546, \emptyset, 28447\}$ amb $x_i \in_R \{0,1\}^{16}$
   - $y_1 = f_1(25816) = 25816^{18541} \pmod{28907} = 15266$
   - $y_2 = f_2(11546) = 11546^{22491} \pmod{41917} = 38905$
   - $y_4 = f_4(28447) = 28447^{17539} \pmod{32743} = 11683$
2. Troba el valor y_3 que soluciona l'equació: $C_{16962,29424}(15266, 38905, y_3, 11683) = 29424$ obtenint com a solució $y_3 = 33272$
3. Calcula: $x_3 = f_3^{-1}(33272) = 33272^{27013} \pmod{39407} = 4541$

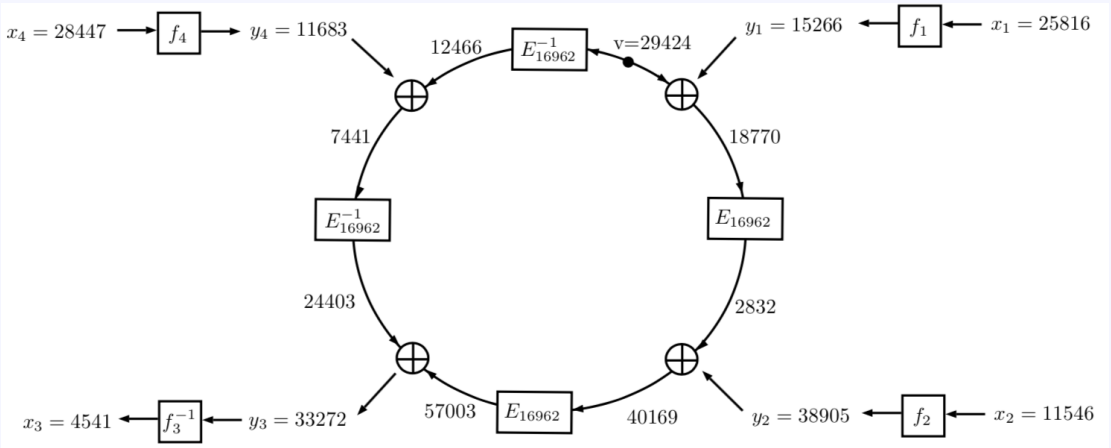
Per tant, el valor de la signatura serà:

$$\begin{aligned}\sigma &= \{PK_1, PK_2, PK_3, PK_4, v, x_1, x_2, x_3, x_4\} = \\ &= \{(28907, 18541), (41917, 22491), (39407, 26077), (32743, 17539), 29424, 25816, \\ &\quad 11546, 4541, 28447\}\end{aligned}$$

Al Pas 1, el signant realitza tots els càlculs per obtenir els valors necessaris per plantejar l'equació de la funció de combinació. Això inclou generar alguns valors aleatòriament (el valor v i també x_1, x_2 i x_4), i calcular els valors y_i corresponents als altres signants (y_1, y_2 i y_4). Noteu com el valor y_3, corresponent

al signant, no s'ha calculat encara en aquest pas, ja que precisament serà la incògnita de l'equació de la funció de combinació, i es calcularà per tal d'assegurar que el resultat de la funció de combinació és exactament el valor d'inicialització.

Al Pas 2 el signant calcula el valor y_3 resolent l'equació donada per la funció de combinació. En la figura següent es mostra, gràficament, el procés que pot seguir el signant per fer aquest càlcul. El signant parteix del valor d'inicialització seleccionat $v = 29424$ i va calculant tots els valors que se'n deriven reseguint l'esquema de la figura per dos camins diferents: d'una banda, en sentit horari i, d'altra banda, en sentit antihorari. Això li permet anar fent tots els càlculs fins a arribar a trobar el valor y_3.



Una vegada sap que $y_3 = 33272$, al Pas 4 el signant calcula el valor x_3 amb la seva clau privada, i al Pas 4 genera la signatura.

El verificador, per verificar la signatura realitzarà els següents passos:

1. Calcula:
   - $y_1 = f_1(25816) = 25816^{18541} \pmod{28907} = 15266$
   - $y_2 = f_2(11546) = 11546^{22491} \pmod{41917} = 38905$
   - $y_3 = f_3(4541) = 4541^{26077} \pmod{39407} = 33272$
   - $y_4 = f_4(28447) = 28447^{17539} \pmod{32743} = 11683$
   - $k = h(16962) = 16962$
2. Verifica:
$$C_{16962,29424}(15266, 38905, 33272, 11683) = 29424$$

Fixeu-vos, que la verificació de la signatura és molt més immediata, i passa per calcular tots els y_i a partir dels x_i rebuts, i comprovar que l'equació de la funció de combinació es compleix per als valors y_i calculats. En aquest cas, efectivament es compleix, i el verificador dona per vàlida la signatura.

Detalls sobre la combinació de claus públiques

Per tal de simplificar la presentació del protocol, la secció anterior ha passat per alt un detall pel que fa als càlculs realitzats en el protocol. En aquesta secció, presentarem doncs una petita modificació en el protocol, que permetrà que operi correctament.

A l'hora de calcular una signatura, s'utilitzen conjuntament diferents claus RSA que pertanyen a diferents usuaris, i que habitualment tindran mòduls també diferents. A més, aquestes claus podrien tenir, fins i tot, mides diferents. Per tal de poder realitzar la signatura considerant les diferències entre els mòduls de les diferents claus, al Pas 2 del protocol es tria un valor b tal que $2^b > n_i$ per a tots els mòduls de les claus de l'anell, i aleshores es treballa sempre amb valors de b bits.

Si apliquem el protocol tal com s'ha descrit a la secció anterior, cada vegada que s'aplica una funció f_i (o f_i^{-1})

es genera una sortida menor a n_i. Per tant, mai s'obté com a resultat un valor entre n_i i 2^b. Aquest interval serà major o menor en funció del mòdul n_i, però en tot cas fa que la funció f_i no generi una permutació entre tots els elements de b bits, ja que hi ha elements vàlids com a entrada que mai es generen com a sortida. Per a evitar-ho, en comptes de fer servir la funció f_i, el protocol de signatura en anell basat en RSA fa servir la funció g_i definida de la següent manera:

$$g_i(m) = \begin{cases} q_i n_i + f_i(r_i) & \text{si } (q_i+1)n_i \leq 2^b \\ m & \text{altrament} \end{cases}$$

on q_i i r_i són enters no negatius tals que $m = q_i n_i + r_i$ i $0 \leq r_i < n_i$.

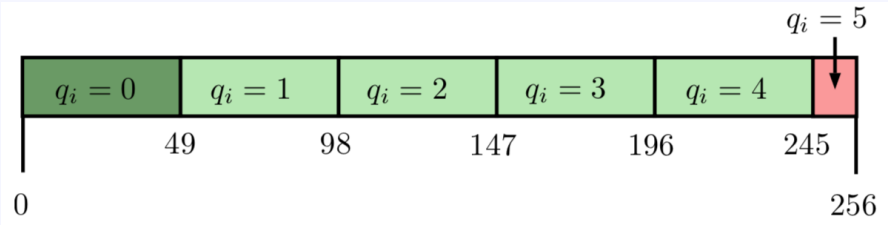
Així, per a valors $m < n_i$, la funció g_i retornarà el mateix que la funció f_i (noteu com aquests casos corresponen a $q_i = 0$). Per a valors d'm superiors al mòdul, intuïtament podem dir que la funció g_i aplicarà la funció f_i al residu (als bits menys significatius) però mantindrà els bits més significatius. Això evitarà reduir la mida de la sortida. Finalment, per als casos excepcionals en els quals la primera expressió podria generar un valor de més de b bits, la funció g_i simplement retorna el valor que rep a l'entrada.

Fixeu-vos que amb l'ús de g_i, s'aconsegueix generar totes les possibles sortides de b bits i, alhora, es manté la propietat de l'RSA que es necessita per al protocol, ja que només l'usuari que sap com invertir f_i podrà invertir també g_i.

Exemple 10.6 Exemple de càlcul de g_i

Suposem un usuari $u_i \in \mathscr{R}$ amb una clau pública RSA $PK_i = (n_i, e_i) = (49, 11)$. A l'hora de fer la signatura d'anell, s'ha triat el valor $b = 8$, que compleix que $2^8 = 256 > 49$.

La imatge següent mostra gràficament els diferents intervals definits per a la funció g_i d'aquest usuari.



Els possibles valors d'entrada m de la funció g_i es troben a l'interval $[0, 256)$ (corresponen als valors que es poden representar amb $b = 8$ bits). La zona colorejada en verd correspon a la primera part de la definició de g_i, en la qual $(q_i+1)n_i \leq 2^b$; la zona colorejada en vermell correspon a la segona part de la definió de g_i, que denota la resta de possibles valors d'entrada.

- El primer interval de la imatge, mostrat de color verd fosc, correspon als valors d'entrada inferiors al mòdul ($m < 49$): el valor q_i és 0, i la funció g_i retorna exactament el mateix que la funció f_i. Aquests són els valors que hem fet servir a l'exemple de la secció anterior, per tal de poder explicar el protocol sense haver de definir g_i. Així, per exemple, per a $m = 34$, $g_i(34) = f_i(34) = 41$.
- La resta d'intervals mostrats en color verd més clar corresponen a valors d'entrada inferiors a 245. En aquests casos, $0 < q_i < 5$, i la funció g_i aplica f_i sobre el residu r_i, però manté la mida de l'entrada. Així, per exemple, per al valor $m = 65$, $q_i = 1$ i $r_i = 16$. Aleshores, $f_i(16) = 4$, i $g_i(65) = 1 \cdot 49 + 4 = 53$. Fixeu-vos que, en aquest cas, la funció g_i retorna un valor superior al mòdul.
- Per últim, l'interval mostrat en vermell a la imatge correspon als valors d'entrada per als quals g_i retorna la mateixa entrada, i que corresponen als valors pels quals l'expressió $q_i n_i + f_i(r_i)$ podria generar una sortida de més de b bits. Així, per exemple, per a $m = 254$, tindríem que $q_i = 5$ i $r_i = 9$.

Aleshores, si calculéssim el resultat de l'expressió anterior:

$$q_i n_i + f_i(r_i) = 5 \cdot 49 + f_i(9) = 5 \cdot 49 + 46 = 291$$

Però $291 \geq 256$, de manera que es produiria un *overflow*. Per a evitar-ho, la funció g_i retorna el valor d'entrada, de manera que $g_i(254) = 254$.
Noteu com aquest aquest interval mostrat en vermell conté valors per als quals l'aplicació de l'expressió $q_i n_i + f_i(r_i)$ genera un valor de més de b bits, però no necessàriament tots els valors de l'interval generen aquest *overflow*. Per exemple, si avaluem l'entrada $m = 249$:

$$q_i n_i + f_i(r_i) = 5 \cdot 49 + f_i(4) = 5 \cdot 49 + 2 = 247$$

veiem que efectivament no sobrepassa el valor màxim de 256. Ara bé, com que l'entrada $m = 249$ és superior a 245, l'expressió que cal aplicar és: $g_i(m) = m$ i per tant $g_i(249) = 249$.

