TREBALL DE FI DE MÀSTER

# Màster en Matemàtica Aplicada

**Títol: Criptosistema de Lucas**

**Autor: Josep Bellot Casanovas**

**Director: M.Paz Morillo Bosch**

**Departament: MA4**

**Convocatòria**: Octubre 2009



Facultat de Matemàtiques i Estadística

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# Índex

# Capítol 1

# Resum

En aquest treball es descriuen exhaustivament les funcions de Lucas i el mètode XTR (mètodes de criptogràfics sobre extessions de cossos finits), i s'analitzen tots els algorismes que s'han publicat fins ara per computar les funcions *trapdoor* que usen aquests dos mètodes en un ambient criptogràfic. També s'ha fet un recull de totes les aplicacions criptogràfiques que han derivat dels articles [3] i [18], introductors dels dos mètodes anteriors.

# Capítol 2

# Introducció

En la criptografia de clau pública es necessita una funció *trapdoor*, que és una funció fàcilment computable, a diferència de la seva inversa, que només es pot computar en un temps raonable si es coneix una informació concreta (anomenada *trapdoor*). L'exponenciació modular n'és un clar exemple i es pot usar tant en esquemes criptogràfics que basen la seva seguretat en la factorització de nombres enters (sistema RSA) com en esquemes que s'ha de resoldre el logaritme discret per tenir èxit en l'atac (sistema ElGamal).

El 1993 J.Smith et al.[3] van introduir una nova funció *trapdoor* basada en les funcions de Lucas creient que anaven cap a una línia diferent a l'exponenciació modular, eliminant així els defectes d'aquesta. Però el 1999 D.Bleichenbacher et al.[9] es van adonar que les funcions de Lucas seguien sent una exponenciació, però en extensions de cossos finits de segon grau. Les funcions de Lucas també es poden usar en esquemes basats en la factorització i en el logaritme discret, dels quals n'hi ha moltes aplicacions que es recullen en aquest treball. Hi ha aplicacions on se'n pot treure avantatge com en les signatures digitals en subliminals o simplement es poden usar com una bona alternativa.

En el cas dels sistemes que basen la seva seguretat en la del logaritme discret, es treballa amb els elements d'un grup cíclic finit. El grup dels sistemes que usen funcions de Lucas, és un subgrup del grup multiplicatiu $GF(p^2)^*$, els elements del qual són representats amb elements de $GF(p)$. Aquest mètode es pot extrapolar a extensions de grau sis, és a dir, representar elements d'un subgrup de $GF(p^6)^*$ a partir d'elements de $GF(p^2)^*$. Aquest mètode en concret, introduït per A.K.Lenstra [18] i que se l'anomena XTR (*Efficient and Compact Subgroup Trace Representation*), té varis avantatges com la ràpida selecció de paràmetres o que el tamany de les claus es pot reduir bastant comparat amb sistemes com l'RSA. El mètode XTR també té vàries aplicacions interessants, totes basades en el problema del logaritme discret.

En la primera secció d'aquest treball es donen uns conceptes matemàtics previs. En la segona, hi ha un recull de tota la informació de les funcions de Lucas aplicades a la criptografia, on s'hi estudien les propietats, la seguretat i la computació d'aquestes, per finalment descriure tots els esquemes criptogràfics en els que es poden usar les funcions de Lucas. En la tercera secció, es comença descrivint del mètode XTR i l'estudi de la seguretat i dels seus algorismes de computació, i s'acaba amb un recull de les aplicacions on es pot usar aquest mètode. Finalment, en l'última secció es dona una visió general dels mètodes per fer criptografia sobre extensions de cossos finits usant el concepte de tor algebraic.

# Capítol 3

# Conceptes matemàtics

En aquesta secció explicarem la base matemàtica necessària de diverses eines que s'usaran durant aquest treball.

## 3.1 Símbol de Legendre

Sigui $\Delta$ un enter i p un enter primer, definim $\left(\frac{\Delta}{p}\right)$ de la següent manera.

$$\left(\frac{\Delta}{p}\right) = \begin{cases} 0 & \text{si p divideix a } \Delta \\ 1 & \text{si } \Delta \text{ és un residu quadràtic mòdul p} \\ -1 & \text{si } \Delta \text{ no és un residu quadràtic mòdul p} \end{cases}$$

## 3.2 Conjugats, traces i normes

Sigui $q$ un enter primer, $m$ un natural, denotem $GF(q)$ el cos finit de $q$ elements i $GF(q^m)$ a la seva extensió finita de grau $m$.

**Definició 1.** *Per qualsevol $\alpha \in GF(q^m)$, els elements $\alpha$, $\alpha^q$, $\alpha^{q^2}, \ldots, \alpha^{q^{m-1}}$ són els conjugats d'$\alpha$ respecte $GF(q)$.*

**Definició 2.** *Per $\alpha \in GF(q^m)$, la traça $Tr_{GF(q^m)/GF(q)}(\alpha)$ d'$\alpha$ sobre $GF(q)$ està definida com*

$$Tr_{GF(q^m)/GF(q)}(\alpha) = \alpha^q + \alpha^{q^2} + \ldots + \alpha^{q^{m-1}}$$

**Propietat 1.** *Si denotem $K = GF(q)$ i $F = GF(q^m)$, llavors la funció traça $Tr_{F/K}$ satisfà les següents propietats:*

1. *$Tr_{F/K}(\alpha + \beta) = Tr_{F/K}(\alpha) + Tr_{F/K}(\beta)$ per tot $\alpha, \beta \in F$.*
2. *$Tr_{F/K}(c\alpha) = cTr_{F/K}(\alpha)$ per tot $c \in K$ i $\alpha \in F$.*
3. *$Tr_{F/K}$ és una transformació lineal de $F$ a $K$, on $F$ i $K$ són vistos com espais vectorials sobre $K$.*
4. *$Tr_{F/K}(a) = ma$ $\forall a \in K$.*
5. *$Tr_{F/K}(\alpha^q) = Tr_{F/K}(\alpha)$ $\forall \alpha \in F$.*

**Observació 1.** *Per agilitzar la notació, quan parlem de traces sempre trobarem $Tr(\alpha)$ independentment de $F$ i $K$. Quan es tractin les funcions de Lucas tindrem que $F = GF(p^2)$ i $K = GF(p)$, en canvi, en el mètode XTR s'usarà $F = GF(p^6)$ i $K = GF(p^2)$.*

## 3.3 Espai $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$

Sigui $\Delta \in \mathbb{Z} \setminus \{1\}$ lliure de quadrats, definim el cos quadràtic $\mathbb{Q}(\sqrt{\Delta})$ de la següent forma.

$$K = \mathbb{Q}(\sqrt{\Delta}) = \left\{x + y\sqrt{\Delta} \mid x, y \in \mathbb{Q}\right\} =< 1, \sqrt{\Delta} >_{\mathbb{Q}}$$