10.6 Proves de coneixement nul

Un dels usos de la criptografia és la gestió de la informació secreta. En ocasions la gestió d'aquesta informació pot comportar que ens interessi convèncer a algú que coneixem certa informació secreta però sense revelar aquesta informació. Dit d'una altra manera, ens interessa un mecanisme per poder demostrar que sabem un secret sense revelar-lo. Aquest concepte, batejat amb el nom de proves de coneixement nul, el van introduir S. Goldwasser, S. Micali i C. Rackoff l'any 1985.

Una **prova de coneixement nul** (en anglès *zero-knowledge poof*) és un protocol entre dos usuaris pel qual l'usuari que actua de provador, P, permet demostrar que coneix un cert valor secret s davant d'un usuari verificador, V, sense proporcionar el valor s. Al final del protocol, V estarà convençut que P coneix el valor s i alhora V no haurà obtingut cap informació sobre aquest valor.

Per tant, una prova de coneixement nul ha de complir les següents propietats:

1. **Correcció:** Si el provador coneix el valor s ha de poder convèncer al verificador que efectivament el coneix.
2. **Robustesa:** La probabilitat que el provador enganyi al verificador ha de ser molt petita. És a dir, si el provador no coneix el valor secret s, la probabilitat que la prova de coneixement nul s'executi correctament és molt petita.
3. **Coneixement nul:** Un cop realitzada la prova de coneixement nul, el verificador no té cap informació sobre el valor secret s que el provador coneix. En particular, el verificador no pot provar a una tercera persona, ni per mitjà d'una prova de coneixement nul, que coneix el secret.

Un exemple gràfic per entendre la mecànica de la majoria de les proves de coneixement nul és el que van proposar J.J. Quisquater i L. Guillou. En aquest exemple tenim una cova, com la que es mostra a la Figura 10.3. La cova té una entrada amb un únic camí. En un punt de la cova, el camí es bifurca i fa una volta fins a tornar-se a unir amb l'altra part del camí. Ara bé, el camí està tancat per una porta que s'obre per mitjà d'una paraula secreta. En Pep (P) coneix aquesta paraula secreta i vol convèncer a en Vicenç (V) que la coneix però no vol donar-li aquesta clau. Per fer-ho executen la següent prova de coneixement nul:

1. En Vicenç és queda a l'entrada de la cova (punt A del gràfic) mentre que en Pep entra dins i tria un dels dos camins fins a arribar a la porta. Per tant, pot estar en el punt C o bé en el punt D depenent de la tria que hagi fet.
2. Un cop en Pep ha arribat davant de la porta, en Vicenç avança fins a la bifurcació (punt B). Des d'allí tria un dels dos camins, el de la dreta o el de l'esquerra i li fa un crit a en Pep perquè surti pel camí

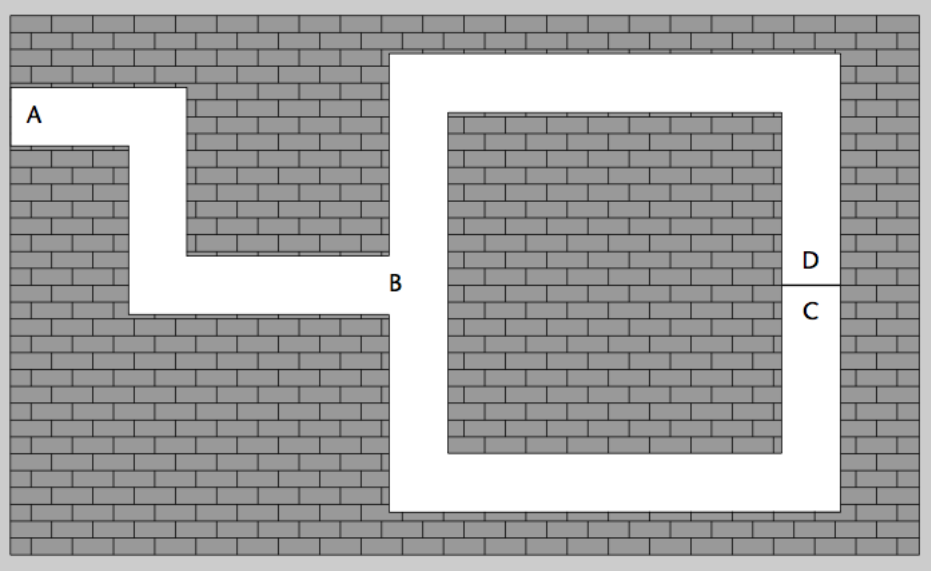


Figura 10.3: Gràfic de la cova de l'exemple

que ha triat.
3. Com que en Pep coneix la clau que obra la porta no tindrà cap problema per sortir pel costat que en Vicenç li ha demanat.

Comprovem ara si amb aquest exemple es compleixen les dues propietats que hem indicat anteriorment.

1. Si en Pep coneix la clau sempre podrà sortir per costat que en Vicenç li demana i per tant podrà demostrar que té el coneixement que vol provar.
   Si tornéssim a fer l'experiment i el repetíssim tantes vegades com volguéssim, en Pep sortiria sempre pel costat que en Vicenç li demanés, ja que coneix la clau que obre la porta i per tant no tindria cap problema.
2. Si en Pep no coneix la clau de la porta no hauria de poder convèncer al Vicenç que sí que la coneix. Fixeu-vos que si no coneix la clau, al fer la prova el Pep tindria una probabilitat d'$1/2$ d'encertar el camí que li demanarà més tard en Vicenç, ja que si en endinsar-se en la cova l'encerta, després podrà sortir pel mateix costat i no li caldrà utilitzar la clau de la porta que de fet no sap. Ara bé, si el procés el repetim un altre cop, en Pep només té $1/4$ de probabilitat d'enganyar-lo. Ja es veu que si repetim la prova n vegades la probabilitat que en Pep enganyi al Vicenç és d'$1/2^n$. Així doncs, si en Vicenç vol estar segur amb probabilitat 0,999023 que en Pep sap la paraula secreta que obre la porta només cal que realitzin la prova 10 vegades.
3. Un cop en Vicenç ha pogut validar que en Pep coneix la clau, en Vicenç no ha obtingut cap informació de la clau i tampoc pot utilitzar informació de la prova que ha fet amb en Pep, malgrat l'hagi repetida 10 vegades, per poder demostrar ell davant d'un tercer que coneix la clau.

L'exemple rebuscat	Com veurem més endavant, aquest exemple il.lustra com funciona una prova de coneixement nul, però òbviament, pel nostre propòsit, n'hi hauria prou en fer entrar en Pep per la dreta i fer-lo sortir per l'esquerra.

En general, les proves de coneixement nul funcionen d'aquesta manera, és a dir, són iteratives de manera que en cada iteració hi ha una probabilitat del 50% d'encertar. A més, en aquests tipus de protocols utilitzen la tècnica anomenada *challenge & response* on el verificador dóna al provador una informació que ell ha generat aleatòriament per tal que el provador la completi utilitzant el secret que coneix. Aquesta tècnica també s'anomena sovint *cut & choose* ja que fa referència al típic protocol de repartir un pastís entre dues persones, en el que una fa les parts (talla) i l'altre tria.

10.6.1 Prova del coneixement del logaritme discret

A continuació veurem un exemple concret d'una prova de coneixement nul aplicada al coneixement del logaritme discret d'un valor, prova que va ser proposada per D. Chaum, J. Evertse i J. Van de Graaf al 1987. Ja hem comentat anteriorment que el càlcul del logaritme discret té una complexitat elevada, és a dir donats uns valors y, g i p és difícil trobar per a quin valor x es compleix que $y = g^x \pmod p$. Per tant, aquesta prova de coneixement nul permet al provador demostrar que coneix el valor x que compleix l'equació $y = g^x \pmod p$ sense necessitat de revelar aquest valor.

El protocol funciona de la següent manera. En primer lloc el protocol estableix tres paràmetres públics (p, g, y), on p és un nombre primer gran, y és un nombre enter tal que $y < p$ i g és un generador del grup multiplicatiu Z_p. El provador ha de demostrar al verificador que coneix el valor x que satisfà l'equació $y = g^x \pmod p$. Per fer-ho el protocol realitza els següents passos:

Pas	Provador (P)		Verificador (V)
1.	Tria $r \in_R \mathbb{Z}_p \setminus \{0,1\}$ Calcula $c = g^r \pmod p$	$\xrightarrow{c}$	
2.		$\xleftarrow{b}$	Tria un bit aleatori $b \in_R \{0,1\}$
3.	Calcula $h = r + b \cdot x \pmod{p-1}$	$\xrightarrow{h}$	
4.			Verifica que $c \cdot y^b = g^h \pmod p$

El protocol consisteix en repetir n vegades els 4 passos descrits anteriorment.

Comprovem com es compleixen les propietats d'una prova de coneixement nul.

1. **Correcció:** en el cas que P conegui el valor x sempre podrà calcular el valor h en el tercer pas del protocol de manera que la validació que farà V en el pas quatre serà correcta.
2. **Robustesa:** per vertificar la propietat de robustesa, analitzarem com s'ho faria P per intentar fer creure a V que coneix x sense realment saber-ho. Per fer-ho, P ha de poder calcular el valor h del pas 3 sense conèixer r. Fixeu-vos que en cas que V li enviï a P el valor $b = 0$ en el pas 2, P calcularà $h = r + b \cdot x \pmod{p-1} = r \pmod{p-1}$ sense necessitat de saber x i aquest valor serà correcte i per tant superarà la validació del pas 4. Ara bé, si V tria $b = 1$ en el pas 2, aleshores P no pot calcular el valor h correcte (li falta el coneixement de x) de manera que no podrà concloure el protocol correctament. Fixeu-vos que la probabilitat que això passi és de $1/2$, ja que és la probabilitat que té V en el pas 2 de triar un 0 ó un 1. Per tant, si repetim el protocol n vegades, la probabilitat que P enganyi a V és d'$\frac{1}{2^n}$.
   Arribats a aquest punt, podríem pensar que no sembla que tingui sentit que V en el pas 2 enviï un 0, ja que en aquest cas, P no necessita conèixer x. Per tant, podríem concloure, erròniament, que en el pas 2, V podria enviar sempre un 1, forçant a P a conèixer x. Ara bé, aquesta estratègia no és correcta. Fixem-nos que si V sempre tria $b = 1$, P pot generar un valor r en el pas 1, però en comptes d'enviar $c = g^r \pmod p$ a V pot enviar $c' = \frac{g^r}{y} \pmod p$. Aleshores, en el pas 3, P envia r en comptes de $r + x$, però la verificació del pas 4 serà correcta perquè $c' \cdot y = \frac{g^r}{y} \cdot y = g^r = g^h$.
   Per tant, fixeu-vos que si P no sap si li arribarà un 0 ó un 1 en el pas 2 (i P no coneix el secret) no sap quina estratègia d'engany ha de seguir en el pas 1, és a dir si ha d'enviar $c = g^r \pmod p$ o bé $c' = \frac{g^r}{y} \pmod p$. Per tant, d'una manera o d'una altra té una probabilitat d'$1/2$ d'enganyar.
3. **Coneixement nul:** el protocol també té la propietat de coneixement nul ja que després d'executar-lo, V només coneix el valor g^r rebut en el pas 1, valor que no té cap relació amb el secret x. A més, el valor h rebut en el pas 3, tan pot correspondre al valor r com al valor $r + x$ i ambdós es presenten com a valors aleatoris per a V i per tant no poden proporcionar cap informació d'r.