Llavors tenim dues aplicacions conegudes, la traça i la norma. Denotem com $\overline{\alpha}$ al conjugat de $\alpha$, és a dir, $\overline{\alpha} = x - y\sqrt{\Delta}$ si $\alpha = x + y\sqrt{\Delta}$.

$$\begin{array}{rccc} Tr: & K & \longrightarrow & \mathbb{Q} \\ & \alpha & \longmapsto & \alpha + \overline{\alpha} \end{array}$$

$$\begin{array}{rccc} N: & K & \longrightarrow & \mathbb{Q} \\ & \alpha & \longmapsto & \alpha \cdot \overline{\alpha} \end{array}$$

**Observació 2.** *La traça és $\mathbb{Q}$-linial i la norma és un homomorfisme entre $K^*$ i $\mathbb{Q}^*$.*

**Observació 3.** *Donat $\alpha = x + y\sqrt{\Delta}$ (amb $x, y \in \mathbb{Q}$), tenim:*

$$\begin{array}{rcl} Tr(\alpha) & = & 2x \\ N(\alpha) & = & x^2 - \Delta y^2. \end{array}$$

En aquest context definim l'anell d'enters de K com

$$\mathcal{O}_\Delta \;:=\; \{\alpha \in K \mid Tr(\alpha), N(\alpha) \in \mathbb{Z}\}$$

i com podem observar són els elements del cos quadràtic tals que la seva norma i la seva traça són enters. Per tenir una idea de com són els elements de l'anell d'enters, tenim una proposició que ens separa dos casos en funció del valor de $\Delta$.

**Proposició 1.**

$$\mathcal{O}_\Delta := \begin{cases} \mathbb{Z}[\sqrt{\Delta}] & \textit{si } \Delta \equiv \textit{ 2 ò 3} \pmod 4 \\ \mathbb{Z}[\frac{1+\sqrt{\Delta}}{2}] & \textit{si } \Delta \equiv 1 \pmod 4 \end{cases}$$

*En particular, $\mathcal{O}_\Delta$ és un $\mathbb{Z}$-mòdul lliure de rang 2.*

**Dem.** Veure [43].

Sigui $a \in \mathbb{Z}$, ara ens interessa veure com és l'espai $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$ i per això primer prenem l'ideal $a\mathcal{O}_\Delta$, que és l'ideal principal $(a) = \{a\alpha$, on $\alpha \in \mathcal{O}_\Delta\}$ i llavors fem el quocient d'anells $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$.

**Proposició 2.**

$$\mathcal{O}_\Delta/a\mathcal{O}_\Delta := \left\{x + y\sqrt{\Delta} \mid x, y \in \mathbb{Z}/a\mathbb{Z}\right\}$$

**Dem.** Per la proposició (1), sabem que $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$ pot ser de la forma $\mathbb{Z}[\sqrt{\Delta}]$ o $\mathbb{Z}[\frac{1+\sqrt{\Delta}}{2}]$, per tant veurem què li passa a cada cas si li fem mòdul $a$.

- En el cas $\mathbb{Z}[\sqrt{\Delta}]/a\mathbb{Z}[\sqrt{\Delta}]$, tenim elements de la forma $\left\{x + y\sqrt{\Delta} \mid x, y \in \mathbb{Z}\right\}$ amb la relació d'equivalència $\sim$ tal que

  $$x_1 + y_1\sqrt{\Delta} \sim x_2 + y_2\sqrt{\Delta} \Leftrightarrow (x_1 - x_2) + (y_1 - y_2)\sqrt{\Delta} \in a\mathbb{Z}[\sqrt{\Delta}]$$

  D'aquí obtenim que $x_1 \equiv x_2 \pmod a$ i que $y_1 \equiv y_2 \pmod a$.

- En l'altre cas, $\mathbb{Z}[\frac{1+\sqrt{\Delta}}{2}]/a\mathbb{Z}[\frac{1+\sqrt{\Delta}}{2}]$, els elements són de la forma $\left\{x + y\frac{1+\sqrt{\Delta}}{2} \mid x, y \in \mathbb{Z}\right\}$ amb la relació d'equivalència $\sim$ tal que

$$x_1 + y_1\frac{1+\sqrt{\Delta}}{2} \sim x_2 + y_2\frac{1+\sqrt{\Delta}}{2} \Leftrightarrow (x_1 - x_2) + (y_1 - y_2)\frac{1+\sqrt{\Delta}}{2} \in a\mathbb{Z}[\frac{1+\sqrt{\Delta}}{2}]$$

  Si suposem que l'element de $a\mathbb{Z}[\frac{1+\sqrt{\Delta}}{2}]$ és $a\gamma + a\delta\frac{1+\sqrt{\Delta}}{2}$ i reagrupem els coeficients de l'arrel, obtenim $\frac{y_1-y_2}{2} = \frac{a\delta}{2}$, que és equivalent a dir que $y_1 \equiv y_2 \pmod a$. Reagrupant els altres coefiecients ens queda $(x_1 - x_2) + \frac{y_1-y_2}{2} = a\gamma + \frac{a\delta}{2}$, d'on podem afirmar que $x_1 \equiv x_2 \pmod a$.

□

Sigui $\Pi$ l'aplicació canònica $\mathcal{O}_\Delta \to \mathcal{O}_\Delta/a\mathcal{O}_\Delta$, definim la norma i la traça en $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$ de la següent manera.

$$\begin{aligned} \forall\alpha \in \mathcal{O}_\Delta,\, N_{\mathcal{O}_\Delta/a\mathcal{O}_\Delta}(\Pi(\alpha)) &\equiv N_{\mathcal{O}_\Delta}(\alpha) \pmod a \\ \forall\alpha \in \mathcal{O}_\Delta,\, Tr_{\mathcal{O}_\Delta/a\mathcal{O}_\Delta}(\Pi(\alpha)) &\equiv Tr_{\mathcal{O}_\Delta}(\alpha) \pmod a \end{aligned}$$

## 3.4 Teorema xinès dels residus

Sigui n=pq i $y \in \mathbb{Z}_n$, considerem els nombres

$$\begin{aligned} a_1 &:= y \pmod p \in \mathbb{Z}_p \\ a_2 &:= y \pmod q \in \mathbb{Z}_q. \end{aligned}$$

El teorema xinès dels residus considera la qüestió de recombinar $a_1$ i $a_2$ per tornar a aconseguir $y$. Concretament ens diu que hi ha una única forma de recuperar $y$ sota unes certes condicions, i sota aquestes, també ens diu com.

**Exemple 1.** $n = 6 = 3 \cdot 2$

$$\begin{aligned} 0 &\longrightarrow (0,0) \\ 1 &\longrightarrow (1,1) \\ 2 &\longrightarrow (2,0) \\ 3 &\longrightarrow (0,1) \\ 4 &\longrightarrow (1,0) \\ 5 &\longrightarrow (2,1) \end{aligned}$$

**Exemple 2.** $n = 4 = 2 \cdot 2$

$$\begin{aligned} 0 &\longrightarrow (0,0) \\ 1 &\longrightarrow (1,1) \\ 2 &\longrightarrow (0,0) \\ 3 &\longrightarrow (1,1) \end{aligned}$$

La diferència entre els dos exemples és que en el primer la correspondència entre $\mathbb{Z}_n$ i $\mathbb{Z}_p \times \mathbb{Z}_q$ és única, en canvi, en el segon no. Això és degut a que p i q han de ser primers entre sí. A continuació una simplificació del teorema xinès on no queda detallat com trobar explícitament la $y$.