Exemple 10.7 Exemple de protocol de prova de coneixement nul del logaritme discret Suposem que els paràmetres del protocol seran $p = 89$ i $g = 3$. El provador P coneix el logaritme discret de $y = 14 \pmod{89}$ que és $x = 9$. Suposarem que V tria el valor $b = 1$ en el pas 2. D'aquesta manera el protocol

tindria els següents valors.

Pas	Provador (P)		Verificador (V)
1.	Tria $r = 20 \in_R \mathbb{Z}_{89} \setminus \{0,1\}$ Calcula $c = 3^{20} = 73 \pmod{89}$	$\xrightarrow{c=73}$	
2.		$\xleftarrow{b=1}$	Tria un bit aleatori $b = 1$
3.	Calcula $h = 20 + 1 \cdot 9 = 29 \pmod{88}$	$\xrightarrow{h=29}$	
4.			Verifica que $c \cdot y^b = 73 \cdot 14^1 = 43 \pmod{89}$ $g^h \pmod{p} = 3^{29} = 43 \pmod{89}$

Exercici 10.7 Voleu realitzar una prova de coneixement nul per demostrar que coneixeu el logaritme discret en base 7 de $y = 94$ a $\mathbb{Z}_{97}$, és a dir el valor x tal que $y = 7^x \pmod{97}$. El problema és que realment no coneixeu el valor x però voleu enganyar a un usuari fent una prova de coneixement nul i que es pugui convèncer que sí que el coneixeu. Afortunadament per vosaltres, el generador pseudoaleatori que fa servir el provador té una vulnerabilitat i vosaltres podeu saber el valor dels bits que genera en el pas 2 del protocol. L'usuari en qüestió vol fer una prova de coneixement nul que li asseguri que coneixeu el valor amb probabilitat superior a 0,75. Desenvolupeu tot el protocol de prova de coneixement nul assumint que el generador aleatori de V produeix els següents bits: 010011100.... Doneu el detall de les operacions i valors que s'intercanvien els usuaris en cada pas del protocol.

10.6.2 Aplicacions de les proves de coneixement nul

Les proves de coneixement nul tenen diferents camps d'aplicació. El primer camp és en els sistemes d'autenticació. El tradicional mètode de contrasenya comença a ser insuficient per a certes aplicacions ja que tant si aquesta, de forma incorrecta, es guarda en clar com si es guarda com a imatge d'una funció hash en algun moment l'usuari ha d'introduir-la en clar i és aleshores quan pot ser interceptada. A més, la utilització de la mateixa informació per a diferents processos d'autenticació pot donar lloc a atacs de repetició en el que un atacant utilitza informació d'una autenticació anterior per autenticar-se posteriorment. Utilitzant proves de coneixement nul, donat que el verificador no pot obtenir cap informació sobre el valor secret que té el provador, la possibilitat d'atacs de repetició desapareix.

Un altre camp on les proves de coneixement nul són importants és en la verificació de paràmetres en protocols criptogràfics més complexos. Per exemple, en protocols de votació electrònica, els votants han de proporcionar certs paràmetres per poder realitzar la votació. Alguns d'aquests paràmetres han de ser secrets, per preservar l'anonimat del vot, però a la vegada han de tenir certes característiques per tal que el protocol funcioni de forma correcta. Les proves de coneixement nul s'utilitzen per provar que un usuari coneix un paràmetre del protocol amb certes característiques sense haver de revelar cap informació del paràmetre en qüestió.

10.7 Protocol de transferència inconscient

Els protocols de transferència inconscient permeten que un usuari emissor "transmeti" informació a un altre usuari receptor de manera que al final de la transmissió, l'usuari receptor només obté una part de la informació "transmesa". A més, la particularitat d'aquests esquemes és que, d'una banda, l'emissor no sap quina informació finalment ha rebut el receptor i, d'altra banda, el receptor no obté cap informació de la informació que no li ha arribat.

0-1 OT	Aquest protocol es basa en la dificultat de calcular arrels quadrades modulars i amb la relació d'aquesta operació i la factorització d'enters.

El concepte de protocol de transferència inconscient va ser presentat per M. O. Rabin l'any 1981. La proposta de Rabin era un protocol en el qual l'emissor té un secret i, amb probabilitat 1/2 l'envia al receptor. Al final del protocol, el receptor pot tenir el secret o no tenir-lo (amb probabilitat 1/2) però l'emissor no pot saber si l'ha rebut o no. Aquest seria el que es coneix com a protocol de transferència inconscient 0-1. En aquest apartat, però, ens centrarem en els protocols de transferència inconscient 1-2.

En un **protocol de transferència inconscient 1-2** (en anglès 1-2 *Oblivious Transfer*) l'usuari A té dos secrets s_0 i s_1. Al final de l'execució del protocol entre A i B, B obté un dels dos secrets amb igual probabilitat. A més, A no pots saber quin secret ha rebut B i B no obtindrà cap informació sobre el secret que no ha rebut.

A continuació veurem un exemple concret d'aquest tipus de protocol.

10.7.1 Protocol d'Even, Goldreich i Lempel

Aquest protocol va ser proposat el 1985 pels criptogràfs Shimon Even, Oded Goldreich, i Abraham Lempel. La proposta utilitza el criptosistema RSA per tal de xifrar els valors secrets que hi intervenen. El protocol permet l'intercanvi inconscient 1-2 dels secrets s_0 i s_1 entre l'usuari A que és qui coneix els dos valors i l'usuari B que és qui en rebrà un dels dos. El funcionament del protocol així com les accions i els missatges que s'intercanvien en el protocol es mostra gràficament en el següent esquema:

Pas	Alice		Bob
1.	Secrets s_0 i s_1. Generació de la clau: $n = p \cdot q$ amb p,q primers $e \cdot d = 1 \pmod{\phi(n)}$ Genera $x_0, x_1 \in_R \mathbb{Z}_n$	$\xrightarrow{(e,n,x_0,x_1)}$	
2.		$\xleftarrow{v}$	Tria un bit aleatori b Genera $k \in_R \mathbb{Z}_n$ Calcula $v = x_b + k^e \pmod{n}$
3.	Calcula $k_0 = (v - x_0)^d \pmod{n}$ Calcula $k_1 = (v - x_1)^d \pmod{n}$ Calcula $s'_0 = s_0 + k_0 \pmod{n}$ Calcula $s'_1 = s_1 + k_1 \pmod{n}$	$\xrightarrow{(s'_0,s'_1)}$	
4.			Coneixent el valor b calcula $s_b = s'_b - k \pmod{n}$

En el Pas 1, A genera el parell de claus pública-privada i dos valors aleatoris. Envia la clau pública i els valors aleatoris a B. En el Pas 2, B triarà un dels dos valors aleatoris i l'amagarà utilitzant una clau. Fixeu-vos que la clau que utilitza per amagar el valor aleatori triat és k^e. Com que k ha estat triat aleatòriament, k^e també és un valor aleatori i el resultat v també aparenta un valor aleatori per a A ja que no coneix ni k ni k^e. En el Pas 3, A calcula dues claus k_0 i k_1 que utilitzarà per amagar els secrets s_0 i s_1 de la transferència inconscient obtenint els valors s'_0 i s'_1. El punt important està en com es calculen aquestes claus k_0 i k_1. Si ens fixem per exemple en k_0, en el cas que B hagi triat el calor x_0 en el Pas 2 tenim que:

$$k_0 = (v - x_0)^d = (x_0 + k^e - x_0)^d = (k^e)^d = k$$

És a dir que A haurà amagat el valor s_0 amb la clau k que B ha triat en el Pas 2. Per tant, B podrà descobrir el valor s_0 en el Pas 3 simplement restant-ne el valor k. En el cas que B hagi triat x_1 en comptes de x_0 en el Pas

2, podrà recuperar el secret s_1 ja que k_1 serà igual a k. Fixeu-vos que en aquest segon cas (B ha triat x_1) B no pot fer res amb el valor s'_0 per intentar esbrinar s_0 ja que no té cap informació de k_0 ja que:

$$k_0 = (v - x_0)^d = (x_1 - k - x_0)^d \neq k$$

Exemple 10.8 Exemple de protocol de transferència inconscient 1-2 Suposem que Alice vol fer una transferència inconscient 1-2 a Bob dels secrets $s_0 = 22$ i $s_1 = 34$.

Pas	Alice		Bob
1.	Secrets $s_0 = 22$ i $s_1 = 34$. Generació de la clau: $n = 19 \cdot 29 = 551$ $e = 19$ i $d = 451$ Genera: $x_0 = 130, x_1 = 525$ aleatoris.	$\xrightarrow[x_0=130,x_1=525)]{(e=19,n=551,}$	
2.		$\xleftarrow{v}$	Tria un bit aleatori $b = 0$ Genera $k = 174$ Calcula: $v = 130 + 174^{19} = 304 \pmod{551}$
3.	Calcula: $k_0 = (304 - 130)^{451} = 174 \pmod{551}$ Calcula: $k_1 = (304 - 525)^{451} = 26 \pmod{551}$ Calcula: $s'_0 = 22 + 174 = 196 \pmod{551}$ Calcula: $s'_1 = 34 + 26 = 60 \pmod{551}$	$\xrightarrow{(s'_0=196,s'_1=60)}$	
4.			Coneixent el valor $b = 0$ calcula $s_0 = 196 - 174 = 22 \pmod{551}$

10.7.2 Aplicacions de la transferència inconscient

Com ja hem dit abans aquest protocol per si sol pot no tenir gaire interès però és la base d'altres esquemes com poden ser la signatura de contractes. Suposem l'escenari en el qual dos usuaris A i B volen signar digitalment un contracte però cap d'ells vol enviar primer la signatura a l'altre per no estar en desavantatge. Vegem com es pot aplicar la transferència inconscient 1-2 per solucionar aquesta situació.

L'usuari A descompon la seva signatura en $2n$ trossos d'm bits cada un, que denotarem per $\{a_i, 1 \leq i \leq 2n\}$. L'usuari B fa el mateix amb la seva signatura i obté els trossos $\{b_i, 1 \leq i \leq 2n\}$. Aleshores:

1. A divideix els seus $2n$ trossos de la seva signatura en n parells, per exemple (a_{2j-1}, a_{2j}) per $j = 1, \cdots, n$ i envia a B un element de cada parell utilitzant una transferència inconscient 1-2, per la qual cosa B rep a_{2j-1} o bé a_{2j}, per $j = 1, \cdots, n$, però A no sap quin dels elements ha rebut B (recordem que cada element del parell té un 50% de probabilitat de ser enviat).
2. Simultàniament al pas 1, B fa exactament el mateix amb els seus $2n$ trossos de la seva signatura: els divideix en parells i envia un element de cada parell a A utilitzant una transferència inconscient 1-2.
3. A i B s'envien l'un a l'altre el primer bit de tots els seus trossos a_i i b_i per $i = 1, \cdots, 2n$, després el segon bit, i així fins al final. Si A vol enganyar B, només té la probabilitat d'$1/2^n$ d'aconseguir-ho ja que B ja té n dels $2n$ nombres secrets del pas 1 i A no sap quins són. Simètricament, es pot aplicar el mateix si B vol enganyar a A.

Fixeu-vos que d'aquesta manera, A i B poden intercanviar la signatura del contracte i cap d'ells no està mai en avantatge de més d'un únic bit.

10.8 Protocols de recuperació privada d'informació

Els protocols de recuperació privada d'informació (coneguts també com a PIR per les seves sigles en anglès, *Private Information Retrival*) són una versió més laxa dels protocols de transferència inconscient presentats al capítol anterior. En ambdós tipus de protocols, un emissor envia dades a un receptor, amb la particularitat que l'emissor no sap quina informació finalment ha rebut el receptor. Els protocols es diferencien doncs en la informació que rep el receptor: mentre que en els protocols de transferència inconscient el receptor no obté cap informació de la dada que no li ha arribat, en els protocols de recuperació privada d'informació el receptor pot rebre informació addicional.

En els protocols de PIR hi intervenen, com a mínim, dues parts: un servidor que emmagatzema una base dades i un usuari que vol fer una consulta sobre la base de dades. L'objectiu del protocol de recuperació privada d'informació és permetre a l'usuari recuperar un ítem de la base de dades sense que el servidor sàpiga quin ítem s'ha recuperat. Com comentàvem al paràgraf anterior, els protocols accepten que l'usuari recuperi més informació de la solicitada: l'objectiu del protocol és protegir la privadesa de la consulta de l'usuari envers del servidor que emmgatzema les dades.

> Un **protocol de recuperació privada d'informació sobre una sola base de dades** (en anglès, *single-database Private Information Retrival protocol*) és un protocol d'intercanvi d'informació entre dos usuaris: una base de dades i un client. La base de dades té un conjunt d'n bits $D = b_1 b_2 \cdots b_n$ i l'usuari vol recuperar el bit en la posició i de la base de dades D sense revel·lar a la base de dades l'índex del bit que s'ha recuperat i.

Així doncs, un exemple de protocol de recuperació privada trivial consistiria en que l'emissor enviés la totalitat de les dades al receptor cada vegada que aquest fes una consulta. D'aquesta manera, efectivament, l'usuari rebria el bit d'interès i, alhora, l'emissor no sabria quin bit volia recuperar el receptor. Òbviament, aquest protocol és molt ineficient (la complexitat de la comunicació és de l'ordre de la mida de la base de dades), i se'n coneixen d'altres que milloren la complexitat d'aquesta versió trivial.

D'altra banda, hi ha protocols de PIR que es basen en la replicació de la base de dades. En aquests protocols, es creen k còpies de la base de dades, que s'emmagatzemen en servidors diferents. Aleshores, s'assumeix que els diferents servidors no poden comunicar-se entre ells (és a dir, no poden col·laborar en la seva tasca d'intentar esbrinar quina consulta ha fet l'usuari). L'usuari extreu informació parcial de cadascuna de les còpies, i la combina per tal de recuperar la informació que volia consultar. Els diferents servidors de bases de dades no aprenen res, individualment, sobre la consulta que ha fet l'usuari.

> Un **protocol de recuperació privada d'informació amb k còpies de la base de dades** (en anglès, *k-database Private Information Retrival protocol*) és un protocol d'intercanvi d'informació entre un usuari i k servidors de base de dades. Cada servidor de base de dades té una còpia completa de la base de dades, un conjunt d'n bits $D = b_1 b_2 \cdots b_n$, i l'usuari vol recuperar el bit en la posició i de la base de dades D interactuant individualment amb cadascun dels servidors i sense revel·lar a cap servidor l'índex del bit que vol recuperar i, assumint que els servidors no poden comunicar-se entre ells i, per tant, no poden compartir la informació que coneixen de la consulta de l'usuari.

A continuació veurem el protocol de Kushilevitz i Ostrovsky, un exemple de protocol de PIR sobre una única base de dades i, després, el protocol de Chor et al., un exemple de protocol de PIR basat en la replicació de

la base de dades.

10.8.1 Protocol de Kushilevitz i Ostrovsky

Notació Direm que $a \in QR(n)$ si $a \in \mathbb{Z}_n$ és un residu quadràtic. Per contra, direm que $a \in QNR(n)$ si $a \in \mathbb{Z}_n$ no és un residu quadràtic.

Aquest protocol va ser proposat el 1997 per Eyal Kushilevitz i Rafail Ostrovsky. La proposta utilitza el criptosistema probabilístic de Golwasser i Micali, per a xifrar la consulta que l'usuari realitza, de manera que la base de dades calcula el resultat sense conèixer els índexs que l'usuari està consultant.

En aquest protocol la base de dades és una matriu de bits, que té s files i t columnes, i que denotarem per $M_{s\times t} = (m_{ij})$. Així, el bit que es troba a la fila i columna j correspon al bit m_{ij}, i la base de dades té $s \times t$ bits emmagatzemats. L'usuari farà una consulta per recuperar un bit específic de la base de dades, que correspondrà al bit que es troba a la fila i' columna j', $m_{i'j'}$.

El funcionament del protocol així com les accions i els missatges que s'intercanvien en el protocol es mostra gràficament a l'esquema següent:

Pas	Usuari		Base de dades
	Índexs del bit a recuperar $i'\,j'$.		Matriu de bits $M_{s\times t} = (m_{ij})$
1.	Generació de la clau: $n = p \cdot q$ amb p,q primers aleatoris $a \in_R QNR(n)$		
2.	Calcula: $r_j \in_R \mathbb{Z}_n$ per tot $1 \leq j \leq t$ $x_j = \begin{cases} ar_j^2 \pmod n, & j = j' \\ r_j^2 \pmod n, & j \neq j' \end{cases}$	$\xrightarrow{(n,\{x_1,\cdots,x_t\})}$	
3.			Calcula: $z_i = \prod_{j=1}^{t} y_{ij}\ \forall\ 1 \leq i \leq s$ amb $y_{ij} = \begin{cases} x_j^2 \pmod n & \text{si } m_{ij} = 0 \\ x_j \pmod n & \text{si } m_{ij} = 1 \end{cases}$
		$\xleftarrow{\{z_1,\cdots,z_s\}}$	
4.	Recupera el bit $m_{i'j'}$ comprovant si z_i' és un QR: Si $z_{i'} \in QR(n)$, llavors $m_{i'j'} = 0$ Si $z_{i'} \in QNR(n)$, llavors $m_{i'j'} = 1$		

En el Pas 1, l'usuari que vol consultar la base de dades genera aleatòriament un parell de claus pública i privada del criptosistema de Goldwasser-Micali. La clau pública correspon als valors (n,a), mentre que la clau privada són els primers que factoritzen n: (p,q). El valor a es tria aleatòriament entre els valors que no són residus quadràtics mòdul n, de manera que no existeix cap x tal que $x^2 = a \pmod n$.

Una vegada l'usuari ha generat el parell de claus, procedeix a calcular els valors que codifiquen la consulta que vol fer a la base de dades. Així, en el Pas 2 l'usuari genera t valors aleatoris r_j, un per cada columna de la matriu que conforma la base de dades, i procedeix a calcular els t valors x_j, cadascun dels quals fa servir el corresponent valor aleatori r_j. Els valors x_j es calculen elevant al quadrat els r_j, a excepció de l'$x_{j'}$ (el valor que correspon a la columna que conté el bit d'interès), que es calcula elevant el valor aleatòri al quadrat i multiplicant-lo per a. D'aquesta manera, els valors x_j són residus quadràtics a excepció de l'$x_{j'}$, que no ho és. L'usuari envia aleshores els t valors x_j i el valor n a la base de dades.

Xifrat amb Goldwasser-Micali

Donada una clau pública (n,a) i un bit en clar m, la funció de xifrat del criptosistema de Goldwasser-Micali és: $E(m) = a^m r^2 \pmod{n}$, amb r un valor aleatori de $\mathbb{Z}_n^*$.

Fixeu-vos, d'una banda, que el Pas 2 és equivalent a xifrar una tira de t bits amb el criptosistema de Goldwasser-Micali, on tots els bits són 0 a excepció del que es troba a la columna j' (que és la columna on hi ha el bit que l'usuari vol recuperar). D'altra banda, noteu com al rebre els valors x_j i el valor del mòdul n, la base de dades no aprén res sobre els índexs de la consulta que l'usuari està realitzant: com que la base de dades no coneix la factorització del mòdul n, no pot distingir els x_j que són residus quadràtics del que no ho és i, per tant, desconeix quina és la columna d'interès per a l'usuari.

Al Pas 3 la base de dades procedeix a calcular el resultat de la consulta. Per fer-ho, calcula un valor z_i per a cada fila de la matriu, que correspon al producte dels x_j que ha rebut de l'usuari, elevant-los al quadrat si el bit m_{ij} de la matriu és un 0. Després, la base de dades envia el resultat de la consulta, és a dir, el conjunt d's valors z_i, a l'usuari.

Finalment, l'usuari recupera el resultat de la consulta a partir de les dades enviades per la base de dades al Pas 4, comprovant si el valor $z_{i'}$ (corresponent a la fila d'interès) és o no un residu quadràtic: si ho és, aleshores el bit recuperat de la base de dades, $m_{i'j'}$, és 0; per contra, si $z_{i'}$ no és un residu quadràtic, aleshores el bit recuperat és 1. En efecte, el valor $z_{i'}$ conté el producte dels x_j elevats al quadrat si corresponen a un bit 0 de la matriu i sense elevar al quadrat si corresponen a un bit 1 de la matriu. Si recordem com s'havien construït els x_j, veurem com aquests són residus quadràtics a excepció de l'$x_{j'}$, que no ho és. Per tant, el valor $z_{i'}$ serà un producte de residus quadràtics si $m_{i'j'}$ és 0 (l'únic valor que no ho era s'ha elevat al quadrat al Pas 3). En canvi, si $m_{i'j'}$ és 1, el producte $z_{i'}$ contindrà un factor que no és un residu quadràtic (precisament el que es troba en la posició j'), fent que el resultat $z_{i'}$ no sigui un residu quadràtic.