**Teorema 1** (*Teorema xinès dels residus*)**.** *Siguin $n_1, n_2, \ldots, n_k$ siguin enters relativament primers dos a dos. Sigui $a_i \in \mathbb{Z}_{n_i}$ per $1 \leq i \leq k$ i $n = n_1 n_2 \ldots n_k$. Llavors existeix una única $y \in \mathbb{Z}_n$ tal que $y \equiv a_i \mod n_i$ per $i = 1 \ldots k$. A més a més hi ha un algorisme de complexitat $O(k^2)$ (on $k = max(|\, n_1, |\, n_2)$) que donats $a_1, a_2, n_1$ i $n_2$, calcula $y$.*

# Capítol 4

# Funcions de Lucas

Les funcions de Lucas, definides a partir d'unes seqüències recursives de segon ordre, són una alternativa actual a les funcions *trapdoor*. Com ja veurem, tenen diverses característiques que les converteixen en bones candidates per a dissenyar nous sistemes criptogràfics. Per això se n'estudiarà la seguretat a partir d'uns nous problemes que definirem i la computació a partir dels algorismes que la fan eficient.

## 4.1 Seqüències de Lucas

Siguin P i Q $\in \mathbb{Z}$ tal que $\Delta = P^2 - 4Q$ no és un quadrat, definim les seqüències de Lucas de la següent manera,

$$U_{k+1}(P,Q) = PU_k(P,Q) - QU_{k-1}(P,Q) \quad \text{amb} \quad U_1(P,Q) = 1 \text{ i } U_0(P,Q) = 0, \tag{4.1}$$
$$V_{k+1}(P,Q) = PV_k(P,Q) - QV_{k-1}(P,Q) \quad \text{amb} \quad V_1(P,Q) = P \text{ i } V_0(P,Q) = 2. \tag{4.2}$$

Si considerem l'equació $x^2 - Px + Q = 0$ i a les seves arrels les denotem $\alpha$ i $\beta$ és senzill (per exemple, usant inducció) verificar que la forma general de cada seqüència és

$$U_k(P,Q) = \frac{\alpha^k - \beta^k}{\alpha - \beta} \qquad \text{i} \qquad V_k(P,Q) = \alpha^k + \beta^k. \tag{4.3}$$

Aquestes dues funcions que depenen de les variables k, P i Q, seran les funcions de Lucas que usarem.

**Observació 4.** *$U_k(P,Q)$ i $V_k(P,Q)$ són seqüències d'enters degut a com estan construïdes en (4.1) i (4.2).*

**Observació 5.** *A partir d'aquí ens referirem bàsicament a $V_k(P,Q)$ ja que és la funció que usarem més endavant per construir el sistema criptogràfic.*

**Observació 6.** *Sigui $n \in \mathbb{N}$, no tindrem problemes al usar mòduls ja que*

$$V_k(P \bmod n, Q \bmod n) = V_k(P,Q) \mod n.$$

**Observació 7.** *En diversos casos, denotarem $V_k$ i $U_k$ enlloc de $V_k(P,Q)$ i $U_k(P,Q)$ quan $P$ i $Q$ es puguin sobreentendre pel context. Si trobem $V_k(P)$ i $U_k(P)$, voldrà dir que s'ha pres $Q = 1$.*

D'altra banda, també podem expressar aquestes seqüències com a coeficients de les potències d'$\alpha$ tal com se'n van adonar a [9]. Veiem-ho: de les definicions de (4.3), si sumem $V_k$ i $(\alpha-\beta)U_k$ ens queda $2\alpha^k = V_k + (\alpha-\beta)U_k$, on $\alpha-\beta=\sqrt{\Delta}$, i per tant $\forall k\in\mathbb{Z}$,

$$\alpha^k = \frac{V_k + U_k\sqrt{\Delta}}{2}. \tag{4.4}$$

Llavors, per l'observació 4 i la relació (4.4) podem afirmar que $\alpha$ i totes les seves potències pertanyen a $\mathcal{O}_\Delta$. Recordem que els elements de l'anell d'enters d'un cos K són aquells elements del cos tals que la seva traça i la seva norma són enters.

## 4.2 Espai $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$

Fins ara hem vist com es construeixen aquestes seqüències de Lucas, però perquè siguin útils en un ambient criptogràfic aquestes han d'estar en un context adient. Per tant, com és habitual usarem mòduls, i per això prendrem un enter $a$ per construir l'espai $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$ com en l'apartat 3.3.

Si calculem la norma de $\alpha^1$ de (4.4) obtenim

$$N(\alpha) = N(\frac{P+\sqrt{\Delta}}{2}) = \frac{P^2}{4} - (P^2-4Q)\frac{1}{4} = Q.$$

Per tant, si prenem $Q=1$, la norma d'$\alpha$ és 1 i com que per l'observació (2) la norma és un homomorfisme del grup $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^*$ en $(\mathbb{Z}/a\mathbb{Z})^*$, llavors els elements de $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^*$ de norma 1 formen un grup multiplicatiu que denotarem com a $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$. Aquest grup, doncs, serà el que usarem per construir els diferents esquemes criptogràfics. Per això ens interessa saber, per exemple, el seu ordre i com hi operem.

**Proposició 3.** *1. Sigui $p$ un primer senar tal que $p \nmid \Delta$, per a qualsevol $r \geq 1$ l'ordre del grup $(\mathcal{O}_\Delta/p^r\mathcal{O}_\Delta)^\wedge$ és $\varphi_\Delta(p^r) = p^{r-1}(p-(\frac{\Delta}{p}))$.*

*2. Sigui $a$ un enter senar de la forma $a = p_1^{r_1}\cdots p_l^{r_l}$ on $p_1, p_2, \ldots, p_l$ són primers diferents. Si $gcd(\Delta, a) = 1$, tenim el següent isomorfisme:*

$$(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge \approx (\mathcal{O}_\Delta/p_1^{r_1}\mathcal{O}_\Delta)^\wedge \times \cdots \times (\mathcal{O}_\Delta/p_l^{r_l}\mathcal{O}_\Delta)^\wedge.$$

*I per tant, l'ordre de $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$ és $\varphi_\Delta(a) = \sqcap_{i=1}^{l}\varphi_\Delta(p_i^{r_i})$.*

La demostració d'aquesta proposició consisteix en comptar els punts de la cònica $x^2 - \Delta y^2 = 1$ (la coneguda equació de Pell), en els espais $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$. Es pot trobar un esquema de la demostració en el llibre [44].