Fixeu-vos que el procés que permet recuperar els bits de la base de dades consisteix a calcular si un valor és o no un residu quadràtic mòdul n. Aquest és un problema computacionalment intractable quan n és un valor compost però, en canvi, és molt simple de calcular si el mòdul és un valor primer, o bé si es coneix la factorització d'n. Així doncs, l'usuari que fa la consulta coneix els dos primers p i q tals que $n = pq$, i és aquest coneixement el que li permetrà calcular si el valor $z_{i'}$ és o no un residu quadràtic.

És interessant notar que aquest protocol és un protocol de recuperació privada d'informació, ja que com a resultat de la consulta l'usuari obté més informació de la que ha demanat. Així, mentre que l'usuari únicament estava interessat a recuperar un únic bit, $m_{i'j'}$, l'usuari rep de la base de dades els z_i corresponents a totes les files. Per tant, l'usuari podria seguir el mateix procediment de recuperació per a tots els z_i, i obtindria així tota la columna j' de la base de dades, és a dir, tots els $m_{ij'}$ per a $1 \leq i \leq s$.

Exemple 10.9 Exemple de protocol Kushilevitz i Ostrovsky (recuperació d'un 1) Suposem que un usuari vol consultar una base de dades que està formada per una matriu de bits amb tres files i cinc columnes. L'usuari està interessat a recuperar el valor situat a la segona fila, tercera columna, fent servir el protocol de Kushilevitz i Ostrovsky.

Pas	Usuari		Base de dades
	Índexs del bit a recuperar $2,3$.		Matriu de bits $M_{3\times5}=(m_{ij})$ $M_{3\times5}=\begin{pmatrix}1&0&0&0&1\\0&0&1&0&0\\1&0&0&1&1\end{pmatrix}$
1.	Generació de la clau: $p=59, q=19$ (aleatòriament) $n=59\cdot19=1121$ $325\in_R QNR(n)$		
2.	Calcula: $r=\{25,711,77,47,779\}$ (aleatoris) $x=\{625,1071,1047,1088,380\}$ ja que $x_1=25^2 \pmod{1121}=625$ $x_2=711^2 \pmod{1121}=1071$ $x_3=325\cdot77^2 \pmod{1121}=1047$ $x_4=47^2 \pmod{1121}=1088$ $x_5=779^2 \pmod{1121}=380$	$\xrightarrow{(1121,\{625,1071,1047,1088,380\})}$	
3.			Calcula: $z=\{1083,437,171\}$ ja que $z_1=625\cdot1071^2\cdot1047^2\cdot1088^2\cdot380=$ $=1083 \pmod{1121}$ $z_2=625^2\cdot1071^2\cdot1047\cdot1088^2\cdot380^2=$ $=437 \pmod{1121}$ $z_3=625\cdot1071^2\cdot1047^2\cdot1088\cdot380=$ $=171$
		$\xleftarrow{\{1083,437,171\}}$	
4.	Recupera el bit m_{23} comprovant si z_2 és un QR: $437\in QNR(n)$ Per tant, $m_{23}=1$.		

Exemple 10.10 Exemple de protocol Kushilevitz i Ostrovsky (recuperació d'un 0)
Per tal de veure una execució del protocol on es recuperi un bit que sigui 0 en comptes d'1, podem refer l'exemple anterior suposant que l'usuari volia recuperar el bit en la posició $(1,3)$. En aquest cas, fixeu-vos que els passos 1, 2 i 3 serien exactament els mateixos que a l'exemple anterior, ja que el bit a recuperar es troba també en la columna 3, de manera que l'única diferència estaria en l'últim pas. Ara, al Pas 4, l'usuari recuperaria el bit m_{13} comprovant si $z_1=1083$ és un residu quadràtic: $1083\in QR(n)$ (ja que $209^2=1083 \pmod{1121}$) i, per tant, l'usuari aprendria que $m_{13}=0$.

De manera anàloga, l'usuari podria recuperar també el bit a la posició $(3,3)$, ja que amb la informació rebuda, pot recuperar qualsevol valor de la tercera columna. Al Pas 4, l'usuari recuperaria el bit m_{33} comprovant si $z_3=171$ és un residu quadràtic: $171\in QR(n)$ (ja que $76^2=171 \pmod{1121}$) i, per tant, l'usuari aprendria que $m_{33}=0$.

Els dos exemples anteriors demostren la propietat diferenciadora dels protocols de recuperació privada d'informació envers dels protocols de trasferència inconscient: tot i que l'usuari només volia recuperar el bit a la posició $(2,3)$ de la base de dades (que corresponia a la consulta del primer exemple), l'usuari ha pogut recuperar informació addicional de la resposta de la base de dades. En efecte, l'usuari ha pogut obtenir també el bit en la posició $(1,3)$, sense necessitat de realitzar cap nova consulta a la base de dades, a partir de la informació obtinguda a la primera execució del protocol.

10.8.2 Protocol de Chor et al.

Aquest protocol va ser proposat el 1998 per Chor, Goldreich, Kushilevitz i Sudan, i es basa en replicar la base de dades entre diversos servidors incomunicats entre ells per assegurar que la consulta de l'usuari no es revela a cap dels servidors individuals. L'usuari interactuarà amb cadascun dels servidors, i reconstruirà el resultat de la consulta a partir de la informació parcial que li arriba de cadascuna de les còpies de la base de dades.

En aquest protocol la base de dades és una matriu de bits quadrada, que té s files i s columnes, i que denotarem per $M_{s\times s} = (m_{ij})$. Com al protocol anterior, l'usuari farà una consulta per recuperar un bit específic de la base de dades, que correspondrà al bit que es troba a la fila i' columna j', $m_{i'j'}$.

Per tal de descriure el protocol, definirem la funció de canviar bits (*bit flipping*), $f(x,i)$, com una funció que rep una seqüència x d'n bits i un enter $i \leq n$, i retorna una seqüència també d'n bits que resulta de canviar el bit en la posició i de la seqüència x (deixant la resta de bits iguals).

Exemple de *bit flipping*

Per a la seqüència de bits $x = 01010$, $f(x,1) = 11010$ i $f(x,3) = 01110$.

El funcionament del protocol així com les accions i els missatges que s'intercanvien en el protocol es mostra gràficament a l'esquema següent, per a una execució on la base de dades es troba replicada en quatre servidors diferents, que denotarem per DB^{uv} amb $u,v \in \{0,1\}$ (és a dir, DB^{00}, DB^{01}, DB^{10} i DB^{11}):

Pas	Usuari		Base de dades
	Índexs del bit a recuperar $i'j'$.		Matriu de bits $M_{s\times s} = (m_{ij})$
1.	Calcula: $x^0 = \{x_i \in_R \{0,1\}\}$ per tot $1 \leq i \leq s$ $y^0 = \{y_j \in_R \{0,1\}\}$ per tot $1 \leq j \leq s$ $x^1 = f(x^0, i')$ $y^1 = f(y^0, j')$	$\xrightarrow{x^0,y^0} DB^{00}$ $\xrightarrow{x^0,y^1} DB^{01}$ $\xrightarrow{x^1,y^0} DB^{10}$ $\xrightarrow{x^1,y^1} DB^{11}$	
2.		$\xleftarrow{z^{00}} DB^{00}$ $\xleftarrow{z^{01}} DB^{01}$ $\xleftarrow{z^{10}} DB^{10}$ $\xleftarrow{z^{11}} DB^{11}$	Cada BD^{uv} calcula: $z^{uv} = \oplus_{\substack{\forall i \mid x_i^u = 1 \\ \forall j \mid y_j^v = 1}} m_{ij}$
3.	Recupera el bit $m_{i'j'}$ calculant: $m_{i'j'} = z^{00} \oplus z^{01} \oplus z^{10} \oplus z^{11}$		

En el Pas 1, l'usuari que vol consultar la base de dades genera dues seqüències d's bits aleatòries (x^0 i y^0). Després, genera dues seqüències addicionals (x^1 i y^1), també d's bits, que correspondran a una còpia de les seqüències aleatòries generades en les quals s'ha modificat únicament un bit. Així, la seqüència x^1 serà una còpia d'x^0 amb el bit i' canviat (és a dir, $x^1 = f(x^0, i')$); i la seqüència y^1 serà una còpia d'y^0 amb el bit j' canviat (és a dir, $y^1 = f(y^0, j')$). L'usuari enviarà ara a cada base de dades dues de les seqüències calculades (una de les que codifiquen l'índex i' i una de les que codifiquen l'índex j'). Així, l'usuari enviarà a DB^{00} les seqüències x^0, y^0; a DB^{01} les seqüències x^0, y^1; a DB^{10} les seqüències x^1, y^0; i finalment a DB^{11} les seqüències x^1, y^1. Fixeu-vos que cada base de dades rep un parell de seqüències diferent, i que cada seqüència individual és enviada a dues bases de dades.

Al Pas 2, cada servidor de bases de dades calcularà la seva resposta. Per fer-ho, calcularà una xor de tots els bits de la matriu de la base de dades m_{ij} indicats per les dues seqüències de bits que ha rebut, x^u i y^v. En concret, calcularà la xor de tots els bits m_{ij} per a tots els índex i tals que x_i^u és un 1; i per a tots els índex j tals que y_j^v és un 1. Així, per exemple, la base de dades BD^{01} calcularà el valor z^{01} resultant de la xor entre els valors de la matriu m_{ij} amb els i tals que x_i^0 és 1 i els j tals que y_j^1 és 1. Cadascuna de les bases de dades retornarà el bit z^{uv} calculat a l'usuari.

Finalment, al Pas 3, l'usuari recupera el bit $m_{i'j'}$ calculant una xor dels quatre bits individuals que ha rebut (un de cada base de dades). Noteu com aquesta operació permet recuperar el valor del bit $m_{i'j'}$: degut a la construcció de les seqüències de bits, tots els bits que es tenen en compte a les xors que fa cada base de dades ho fan un número parell de vegades, a excepció del bit seleccionat, que només es té en compte una vegada. Així, quan l'usuari fa una xor de tots els bits z^{uv} rebuts, els que han aparegut un número parell de vegades es cancel·len, i el resultat és per tant el bit consultat $m_{i'j'}$.

Exemple 10.11 **Exemple d'execució del protocol de Chor et al.** Suposem que un usuari vol consultar una base de dades que està formada per una matriu de bits amb quatre files i quatre columnes. L'usuari està interessat a recuperar el valor situat a la segona fila, tercera columna, fent servir el protocol de Chor et al.