**Corol·lari 1.** *Sigui $n = pq$ on $p$ i $q$ són dos primers senars. L'ordre de $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$ és $\varphi_\Delta(n) = (p-(\frac{\Delta}{p}))(q-(\frac{\Delta}{q}))$*

Si intentem calcular la relació (4.4) mòdul $a\mathcal{O}_\Delta$ on $a$ és relativament primer amb $\Delta$ obtenim el següent resultat (com podem veure no només es centra amb els elements de $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$).

**Lema 1.** *Sigui $\Delta$ un enter no quadrat i $a$ un enter senar. Sigui $\alpha \in \mathcal{O}_\Delta$ i $x$, $y$ enters tal que $\alpha \equiv x + y\sqrt{\Delta} \pmod{a\mathcal{O}_\Delta}$. Llavors $\forall k \in \mathbb{N}$*

$$\alpha^k \equiv \frac{V_k}{2} + yU_k\sqrt{\Delta} \pmod{a\mathcal{O}_\Delta}.$$

**Dem.** Usarem inducció sobre k. Els casos $k = 0$ i $k = 1$ són trivials usant les condicions inicials de $U_k$ i $V_k$. Llavors, suposem cert per $k$ i $k-1$ i ho demostrem per k+1 usant les equacions (4.1) i (4.2) a part de les hipòtesis d'inducció.

$$\begin{aligned}
\alpha^{k+1} &\equiv P\alpha^k - Q\alpha^{k-1} \\
&\equiv P\frac{V_k}{2} + PyU_k\sqrt{\Delta} - Q\frac{V_{k-1}}{2} - QyU_{k-1}\sqrt{\Delta} \\
&\equiv \frac{V_{k+1}}{2} + y(PU_k - QU_{k-1})\sqrt{\Delta} \\
&\equiv \frac{V_{k+1}}{2} + yU_{k+1}\sqrt{\Delta}
\end{aligned}$$

□

**Corol·lari 2.** *Amb les condiciones del lema anterior,*

$$Tr(\alpha^k) \equiv V_k(P,Q) \mod a. \tag{4.5}$$

**Dem.** Usant el lema 1 i la definició de traça obtenim el resultat directament. □

Per definir les funcions de Lucas, hem vist la notació de les seqüències recursives enteres a les quals s'hi poden aplicar mòduls treballant així a $\mathbb{Z}/a\mathbb{Z}$, com en l'article [3]. Per una altra banda, també s'ha vist que aquestes funcions les podem entendre amb una notació exponencial (usada en [9] i [10] majoritàriament) dintre l'espai $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^*$. La importància del corol·lari anterior és que ens relaciona aquestes dues formes de veure les seqüències de Lucas.

Fins ara, ens hem trobat dos grups multiplicatius amb els que podem fer criptografia, $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^*$ i $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$; en termes de funcions de Lucas, en el primer tenim $Q \in \mathbb{Z}$, i en el segon restringim la $Q$ a 1. A la pràctica, usarem bàsicament Q=1, però en la propera secció veurem vàries propietats en funció de P i Q.

## 4.3 Propietats de la funció $V_k$

Com havíem comentat en l'observació 5, de les dues funcions de Lucas, la que té les propietats adients per ser una bona candidata a funció *trapdoor* és la $V_k$, i per això la majoria de les propietats que detallarem en aquesta secció són referents a aquesta.

El primer estudi extès de les funcions de Lucas el va fer Lehmer D.H. a [1], on ens detalla diverses propietats d'aquestes funcions, encara que, posteriorment, també en trobem de noves a [2], [3] i [6]. A continuació en tenim un recull.

**Propietat 2.** *Siguin k, P i Q enters positius*

$$V_k(P,Q)^2 = \Delta U_k(P,Q)^2 + 4Q^k \tag{4.6}$$

$$2U_{k+m}(P,Q) = U_k(P,Q)V_m(P,Q) + V_k(P,Q)U_m(P,Q) \tag{4.7}$$

$$2V_{k+m}(P,Q) = V_k(P,Q)V_m(P,Q) + \Delta U_k(P,Q)U_m(P,Q) \tag{4.8}$$

$$2Q^m V_{k-m}(P,Q) = V_k(P,Q)V_m(P,Q) - \Delta U_k(P,Q)U_m(P,Q) \tag{4.9}$$

**Dem.** Per demostrar aquestes quatre propietats, s'han d'usar les relacions (4.3), és a dir, canviar les funcions $U$ i $V$ a notació exponencial (tenint en compte que $\Delta = (\alpha - \beta)^2$ i $Q = \alpha\beta$) i operar. □

**Propietat 3** (Asimètrica)**.** *Sigui $P$ un enter, tenim que per tot $k \geq 0$*

$$V_k(-P, 1) = (-1)^k V_k(P, 1) \tag{4.10}$$

**Dem.** Siguin $-\alpha$ i $-\alpha^{-1}$ les arrels de $x^2 + Px + 1 = 0$, com que $(-1)^{-k} = (-1)^k$ llavors $V_k(-P, 1) = (-\alpha)^k + (-\alpha)^{-k} = (-1)^k(\alpha^k + \alpha^{-k}) = (-1)^k V_k(P, 1)$. □

Ara detallarem un lema molt útil per afirmar que la funció $V_n$ és una bona candidata a funció unidireccional. La primera part d'aquest lema es troba demostrada de dues formes diferents en els articles [3] i [9], però només detallaré la de [9] que treballa amb notació exponencial (en funció d'$\alpha$). Cal dir però, que l'altra demostració també és interessant ja que només treballa amb el polinomi irreductible i la funció $V_k$ en si.

**Lema 2.** *Per tot $k$, $m$, $P$ i $Q$*

$$V_{km}(P, Q) = V_m(V_k(P, Q), Q^k) \tag{4.11}$$

*i*

$$U_{km}(P, Q) = U_m(V_k(P, Q), Q^k)U_k(P, Q) \tag{4.12}$$

**Dem.** Usant la relació (4.6) i expressant $\alpha^k$ com en (4.4)

$$\alpha^k = \frac{V_k + U_k\sqrt{\Delta}}{2} = \frac{V_k + \sqrt{U_k^2\Delta}}{2} = \frac{V_k + \sqrt{V_k^2 - 4Q^k}}{2},$$

que si denotem $P' = V_k$, $Q' = Q^k$ i $\sqrt{\Delta'} = \sqrt{(P')^2 - 4Q'} = U_k\sqrt{\Delta}$ podem entendre l'$\alpha^k$ com un altre $\alpha$ amb uns nous P i Q (que seran $P'$ i $Q'$), és a dir, $\alpha^k$ serà l'arrel de l'equació $x^2 - P'x + Q' = 0$. Amb això aplicarem la relació (4.4) a $\alpha^k$

$$(\alpha^k)^m = \frac{V_m(P', Q') + U_m(P', Q')\sqrt{(P')^2 - 4Q'}}{2} = \frac{V_m(V_k(P, Q), Q^k) + U_m(V_k(P, Q), Q^k)U_k(P, Q)\sqrt{\Delta}}{2}$$

Per una altra banda, també sabem que

$$\alpha^{km} = \frac{V_{km}(P, Q) + U_{km}(P, Q)\sqrt{\Delta}}{2}.$$

Si comparem els coeficients, obtenim les igualtats que volíem. □

**Observació 8.** *Per estalviar càlculs computacionals de la funció $V_k$, considerem Q=1. D'aquesta forma, la relació (4.11) ens queda molt més senzilla*

$$V_{km}(P, 1) = V_k(V_m(P, 1), 1)$$

*Cal comentar, que segons [9] posant Q=1, no perdem seguretat, així que és evidentment recomanable usar sempre Q=1, i com hem dit en l'apartat 4.2, treballar amb l'$\alpha \in (\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$.*

Una altra propietat de [9] ens mostra que per k's molt grans pot ser fàcil computar la funció $V_k$ per quadrats i multiplicacions. Més endavant ja veurem com s'usa aquesta propietat.