Pas	Usuari		Base de dades
	Índexs del bit a recuperar $2,3$.		Matriu de bits $M_{4\times4}=(m_{ij})$ $M_{4\times4}=\begin{pmatrix}1&0&0&1\\0&0&0&1\\1&1&0&0\\0&1&1&0\end{pmatrix}$
1.	Calcula: $x^0=\{0,0,1,0\}$ (aleatòriament) $y^0=\{0,1,0,0\}$ (aleatòriament) $x^1=f(\{0,0,1,0\},2)=\{0,1,1,0\}$ $y^1=f(\{0,1,0,0\},3)=\{0,1,1,0\}$		
		$\xrightarrow{x^0,y^0} DB^{00}$ $\xrightarrow{x^0,y^1} DB^{01}$ $\xrightarrow{x^1,y^0} DB^{10}$ $\xrightarrow{x^1,y^1} DB^{11}$	
2.			BD^{00} calcula: $z^{00}=\oplus_{\forall i\|x_i^0=1,\forall j\|y_j^0=1} m_{ij}=$ $=m_{32}=1$ BD^{01} calcula: $z^{01}=\oplus_{\forall i\|x_i^0=1,\forall j\|y_j^1=1} m_{ij}=$ $=m_{32}\oplus m_{33}=1\oplus0=1$ BD^{10} calcula: $z^{10}=\oplus_{\forall i\|x_i^1=1,\forall j\|y_j^0=1} m_{ij}=$ $=m_{22}\oplus m_{32}=0\oplus1=1$ BD^{11} calcula: $z^{11}=\oplus_{\forall i\|x_i^1=1,\forall j\|y_j^1=1} m_{ij}=$ $=m_{22}\oplus m_{23}\oplus m_{32}\oplus m_{33}=$ $=0\oplus0\oplus1\oplus0=1$
		$\xleftarrow{z^{00}=1} DB^{00}$ $\xleftarrow{z^{01}=1} DB^{01}$ $\xleftarrow{z^{10}=1} DB^{10}$ $\xleftarrow{z^{11}=1} DB^{11}$	
3.	Recupera el bit m_{23} calculant: $m_{23}=1\oplus1\oplus1\oplus1=0$		

Fixeu-vos en els càlculs que realitzen els diferents servidors de base de dades al Pas 2.

- BD^{00} rep $x^0 = \{0,0,1,0\}$ i $y^0 = \{0,1,0,0\}$, de manera que calcula la xor dels bits de la matriu que es troben a la fila $i \in \{3\}$ (ja que el bit a la posició 3 d'x^0 és l'únic bit que és 1) i a la columna $j \in \{2\}$ (ja que el bit a la posició 2 d'y^0 és l'únic bit que és un 1 d'y^0). Per tant, $z^{00} = m_{32}$.
- BD^{01} rep $x^0 = \{0,0,1,0\}$ i $y^1 = \{0,1,1,0\}$, de manera que calcula la xor dels bits de la matriu que es troben a la fila $i \in \{3\}$ (ja que el bit a la posició 3 d'x^0 és l'únic bit que és 1) i a la columna $j \in \{2,3\}$ (ja que els bits a les posicions 2 i 3 d'y^1 són els bits que tenen com a valor 1). Per tant, $z^{01} = m_{32} \oplus m_{33}$.
- BD^{10} rep $x^1 = \{0,1,1,0\}$ i $y^0 = \{0,1,0,0\}$, de manera que calcula la xor dels bits de la matriu que es troben a les files $i \in \{2,3\}$ i a la columna $j \in \{2\}$. Per tant, $z^{01} = m_{22} \oplus m_{32}$.
- BD^{11} rep $x^1 = \{0,1,1,0\}$ i $y^1 = \{0,1,1,0\}$, de manera que calcula la xor dels bits de la matriu que es troben a les files $i \in \{2,3\}$ i a les columnes $j \in \{2,3\}$. Per tant, $z^{11} = m_{22} \oplus m_{23} \oplus m_{32} \oplus m_{33}$.

D'altra banda, noteu perquè l'usuari pot recuperar el valor m_{23} quan fa una xor dels valors rebuts de les quatre bases de dades:

$$\begin{aligned} m_{23} &= z^{00} \oplus z^{01} \oplus z^{10} \oplus z^{11} = \\ &= (m_{32}) \oplus (m_{32} \oplus m_{33}) \oplus (m_{22} \oplus m_{32}) \oplus (m_{22} \oplus m_{23} \oplus m_{32} \oplus m_{33}) = \\ &= m_{22} \oplus m_{22} \oplus m_{23} \oplus m_{32} \oplus m_{32} \oplus m_{32} \oplus m_{32} \oplus m_{33} \oplus m_{33} = \\ &= m_{23} \end{aligned}$$

Efectivament, m_{23} és l'únic bit que no apareix un número parell de vegades, de manera que és l'únic bit que no s'anul·la, fent que el resultat de l'operació resulti en el bit de la base de dades que l'usuari volia recuperar.

Finalment, és interessant destacar perquè és necessari que els servidors de base de dades no col·laborin entre ells per intentar revel·lar la consulta que l'usuari realitza. Per exemple, si els servidors BD^{00} i BD^{11} col·laboressin i compartissin la informació que tenen de la consulta, podrien calcular:

$$\begin{aligned} x^1 &= f(x^0, i'); \{0,1,1,0\} = f(\{0,0,1,0\}, i') \Rightarrow i' = 2 \\ y^1 &= f(y^0, j'); \{0,1,1,0\} = f(\{0,1,0,0\}, j') \Rightarrow j' = 3 \end{aligned}$$

recuperant així els índexs de la consulta de l'usuari, $(2,3)$. En canvi, cada base de dades individual, només coneix un x^u i un y^v, que són seqüències aleatòries de bits que no aporten cap mena d'informació sobre la consulta que fa l'usuari.

10.9 Protocol multipart segur

En algunes aplicacions ens pot interessar que un conjunt d'usuaris realitzi un cert càlcul de forma que, tot i que cada usuari aporta una entrada per a la realització del càcul, al final del procés cada usuari només podrà obtenir el resultat del càlcul però no podrà obtenir els valors d'entrada d'altres usuaris. Aquests tipus de protocols es coneixen com a protocols multipart segurs.

En un **protocol multipart segur** (en anglès *multiparty computation*) un conjunt d'n participants cooperen per a avaluar el valor d'una funció f sobre un conjunt de valors $(v_1, \cdots, v_n)$ aportats pels participants. Com a sortida del protocol, cada usuari u_i obté l'avaluació de la funció $f(v_1, \cdots, v_n)$ però no obté cap informació sobre el contingut dels valors v_j per a $j \in [1,n]$ i $j \neq i$.

Com en la majoria de protocols criptogràfics, una solució simple en aquest escenari és la utilització d'una tercera part de confiança en la qual tothom confia. Aquesta tercera part és la que pot realitzar l'avaluació de

la funció f i, com que tothom hi confia, tothom està segur que un cop cada usuari li ha lliurat el seu tros d'informació, no el mostrarà a cap altra part.

Justament, el que proporcionen els protocols multipart segurs és un mecanisme per poder prescindir de la tercera part de confiança. En els següents apartats veurem dos exemples concrets de protocol multipart segur.

10.9.1 El problema del milionari

Un exemple d'un protocol de càlcul segur a múltiples bandes, en aquest cas a dues bandes, és el protocol proposat per C. Yao l'any 1982, conegut com el problema del milionari. En aquest escenari, dos milionaris A i B volen saber qui és el més ric però no volen revelar el valor de la seva fortuna. És a dir, la funció que volem avaluar de forma segura és una comparació de la mida de dos valors en que cada un dels dos participants aporta un valor.

Per tal de simplificar una mica el problema (de fet, seria equivalent a fitar la fortuna que tenen els participants) transformarem aquest problema en un problema equivalent que consistirà en que l'Alice i en Bob volen saber qui és més gran sense dir quina edat té cada un. Suposarem que tots dos són honestos i que utilitzen les seves edats reals.

Suposarem que l'Alice té x anys, en Bob y i cap dels dos no en té més de 100, és a dir $1 \leq x, y \leq 100$. Per a realitzar aquest protocol utilitzarem un criptosistema de clau pública. Així, tant A com B tindran cada un d'ells un parell de claus pública i privada que seran (E_A, D_A) i (E_B, D_B) respectivament. D'altra banda també assumirem que ambdós usuaris coneixen la clau pública de l'altre participant. A més, A i B també es posen d'acord en la mida màxima que tindran dos dels valors utilitzats en el protocol, t_a i t_b. Així poden assegurar que els valors p_a i p_b triats en el pas 6 són més petits que aquests dos valors.

El protocol funciona tal i com es descriu en l'esquema de la Taula 10.5.

Taula 10.5: Protocol del milionari

Pas	Alice		Bob
1.	Tria $t_a \in_R \mathbb{Z}$		Tria $t_b \in_R \mathbb{Z}$
2.	Calcula $k_a = E_B(t_a)$ $K_a = k_a - x$		Calcula $k_b = E_A(t_b)$ $K_b = k_b - y$
3.		$\xrightarrow{K_a}$	
4.		$\xleftarrow{K_b}$	
5.	Calcula: $f_i = D_A(K_b + i)$ per $1 \leq i \leq 100$		Calcula: $f'_i = D_B(K_a + i)$ per $1 \leq i \leq 100$
6.	Tria $p_a < t_b$ Calcula: $g_i = f_i \pmod{p_a}$ per $1 \leq i \leq 100$ assegurant que $\lvert g_i - g_j \rvert \geq 2$ per a $i \neq j, 1 \leq i, j \leq 100$ Crea la seqüència: $G = \{g_1, \cdots, g_x, g_{x+1} + 1, g_{x+2} + 1, \cdots$ $g_{100} + 1, p_a\}$		Tria $p_b < t_a$ Calcula: $g'_i = f'_i \pmod{p_b}$ per $1 \leq i \leq 100$ assegurant que $\lvert g'_i - g'_j \rvert \geq 2$ per a $i \neq j, 1 \leq i, j \leq 100$ Crea la seqüència: $G' = \{g'_1, \cdots, g'_y, g'_{y+1} + 1, g'_{y+2} + 1, \cdots,$ $g'_{100} + 1, p_b\}$
7.		$\xrightarrow{G}$	
8.		$\xleftarrow{G'}$	
9.	Comprova: Si $G'_x = t_a \pmod{p_b}$, aleshores $y \geq x$ sinó $y < x$		Comprova: Si $G_y = t_b \pmod{p_a}$, aleshores $x \geq y$ sinó $x < y$

Com es pot veure en el protocol, la idea és que tant A com B creen una seqüència de valors, en aquest cas 100 que és el màxim de l'edat dels participants. La particularitat d'aquestes seqüències és que, per exemple,

prenent la seqüència G que genera l'usuari A, per a índex inferiors o iguals a l'índex que determina l'edat de l'usuari A, el valors són congruents amb el valor aleatori que ha triat B, si l'edat d'A és major que la d'B.

Les conclusions sobre qui té més edat que cada usuari obté en el pas 9 són correctes. Per exemple, el raonament respecte a la comprovació de l'usuari B seria la següent: Si A té més edat que B, és a dir $x \geq y$ aleshores el valor de la posició y de la seqüència que A envia a B en el pas 7 és $G_y = g_y$. Per tant, $G_y = f_y \pmod{p_a}$. Com que $f_y = D_A(K_b + y) = D_A(k_b - y + y) = D_A(k_b)$ i $k_b = E_A(t_b)$ ens queda que $f_y = D_A(E_A(t_b)) = t_b$ i per tant $G_y = t_b \pmod{p_a}$.

D'altra banda, si $x < y$ aleshores el valor G_y verifica que:

$$G_y = g_y + 1 \neq g_y = f_y = t_b \pmod{p_a}$$

Exercici 10.8 L'Alice i el Bob han estat de sort i els ha tocat la loteria, que reparteix com a màxim 5 milions. L'Alice ha tingut més sort que en Bob i li han tocat 4 milions, mentre que al Bob n'hi han tocat 2. Com que cap d'ells vol dir quina quantitat li ha tocat, decideixen saber qui és més ric utilitzant el protocol del milionari. Desenvolupeu el protocol per tal que els dos puguin saber qui ha guanyat més diners sense saber quants diners li han tocat a l'altre. Suposarem que utilitzem com a sistema de clau pública l'RSA i el parell de claus pública-privada d'A val $[(e_A = 2573, n_A 5911), (d_A = 197, n_A = 5911)]$, mentre que el parell de B val $[(e = 3109, n_B = 5191), (d_B = 1795, n_B = 5191)]$.

10.9.2 El problema del milionari socialista

En aquest segon protocol, A i B tenen cada un la seva fortuna, representada pels valors x i y respectivament, però en comptes de saber qui és més ric el que volen saber és si la seva fortuna és igual o no. L'execució d'aquest protocol és més elaborada que la de l'exercici anterior ja que el nombre de missatges que s'intercanvien és més elevat, a causa que el protocol utilitza com a subprotocol, en diverses etapes, el protocol d'intercanvi de claus de Diffie i Hellman.

El protocol defineix dos paràmetres generals: un nombre primer p i un valor $h \in \mathbb{Z}_p$ tal que $h \neq 1$. El valor de p ha de ser més gran que la fortuna tant d'Alice com del Bob, és a dir $x < p$ i $y < p$.

El funcionament del protocol es mostra en l'esquema de la Taula 10.6.

Com es pot veure en el protocol, els primers 6 passos corresponen a un intercanvi de claus de Diffie-Hellman que permeten que A i B comparteixin dos valors f i g. Donat que la verificació final podria ser errònia en cas que els valors a_1, a_2, b_1, b_2 fossin 0, per aquest motiu es realitza la validació del pas 5.

Al final del protocol, en cas que la darrera comprovació del valor sigui correcta, tan A com B poden estar convençuts que els dos tenen la mateixa fortuna, ja que:

$$P_a P_b^{-1} = f^r (f^s)^{-1} = f^{r-s} = h^{a_2 b_2 (r-s)} \pmod{p}$$

però d'altra banda,

$$\begin{aligned} c &= ((Q_a Q_b^{-1})^{b_2})^{a_2} \\ &= ((h^x g^x)(h^s g^s)^{-1})^{a_2 b_2} \\ &= (h^{(r-s)} g^{(x-y)})^{a_2 b_2} \\ &= (h^{(r-s)} (h^{a_1 b_1})^{(x-y)})^{a_2 b_2} \\ &= h^{a_2 b_2 (r-s)} h^{a_1 b_1 a_2 b_2 (x-y)} \\ &= P_a P_b^{-1} (h^{a_1 b_1 a_2 b_2 (x-y)}) \pmod{p} \end{aligned}$$

i com que els valors a_1, a_2, b_1, b_2 han estat triats aleatòriament per A i B, l'única possibilitat que $c = P_a P_b^{-1} \pmod{p}$ és en el cas que $x = y$, és a dir, que A i B tinguin la mateixa fortuna.

Taula 10.6: Protocol del milionari socialista

Pas	Alice		Bob
1.	Tria $a_1, a_2 \in_R \mathbb{Z}_p$		Tria $b_1, b_2 \in_R \mathbb{Z}_p$
2.	Calcula $h^{a_1} \pmod p$ $h^{a_2} \pmod p$		Calcula $h^{b_1} \pmod p$ $h^{b_2} \pmod p$
3		$\xrightarrow{(h^{a_1}, h^{a_2})}$	
4.		$\xleftarrow{(h^{b_1}, h^{b_2})}$	
5.	Verifica que: $h^{b_1} \neq 1 \pmod p$ $h^{b_2} \neq 1 \pmod p$		Verifica que: $h^{a_1} \neq 1 \pmod p$ $h^{a_2} \neq 1 \pmod p$
6.	Calcula: $g = (h^{b_1})^{a_1} \pmod p$ $f = (h^{b_2})^{a_2} \pmod p$		Calcula: $g = (h^{a_1})^{b_1} \pmod p$ $f = (h^{a_2})^{b_2} \pmod p$
7.	Tria $r \in_R \mathbb{Z}_p$		Tria $s \in_R \mathbb{Z}_p$
8.	Calcula: $P_a = f^r \pmod p$ $Q_a = h^r g^x \pmod p$		Calcula: $P_b = f^s \pmod p$ $Q_b = h^s g^y \pmod p$
9		$\xrightarrow{(P_a, Q_a)}$	
10.		$\xleftarrow{(P_b, Q_b)}$	
11.	Comprova que: $P_a \neq P_b \pmod p$ $Q_a \neq Q_b \pmod p$		Comprova que: $P_a \neq P_b \pmod p$ $Q_a \neq Q_b \pmod p$
12.	Calcula: $(Q_a Q_b^{-1})^{a_2}$		Calcula: $(Q_a Q_b^{-1})^{b_2}$
13.		$\xrightarrow{(Q_a Q_b^{-1})^{a_2}}$	
14.		$\xleftarrow{(Q_a Q_b^{-1})^{b_2}}$	
15.	Calcula: $c = ((Q_a Q_b^{-1})^{b_2})^{a_2} \pmod p$		Calcula: $c = ((Q_a Q_b^{-1})^{a_2})^{b_2} \pmod p$
16.	Comprova que: $c = P_a P_b^{-1} \pmod p$		Comprova que:: $c = P_a P_b^{-1} \pmod p$

En cas que la comprovació no sigui correcta voldrà dir que les fortunes no són iguals però cap dels dos sabrà qui té una fortuna més gran.

10.10 Resum

En aquest capítol hem estudiant diferents protocols criptogràfics que permeten assolir diferents objectius, tots ells relacionats amb la seguretat de la informació. En primer lloc, hem vist com dos usuaris es poden intercanviar un missatge de forma secreta sense haver intercanviat prèviament cap clau, utilitzant el protocol de tres passos de Shamir. També hem vist com funcionen els esquemes de compartició de secrets, que permeten que un secret es descompongui en diferents fragments de manera que amb la unió d'un nombre fixat de fragments es pot recuperar el secret però amb menys sigui impossible.

D'altra banda, hem estudiat també altres protocols en que la seva aplicació directa pot no ser del tot òbvia. Un exemple són les signatures cegues, on el signatari no coneix el missatge que està signant i aquest fet es pot aprofitar per a protocols d'autenticació anònima. Un altre exemple estudiat són les proves de coneixement nul on un usuari pot demostrar davant d'un altre que coneix un secret sense revelar-ne cap informació del mateix. També hem vist com funciona un protocol de transferència inconscient on la comunicació entre dos usuaris es fa de forma probabilística de manera que l'emissor envia dos missatges i el receptor només en rep un. Ara bé, ni l'emissor sap quin missatge ha rebut el receptor ni el receptor pot triar quin dels dos rebre ja que té una probabilitat del 50 % de rebre'n un dels dos.

Finalment, hem analitzat dos exemples de protocols multipart segurs. En els protocols multipart segurs, n usuaris volen obtenir l'avaluació d'una funció $f(x_1, x_2, \cdots, x_n)$ proporcionant cada un d'ells una entrada de la funció x_i. El punt clau del protocol és que tots els usuaris han d'obtenir el resultat de l'avaluació de la funció però no poden obtenir cap informació sobre les entrades que han proporcionat la resta d'usuaris. Els exemples estudiats han mostrat protocols en els que hi intervenien dos usuaris, un d'ells permet avaluar la funció "menor o igual" i l'altre permet avaluar la funció d'igualtat.

10.11 Solucions dels exercicis

Exercici 10.1:

Amb aquests paràmetres, l'usuari A enviarà de forma secreta el missatge $m = 20$ a B amb el protocol de la Taula 10.7:

Taula 10.7: Exercici 1

Pas	Alice		Bob
1.	$c_1 = 20^{19} \pmod{101} = 30$	$\xrightarrow{30}$	
2.		$\xleftarrow{77}$	$c_2 = (30)^{13} \bmod 101 = 77$
3.	$c_3 = (77)^{79} \bmod 101 = 9$	$\xrightarrow{9}$	
4.			$m = (9)^{77} \bmod 101 = 20$

Exercici 10.2:

El polinomi per generar els fragments estarà compost pel terme independent 11, tindrà com a grau $m-1 = 3-1 = 2$ i com a coeficients podem triar aleatòriament, per exemple, els nombres $x_1 = 8$ i $x_2 = 7$. D'aquesta manera el polinomi ens queda determinat per $a(x) = 7x^2 + 8x + 11 \bmod 13$
Per generar els fragments prenem 5 valors qualssevol menors que p i calculem les seves imatges pel polinomi $a(x)$. Prenent com a valors $\{1,2,3,4,5\}$ tindrem:

$$\begin{aligned}
a(1) &= 7+8+11 = 0 \pmod{13} \\
a(2) &= 28+16+11 = 3 \pmod{13} \\
a(3) &= 63+24+11 = 7 \pmod{13} \\
a(4) &= 112+32+11 = 12 \pmod{13} \\
a(5) &= 175+40+11 = 5 \pmod{13}
\end{aligned}$$

Per tant els fragments dels participants són: $(1,0),(2,3),(3,7),(4,12),(5,5)$.