**Propietat 4.** *Siguin $P$ i $Q$ enters positius i $k \geq m \geq 0$ també enters,*

$$V_{k+m} = V_k V_m - Q^m V_{k-m} \tag{4.13}$$

**Dem.** Si passem tots els termes en funció de $\alpha$ i $\beta$ obtenim

$$\alpha^{k+m} + \beta^{k+m} = (\alpha^k + \beta^k)(\alpha^m + \beta^m) - \alpha^m \beta^m (\alpha^{k-m} + \beta^{k-m}).$$

Operant les potències i arreglant-ho, s'acaba veient el que volem. □

Fins aquí, totes les propietats que hem esmentat, es compleixen independentment d'en quin context es trobi la funció $V_k$. Si concretem la $a$ de l'espai $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$ (és a dir prenem mòduls), que és bàsicament on treballarem, existeixen d'altres propietats interessants. Tractarem dos casos que es diferenciaran en la tria de $a$; en el primer cas $a$ serà un primer $p \neq 2$ i en l'altre un enter RSA, és a dir, $a = pq$ on $p$ i $q$ són primers.

### 4.3.1 Cas $\mathcal{O}_\Delta/p\mathcal{O}_\Delta$

Començarem amb el cas $a = p$, on la seguretat dels esquemes dissenyats es basarà en el logaritme discret. Si $(\frac{\Delta}{p}) = -1$, llavors podem afirmar que $(\mathcal{O}_\Delta/p\mathcal{O}_\Delta)^\wedge \cong GF(p^2)$, en canvi, si $(\frac{\Delta}{p}) = 1$ tenim que $(\mathcal{O}_\Delta/p\mathcal{O}_\Delta)^\wedge \cong \mathbb{Z}_p$.

**Lema 3.** *Sigui $P$ un enter, $p$ un enter primer i $\alpha$ una arrel de $f(x) = x^2 - Px + 1$ polinomi irreductible sobre $GF(p)$. Llavors, si $t$ és l'ordre d'$\alpha$, $t \mid (p+1)$ i $t \nmid (p-1)$*

**Dem.** El fet que $t \nmid (p-1)$ està clar ja que $\alpha \notin GF(p)$. Per veure $t \mid (p+1)$ considerem l'automorfisme de Frobenius $x \mapsto x^p$ en $GF(p^2)$. Se sap que aquest permuta les arrels de f. Per tant, com que el producte de les dues arrels conjugades és 1, tenim que $\alpha^p = \alpha^{-1}$. D'això deduïm $\alpha^{p+1} = \alpha^p \alpha = \alpha^{-1}\alpha = 1$ i com que t és l'ordre d'$\alpha$, llavors $t \mid (p+1)$. □

**Remarca 1.** *Si el polinomi $f(x) = x^2 - Px + 1$ no fos irreductible, $\alpha$ seria un residu quadràtic i llavors el seu ordre $t$ verificaria $t \mid (p-1)$.*

A partir d'aquí mantenint $(\frac{\Delta}{p}) = -1$, gràcies a l'equació (4.4), veiem que la funció $V_k$ és periòdica i si el seu període és t, tenim que $V_t = V_0$. Ara seria interessant veure que el període de $V_k$ i l'ordre d'$\alpha$ fossin iguals.

**Teorema 2.** *Sigui $t$ el període de $V_k$ i $T$ l'ordre d'$\alpha$, $T$=$t$*

**Dem.** Com que el cas $t \leq T$ és trivial, només cal veure que $T \leq t$, és a dir, que si t és el període de $V_k$ llavors $\alpha^t = 1$. Com que t és el període de $V_k$ tenim $V_t = V_0 = \alpha^t + \alpha^{-t} = 2$. I per això $\alpha^t$ i $\alpha^{-t}$ són arrels del polinomi $f(x) = x^2 - 2x + 1 = (x-1)^2$. Per tant, com volíem $\alpha^t = 1$. □

Seguidament, podem enunciar i demostrar la *fold property*, que ens explica que les imatges de la funció $V_k$ estan aparellades.

**Propietat 5** (*Fold property*)**.** *Sigui $P$ un enter i $t$ el període de $V_k(P)$, es compleix que $V_k(P) = V_{t-k}(P)$ per qualsevol enter $k$ tal que $0 \leq k \leq t$.*

**Dem.** Com que el període de $V_k$ és el mateix que l'ordre d'$\alpha$, per qualsevol k enter entre 0 i t podem afirmar

$$V_{t-k}(P) = \alpha^{t-k} + (\alpha^{-1})^{t-k} = \alpha^{-k} + \alpha^k = V_k(P).$$

□

A [2], es troba el següent teorema que ens dona el valor de les funcions de Lucas per uns certs valors de k i que serà fonamental per fer criptografia amb les funcions de Lucas com veurem posteriorment. La base d'aquest teorema, per això, es troba en l'article de Lehmer [1].

**Teorema 3.** *Sigui p un primer imparell, $p \nmid Q$ i $\epsilon = \left(\frac{\Delta}{p}\right)$, llavors $\forall m \in \mathbb{Z}$*

$$U_{m(p-\epsilon)}(P,Q) \equiv 0 \pmod{p} \tag{4.14}$$
$$V_{m(p-\epsilon)}(P,Q) \equiv 2Q^{m(1-\epsilon)/2} \pmod{p} \tag{4.15}$$

**Dem.** Com queda demostrat en [1], $U_{p-\epsilon}(P,Q) \equiv 0 \pmod{p}$ i $V_{p-\epsilon}(P,Q) \equiv 2Q^{(1-\epsilon)/2} \pmod{p}$. Llavors, usant inducció sobre m en ambdós casos i amb les equacions (4.7) i (4.8) respectivament, es demostren les dues congruències. □

Quan $(\frac{\Delta}{p}) = 1$, hi ha una propietat que es presenta en [8], per usar-la en l'esquema d'una signatura digital. Ens escriu la relació (4.8) únicament en funció de $V_k$, usant la relació (4.6).