Exercici 10.3:

Donat que tenim un sistema de compartició llindar amb $m = 3$ podem triar, d'entre els diferents fragments, $(58,137),(11,48),(50,99),(80,50),(104,33),(39,114)$, qualsevol conjunt de 3 punts per recuperar el secret. Per exemple, si triem $(50,99),(80,50),(39,114)$ podem plantejar el següent sistema d'equacions:

$$\begin{aligned}
S + a_1 \cdot 50 + a_2 \cdot 50^2 &= 99 \pmod{149} \\
S + a_1 \cdot 80 + a_2 \cdot 80^2 &= 50 \pmod{149} \\
S + a_1 \cdot 39 + a_2 \cdot 39^2 &= 114 \pmod{149}
\end{aligned}$$

Com que només ens interessa resoldre el sistema per la variable S, que és el secret, podem aplicar el mètode de Kramer i obtenim:

$$\frac{\begin{vmatrix} 99 & 50 & 116 \\ 50 & 80 & 142 \\ 114 & 39 & 31 \end{vmatrix}}{\begin{vmatrix} 1 & 50 & 116 \\ 1 & 80 & 142 \\ 1 & 39 & 31 \end{vmatrix}} = \frac{36}{120} = 36 \cdot 113 = 45 \pmod{149}$$

Exercici 10.4:

El gestor ha utilitzat el polinomi $a(x) = S + a_1 x + a_2 x^2$, on S és la clau del sistema.

Quan els tres usuaris es reuneixen poden escriure el següent sistema:

$$\begin{aligned} f_1 + 2 &= S + a_1 + a_2 \\ f_2 + x &= S + 2a_1 + 4a_2 \\ f_3 + 2 &= S + 3a_1 + 9a_2 \end{aligned}$$

on f_1, f_2, f_3 són els fragments respectius d'A, B i C i x és la trampa que ha fet l'usuari B.

La solució per la incògnita S en aquest sistema és la mateixa que pel sistema en el que cap participant fa trampa, ja que l'enunciat indica que han recuperat el mateix secret, per tant:

$$\begin{aligned} f_1 &= S + a_1 + a_2 \\ f_2 &= S + 2a_1 + 4a_2 \\ f_3 &= S + 3a_1 + 9a_2 \end{aligned}$$

Així doncs podem plantejar la següent igualtat:

$$\frac{\begin{vmatrix} f_1+2 & 1 & 1 \\ f_2+x & 2 & 4 \\ f_3+2 & 3 & 9 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{vmatrix}} = \frac{\begin{vmatrix} j & 1 & 1 \\ m & 2 & 4 \\ s & 3 & 9 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{vmatrix}}$$

i si realitzem les operacions dels determinants ens queda que $6x = 3$ i finalment $x = 3 \cdot 6^{-1} = 7$, sempre treballant a $\mathbb{Z}_{13}$. Per tant, la trampa que ha fet l'usuari B ha estat sumar 7 al seu fragment.

Exercici 10.5:

Per comprovar que efectivament, amb els valors $m_1 = 29$ i $r_1 = 90$ es pot obrir el compromís de Pedersen $C(m_1) = 24$, cal calcular el valor del compromís com $27^{29} \cdot 94^{90} \pmod{113} = 27^{29} \cdot 94^{90} \pmod{113} = 66 \cdot 62 \pmod{113} = 4092 \pmod{113} = 24$ que efectivament coincideix amb $C(m_1)$.

Si el compromís de m_2 és $C(m_2) = 91$ i el compromís de m_1 val $C(m_1) = 24$ podem calcular $C(m_1 + m_2)$ com $C(m_1) \cdot C(m_2)$, és a dir $24 \cdot 91 \pmod{113} = 37$. Tot i poder-lo calcular, aquest compromís de la suma no es pot obrir perquè per fer-ho necessitaríem el valor r_2 que permet obrir $C(m_2)$.

Exercici 10.6:

La solució de l'exercici es mostra en la Taula 10.8.

Exercici 10.7:

Com que A vol fer creure a B que coneix el logaritme discret amb un probabilitat de 0,75 això vol dir que caldrà executar de forma satisfactòria 3 vegades del protocol. Com que A coneix el generador pseudoaleatori sap que en la primera execució del protocol, al Pas 2, V triarà $b = 0$, en la segona execució del protocol triarà $b = 1$ i en la tercera execució triarà $b = 0$. Per tant, per tal d'enredar a V:

- En el primer protocol, en el pas 1 triarem qualsevol valor aleatori, per exemple $r = 45$ que serà el mateix valor que retornarem en el pas 3, $h = 45$. La validació del pas 4 feta per V serà correcta.

Pas	Alice		Bob
1.	Genera 5 claus públiques: $(3,8,10,11,14)$ Prepara els 5 missatges per signar: $m_1 = (3\|\|5) = (35), m_2 = (85)$ $m_3 = (105), m_4 = (115), m_5 = (145)$ Genera els 5 valors per tapar-los: $(5,8,15,23,4)$ $t_1 = 5^{19} = 718 \pmod{899}$ $t_2 = 8^{19} = 872 \pmod{899}$ $t_3 = 15^{19} = 773 \pmod{899}$ $t_4 = 23^{19} = 895 \pmod{899}$ $t_5 = 4^{19} = 473 \pmod{899}$ Tapa els 5 missatges: $m_1 \xrightarrow{t_1} m'_1 = 35 \cdot 718 = 857 \pmod{899}$ $m_2 \xrightarrow{t_1} m'_2 = 85 \cdot 872 = 402 \pmod{899}$ $m_3 \xrightarrow{t_1} m'_3 = 105 \cdot 773 = 255 \pmod{899}$ $m_4 \xrightarrow{t_1} m'_4 = 115 \cdot 895 = 439 \pmod{899}$ $m_5 \xrightarrow{t_1} m'_5 = 145 \cdot 473 = 261 \pmod{899}$	$\xrightarrow{(m'_1,m'_2,m'_3,m'_4,m'_5)}$	
2.		$\xleftarrow{i=2}$	Tria $i = 2 \in_R \mathbb{Z}_5$
3.	Envia els valors t_j menys el t_2 seleccionat.	$\xrightarrow{(t_1,t_3,t_4,t_5)}$	
4.			Destapa els valors i comprova que el servei sigui $S = 5$ (últim dígit) $m'_1 \xrightarrow{t_1} m_1 = 35$ $m'_3 \xrightarrow{t_3} m_3 = 105$ $m'_4 \xrightarrow{t_4} m_4 = 115$ $m'_5 \xrightarrow{t_5} m_5 = 145$ Signa el valor no destapat:
		$\xleftarrow{s'_2=371}$	$s'_2 = 402^{619} = 371 \pmod{899}$
5.	Destapa el valor per obtenir la signatura de m_2 $s'_2 \xrightarrow{t_2} s_2 = \frac{371}{8} = 833$ que veiem que coincideix $85^{619} = 833 \pmod{899}$		

Taula 10.8: Solució de l'Exercici 10.6

- En la segona execució del protocol, en el pas 1 triarem, per exemple, $r = 5$. Però enviarem a V el valor $c = \frac{g^r}{y} \bmod p = \frac{7^5}{94} \bmod 97 = 56$. Aleshores en el pas 3 enviarem $h = r = 5$ i la validació que farà V en el pas 4 també serà correcta ja que $c \cdot y^b = 56 \cdot 94^1 = 26 \bmod 97$ i $g^h = 7^5 = 26 \bmod 97$.
- En la tercera execució, aplicarà la mateixa estratègia que en la primera.

Exercici 10.8:

La solució d'aquest exercici es mostra en la Taula 10.9.

Taula 10.9: Exercici 7

Pas	Alice ($x=4$)		Bob ($y=2$)
1.	Tria $t_a = 1349 \in_R \mathbb{Z}$		Tria $t_b = 1547 \in_R \mathbb{Z}$
2.	Calcula $k_a = E_B(t_a) = 1465$ $K_a = k_a - x = 1461$		Calcula $k_b = E_A(t_b) = 2212$ $K_b = k_b - y = 2210$
3.		$\xrightarrow{K_a=1461}$	
4.		$\xleftarrow{K_b=2210}$	
5.	Calcula: $f_1 = D_A(K_b+1) = 4217$ $f_2 = D_A(K_b+2 = 1547)$ $f_3 = D_A(K_b+3) = 3556$ $f_4 = D_A(K_b+4) = 3569$ $f_5 = D_A(K_b+5) = 884$		Calcula: $f'_1 = D_B(K_a+1) = 1177$ $f'_2 = D_B(K_a+2) = 573$ $f'_3 = D_B(K_a+3) = 4426$ $f'_4 = D_B(K_a+4) = 69$ $f'_5 = D_B(K_a+5) = 674$
6.	Tria $p_a = 239 < t_b$ Calcula: $g_1 = f_1 \pmod{p_a} = 154$ $g_2 = f_2 \pmod{p_a} = 113$ $g_3 = f_3 \pmod{p_a} = 210$ $g_4 = f_4 \pmod{p_a} = 223$ $g_5 = f_5 \pmod{p_a} = 167$ Crea la seqüència: $G = \{154,113,210,223,168,239\}$		Tria $p_b = 739 < t_a$ Calcula: $g'_1 = f'_1 \pmod{p_b} = 438$ $g'_2 = f'_2 \pmod{p_b} = 573$ $g'_3 = f'_3 \pmod{p_b} = 731$ $g'_4 = f'_4 \pmod{p_b} = 69$ $g'_5 = f'_5 \pmod{p_b} = 674$ Crea la seqüència: $G' = \{438,573,732,70,675,739\}$
7.		$\xrightarrow{G}$	
8.		$\xleftarrow{G'}$	
9.	Com que $G'_4 = 70 \neq 1349 = t_a \pmod{739}$, aleshores $y < x$ i per tant A té més diners que B.		Com que: $G_2 = 113 = 1547 = t_b \pmod{239}$, aleshores $x \geq y$ i per tant A té tants o més diners que B.

10.12 Bibliografia

Brassard, G.,Crepeau, C., Chaum, D. *Minimum Disclosure Proofs of knowledge.* Journal of Computer and System Sciences, v. 37, n 2. (1988)

Chaum, D.*Blind signatures for untraceable payments.* Advances in Cryptology Proceedings of Crypto. 82 (3): 199-203. (1983)

Chaum, D., Evertse, J.-H., Van de Graaf, J. *"An improved protocol for demonstrating possession of discrete logarithms and some generalizations"*, Proceedings of Eurocrypt'87. Springer-Verlag. (1988).

Even, S.; Goldreich,O.; Lempel, A. *A Randomized Protocol for Signing Contracts.* Communications of the ACM, Volume 28, Issue 6, pg. 637-647 (1985).

Jakobsson, M.; Yung, M. *Proving without knowing: On oblivious, agnostic and blindfolded provers.* Advances in Cryptology - CRYPTO '96, volume 1109 of Lecture Notes in Computer Science. Berlin. pp. 186-200. (1996)

Shamir, A. *How to Share a Secret.* Communications of the ACM, v. 24, n.11, pp. 612-613 (1979)