**Propietat 6.** *Sigui P un enter, p primer i $(\frac{\Delta}{p}) = 1$*

$$2V_{k+m}(P) \equiv V_k(P)V_m(P) \pm \sqrt{(V_k(P)^2 - 4)(V_m(P)^2 - 4)} \mod p \tag{4.16}$$

### 4.3.2 Cas $\mathcal{O}_\Delta/pq\mathcal{O}_\Delta$

En el cas que $a = pq$ on p i q són enters primers, si Q=1 tenim que $(\mathcal{O}_\Delta/pq\mathcal{O}_\Delta)^\wedge$ és un grup multiplicatiu amb l'ordre que hem vist en el corol·lari (1).

Primer veurem un corol·lari del teorema (3) que s'obté usant també el teorema dels residus xinesos enunciat en la secció 3.

**Corol·lari 3.** *Sigui n=pq, llavors $\forall m \in \mathbb{Z}$*

$$U_{m(p^2-1)(q^2-1)}(P) \equiv 0 \pmod{n} \tag{4.17}$$
$$V_{m(p^2-1)(q^2-1)}(P) \equiv 2 \pmod{n} \tag{4.18}$$

**Definició 3.** *Sigui $n = pq$ amb p i q enters senars, definim els espais $\Lambda_n$ i $\Lambda'_n$ de la següent manera:*

$$\Lambda_n = \{P \in \mathbb{N}, P < n, mcd(P^2 - 4, n) = 1\}$$
$$\Lambda'_n = \{P \in \mathbb{N}, 0 < P < n, mcd(P^2 - 4, n) = 1, mcd(P, n) = 1\}$$

**Proposició 4.** *Sigui $e \in \mathbb{Z}$ tal que $(e, (p+1)(p-1)(q+1)(q-1)) = 1$, la funció $V_e(P) \pmod{n}$ és una permutació dels conjunts $\Lambda_n$ i $\Lambda'_n$.*

**Dem.** Començarem veient que la funció està ben definida, és a dir, que si $P \in \Lambda_n$ llavors $V_e(P) \in \Lambda_n$. Prenem $\Delta \equiv P^2 - 4 \pmod{n}$ i $\alpha \equiv \frac{P+\sqrt{\Delta}}{2} \pmod{n\mathcal{O}_\Delta}$. Com que $N(\alpha) \equiv 1 \pmod{n}$, pel lema 1 tenim que $\alpha^e \equiv \frac{V_e(P)+U_e(P)\sqrt{\Delta}}{2} \pmod{n\mathcal{O}_\Delta}$. Si fem la norma d'$\alpha^e$ obtenim $N(\alpha^e) \equiv \frac{V_e(P)^2}{4} + \Delta\frac{U_e(P)^2}{4} \pmod{n}$. Com que $N(\alpha^e) \equiv 1 \pmod{n}$, de l'expressió anterior podem arribar a que $V_e(P)^2 - 4 \equiv U_e(P)^2\Delta \pmod{n}$. Per tant, $V_e(P) \in \Lambda_n$ si $\mathrm{mcd}(U_e(P)^2, n) = 1$ ja que $\Delta$ ja és relativament primer a n.
En efecte, pel corol·lari 1 l'ordre de $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$ és $\varphi_\Delta(n) = (p-(\frac{\Delta}{p}))(q-(\frac{\Delta}{q}))$, i com que hem pres e tal que $(e, (p+1)(p-1)(q+1)(q-1)) = 1$, llavors l'homomorfisme $\alpha \mapsto \alpha^e$ és un automorfisme del grup $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$. Si prenem d de manera que $d \equiv e^{-1} \pmod{\varphi_\Delta(n)}$ i apliquem el lema 1 per calcular $\alpha^{ed}$ obtenim

$$\alpha^{ed} \equiv \frac{V_d(V_e(P)) + U_e(P)U_d(V_e(P))\sqrt{\Delta}}{2} \pmod{n\mathcal{O}_\Delta}$$

A partir d'aquesta expressió, i com que $\alpha^{ed} \equiv \alpha \pmod{n\mathcal{O}_\Delta}$, podem afirmar un parell de coses.

1. $U_e(P)U_d(V_e(P)) \equiv 1 \pmod{n}$
2. $V_d(V_e(P)) \equiv P \pmod{n}$

De 1 deduïm que $U_e(P)$ és relativament primer amb n i per tant la funció està ben definida, i de 2 que la funció $V_e$ és una permutació de $\Lambda_n$.
Ara ja només queda veure que la funció $V_e$ també és una permutació de $\Lambda'_n$. Per aconseguir-ho, només cal veure que els elements $P$ amb $mcd(P, n) \neq 1$ es queden en el mateix conjunt. Si fem mòdul $p$ (de la mateixa manera amb mòdul q), s'ha de demostrar que $V_e(0) \equiv 0 \pmod{p}$, cosa que per la relació (4.5) és trivial. □

## 4.4 Seguretat de les funcions de Lucas

En els apartats anteriors s'ha definit la nova funció $V_k$ i ja s'ha comentat que és una bona candidata a funció *trapdoor*.

Una funció és *trapdoor* si és fàcil de computar en una direcció, però molt difícil en l'altre, a no ser que tinguem alguna informació adicional que ens ajudi, a la que anomenem *trapdoor*. En aquest apartat, es veurà la dificultat que representa invertir la funció $V_k$.

En el cas de l'exponenciació en grups finits, hi ha un problema conegut que és el del logaritme discret, amb el qual es basa la seguretat de moltes aplicacions criptogràfiques com el sistema criptogràfic ElGamal.

**Problema 1.** *(Problema del logaritme discret) Sigui $G$ un grup cíclic finit de $p$ elements (amb $p$ primer) i $g$ un generador d'aquest grup. Donat un $a \in G$ diferent de $g$, cal trobar $k$ tal que $g^k \equiv a \pmod{p}$.*

Actualment, el algorismes més eficients per resoldre aquest problema en un grup general $G$ són els algorismes subexponencials com l'*Index Calculus*.

Amb la funció $V_k$ podem definir un problema molt semblant al del logaritme discret, del qual interessa saber la seva dificultat.

**Problema 2.** *Sigui $p$ un primer diferent de 2, i $P$ un enter tal que $\left(\frac{P^2-4}{p}\right) = -1$ i que 1 i $p+1$ siguin els únics divisors $k$ de $p+1$ tals que $V_k(P) \equiv 2 \mod p$. Donat $a \in \mathbb{Z}_p$ tal que $a \equiv V_k(P) \pmod p$, cal trobar $k$.*

**Observació 9.** *En el problema 2, s'imposa la condició de que $\left(\frac{P^2-4}{p}\right) = -1$ perquè el grup on treballem sigui isomorf a $GF(p^2)$, com ja s'havia comentat en l'apartat de les propietats de $V_k$. Si tinguéssim $\left(\frac{P^2-4}{p}\right) = 1$ treballaríem a $\mathbb{Z}_p^*$ i per això es reduïria al problema del logaritme discret. La segona condició del problema (2) és necessària perquè $P$ sigui un generador del grup $(\mathcal{O}_\Delta/p\mathcal{O}_\Delta)^\wedge$, que pel lema (3) té ordre $p+1$.*

En quant a la seguretat del problema 2, en l'article [11] es demostra que és polinòmicament equivalent a la seguretat del problema 1. Per tant, la manera més ràpida de resoldre el problema 2 és en temps subexponencial. Els atacs que poden resoldre el problema 1 són en temps $L_p\left[\frac{1}{3}, (\frac{64}{9})^{\frac{1}{3}} + o(1)\right]$, per $p \to \infty$, on es denota $L_m[u,v] = e^{v(\log m)^u (\log\log m)^{1-u}}$, per computar un logaritme discret en un grup d'ordre $\approx m$ (veure [23] i [24]).

Per una altra banda, degut a la relació (4.5) i als paràmetres $p$, $P$ i $\Delta$ escollits, el problema 2 és equivalent al problema 1 en el grup multiplicatiu $GF(p^2)$. Per això, per resoldre el problema (2) necessitem resoldre el logaritme discret en un grup d'ordre $\approx p^2$, i per tant es necessita un atac subexponencial que requereix un temps de $L_{p^2}\left[\frac{1}{3}, (\frac{64}{9})^{\frac{1}{3}} + o(1)\right] = L_p\left[\frac{1}{3}, (\frac{128}{9})^{\frac{1}{3}} + o(1)\right]$. Per tant, es pot concloure que l'atac subexponencial tarda més temps amb el problema 2 que amb el primer.

En criptografia, hi ha un altre problema important, el problema RSA, del qual també veurem la seva versió amb la funció $V_k$.

**Problema 3.** *(Problema RSA) Siguin $n$, $e$ i $c$ enters tals que $N$ és el producte de dos enters primers, $2 < e < n$ és coprimer amb $\varphi(n)$ i $c$ triat aleatoriament, cal trobar l'enter $M$ tal que $M^e \equiv c \pmod n$.*

Fins ara, la manera més eficient coneguda de resoldre el problema RSA és factoritzant $n$ i calculant l'invers de $e$ mòdul $\varphi(n)$, cosa que és impracticable si $n$ és suficientment gran.

**Problema 4.** *Siguin $n$, $e$ i $c$ enters primers tals que $N$ és el producte de dos enters, $2 < e < n$ és coprimer amb $\varphi_\Delta(n)$ i $\alpha$ triat aleatoriament de $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$, s'ha de trobar $M$ tal que $V_e(M) \equiv Tr(\alpha) \pmod n$.*

Per decidir quin d'aquests dos problemes és criptogràficament més fort, cal mirar-los d'una perspectiva ben diferent. A continuació veiem que tant el problema (3) com el (4) es poden plantejar com trobar la solució d'un polinomi.

**Propietat 7.** *La funció $V_e(M)$ és un tipus especial de polinomi de Dickson i es pot provar per inducció que és de la forma següent.*

$$V_e(M) = \sum_{i=0}^{\lfloor e/2 \rfloor} \frac{e}{e-1}\binom{e-1}{e} M^{e-2i}.$$

**Observació 10.** *Per resoldre el problema RSA hem de trobar una arrel del polinomi $M^e - c$ i pel de Lucas, una arrel de $V_e(M) - Tr(\alpha)$, que també és un polinomi gràcies a la propietat anterior. Per tant, per resoldre els dos problemes, s'ha de trobar una arrel d'un polinomi de grau $e$ i per això, sembla ser que aquests dos problemes tenen el mateix nivell de seguretat.*

## 4.5 Computació de la funció $V_k(M)$

En l'apartat 4.4 s'ha fet referència a una funció *trapdoor* com una funció fàcil de computar en una direcció; això significa que la $V_k(M)$ s'ha de poder computar en temps polinòmic per usar-la en criptografia. Amb els algorismes que s'explicaran en aquest apartat, veurem que sí que s'assoleix aquest temps, però lo realment interessant és la comparació amb la computació de la funció exponenciació $M^k \pmod n$.

L'exponenciació modular és molt usada en la criptografia de clau pública en esquemes tan comunment usats com l'RSA. Per això és imprescindible una avaluació eficient de $M^k \pmod n$.

El mètode d'exponenciació binària, que és el més usat, computa $M^k \pmod n$ usant l'expressió binària de l'exponent $k$ aconseguint trencar l'exponenciació en una sèrie de quadrats i multiplicacions. Hi ha dues variants molt semblants d'aquest mètode, una que comença a recorre els bits de $k$ des del menys significatiu fins al més significatiu, i un altre que els recorre al revés. El primer consisteix en dues variables, $C = 1$ i $S = M$ que per cada bit es van actualitzant. Si el bit de la iteració és 1, la variable $C = C \cdot S \pmod n$, en canvi, $S$ cada vegada és multiplica per $M$ mòdul $n$ independentment del valor del bit (quan el bit és 0 només es fa una operació). Al final, el resultat de l'exponenciació és el valor que conté la variable C. Els dos mètodes estan explicats detalladament en [29]. Com que es recorre l'expressió binària de $k$, hi haurà $\log k$ iteracions; i en cada iteració, suposant que hi ha les mateixes probabilitats de que un bit sigui 1 o 0, per la meitat de bits es necessitaran dues multiplicacions modulars ($C \cdot S$ i $S \cdot S$) i per l'altre meitat només una. Per tant de mitja, la complexitat d'aquest algorisme serà de $\frac{3}{2} \log k$.

Per millorar els algorismes anteriors, S.-M Yen i C.-S. Laih [30] van desenvolupar el mètode CMM (*Common-Multiplicand Multiplication*) on es centra en computacions del tipus $\{X \cdot Y_i \mid i = 1, 2, \ldots t;\ t \geq 2\}$; el cas anterior és el cas $t = 2$. La idea bàsica del CMM és extraure les parts comuns dels multiplicands i guardar el nombre de sumes binàries per la computació d'aquestes parts. En mitja, aplicant aquest mètode a l'exponenciació binària, s'aconsegueix computar l'exponenciació modular amb $\frac{5}{4} \log k$, sensiblement millor que l'exponenciació binària.

Hi ha d'altres mètodes que es poden combinar amb l'exponenciació binària, per reduir encara més aquest nombre d'operacions com la reducció modular de Montgomery [31] i el mètode SDR (*Signed-Digit Recoding*) [32] i [33].

En quant a la funció $V_k$, des de que P. Smith i J. Lennon van presentar les funcions de Lucas com una eina criptogràfica en l'article [3], hi ha hagut molts intents de millorar la computació de $V_k(M)$. Evidentment, la computació de $V_k(M)$ segons la seva definició és inviable ja que degut a ser recursiva, per valors grans de $k$ és extremadament lenta.

La primera proposta és una tècnica semblant al mètode de multiplicació *russian peasant* dels mateixos P. Smith i J. Lennon. Primer s'expressa $k$ en binari

$$k = (b_0 b_1 \ldots b_t), \tag{4.19}$$

on $t = \log k$, $b_0$ és el bit més significatiu i $k = \sum_{i=0}^{t} b_i 2^{t-i}$, i es defineix $k_p$ com la suma parcial $k_p = \sum_{i=0}^{p} b_i 2^{p-i}$. L'algorisme consistirà en anar calculant els $V_{k_p}(M)$ i $V_{k_p-1}(M)$,

des de $p = 0$ fins a $p = t$, tenint en compte que $V_{k_0} = V_1 = P$ i $V_{k_0-1} = V_0 = 2$.

Per tant, per calcular $V_{k_{p+1}}(M) \pmod n$ i $V_{k_{p+1}-1}(M) \pmod n$ coneixent $V_{k_p}(M) \pmod n$ i $V_{k_p-1}(M) \pmod n$, s'usen

$$\begin{aligned} R_p &\equiv V_{2k_p-1}(M) \pmod n \\ S_p &\equiv V_{2k_p}(M) \pmod n \\ T_p &\equiv V_{2k_p+1}(M) \pmod n. \end{aligned}$$

$R_p$ i $S_p$ s'obtenen amb l'equació (4.13), prenent $k := k_p$ i $m := k_p - 1$, i $k = m := k_p$ respectivament. El valor de $T_p$ s'obten fàcilment usant la definició de $V_k$, de manera que queda $T_p \equiv MS_p - R_p \pmod n$. Llavors, si el bit $b_{p+1} = 0$, $V_{k_{p+1}}(M)$ pren per valor $S_p$ i $V_{k_{p+1}-1}(M)$ pren per valor $R_p$ ja que $k_{p+1} = 2k_p$; en canvi, si $b_{p+1} = 1$, com que $k_{p+1} = 2k_p + 1$, $V_{k_{p+1}}(M)$ pren per valor $T_p$ i $V_{k_{p+1}-1}(M)$ pren per valor $S_p$.

**Exemple 3.** *Pel cas $V_{53}$ on $53 = (110101)$, es mostren les iteracions (una per columna) que fa l'algorisme en la següent graella. En la primera fila hi ha els bits de $k$ tal i com es van recorrent i en la primera columna hi ha els $V_{k_p-1}$ i $V_{k_p}$ que s'acumulen i els $R_p$, $S_p$ i $T_p$ que es van calculant.*

| | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|
| $V_{k_p-1}$ | $V_0$ | $V_2$ | $V_5$ | $V_{12}$ | $V_{25}$ | $V_{52}$ |
| $V_{k_p}$ | $V_1$ | $V_3$ | $V_6$ | $V_{13}$ | $V_{26}$ | $\mathbf{V_{53}}$ |
| $R_p$ | $V_1$ | $V_5$ | $V_{11}$ | $V_{25}$ | $V_{51}$ | |
| $S_p$ | $V_2$ | $V_6$ | $V_{12}$ | $V_{26}$ | $V_{52}$ | |
| $T_p$ | $V_3$ | $V_7$ | $V_{13}$ | $V_{27}$ | $V_{53}$ | |

**Observació 11.** *Com es pot veure en l'exemple, per cada bit de $k$ excepte el menys significatiu, es computen $R_p$, $S_p$ i $T_p$. Com que el càlcul de cada una d'aquestes variables costa una multiplicació, el cost d'aquest algorisme és $3 \log k$ multiplicacions.*

Més endavant, Laih et al. van dissenyar a [5] dos algorismes més eficients que l'anterior, el *left-to-right scanning* i el *rigth-to-left scanning*. Els noms dels algorismes són deguts a la manera de llegir els bits de $k$, el primer comença pel bit més significatiu fins al menys significatiu, com en l'algorisme de Smith i Lennon, i en canvi el segon els llegeix al revés.

L'algorisme *left-to-right scanning* és basa en que, usant la notació anterior, $k_p = 2k_{p-1} + b_p = (k_{p-1} + b_p) + k_{p-1}$, i a l'hora de calcular $V_{k_p}(M)$, tenint en compte la relació (4.13), resulta

$$V_{k_p}(M) = V_{k_{p-1}+b_p} V_{k_{p-1}}(M) - V_{b_p}(M). \tag{4.20}$$

Per tant, per aconseguir $V_{k_p}(M)$ es necessita $V_{k_{p-1}}(M)$ i $V_{k_{p-1}+1}(M)$. Per això, per cada bit $b_p$ es calcularà $V_{k_p}(M)$ i $V_{k_p+1}(M)$, per així poder calcular la següent parella fins a arribar al $V_k$ que volem. Fixant-nos amb els subíndex de la relació (4.20) i denotant $(k_{p-1}, k_{p-1} + 1) = (l, h)$, el càlcul es divideix en els següents casos.

- *Cas $b_p = 0$*: $(k_p, k_p + 1) = (2k_{p-1}, 2k_{p-1} + 1) = (2l, l + h)$.
- *Cas $b_p = 1$*: $(k_p, k_p + 1) = (2k_{p-1} + 1, 2k_{p-1} + 2) = (l + h, 2h)$.

Finalment, en cada iteració de l'algorisme, si $b_p = 0$ $V_l(M) = V_h(M)V_l(M) - M$ i $V_h(M) = V_h^2(M) - 2$, i si $b_p = 1$ $V_h(M) = V_h(M)V_l(M) - M$ i $V_l(M) = V_l^2(M) - 2$.

**Exemple 4.** *Igual que en l'exemple 3, es mostren les iteracions per calcular $V_{53}$, però usant l'algorisme* left-to-right scanning.

| | | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| $V_l$ | $V_0$ | $V_1$ | $V_3$ | $V_6$ | $V_{13}$ | $V_{26}$ | $\mathbf{V_{53}}$ |
| $V_h$ | $V_1$ | $V_2$ | $V_4$ | $V_7$ | $V_{14}$ | $V_{27}$ | $V_{54}$ |

**Observació 12.** *Aquest algorisme s'inicialitza amb $V_l(M) = 2$ i $V_h(M) = M$, i es recorren tots els bits de $k$. Com que en cada iteració es fan dues multiplicacions, el cost de l'algorisme és $2 \log k$.*

Com hem comentat abans, l'algorisme *right-to-left scanning* recorre el paràmetre $k$ en sentit contrari, i per això és més útil usar la notació següent.

$$k = (b_t \dots b_1 b_0),$$

on $t = \log k$, $b_t$ és el bit més significatiu i $k = \sum_{i=0}^{t} b_i 2^i$, i es defineix $k_p$ com la suma parcial $k_p = \sum_{i=0}^{p} b_i 2^i$. L'estratègia per calcular $V_k(M)$, és obtenir $V_{k_p}(M)$ a partir de $V_{k_{p-1}}(M)$. La vairable que guardi el valor $V_{k_p}(M)$ en cada iteració es denotarà $MU$. Per tant, coneixent $MU = V_{k_{p-1}}(M)$ i tenint en compte que $V_{k_p}(M) = V_{b_p 2^p + k_{p-1}}(M)$, s'haurà de considerar els dos següents casos.

- $\underline{Cas\ b_p = 0}$: la nova $MU = V_{k_p}(M)$ serà $MU = V_{k_{p-1}}(M)$, per tant es queda igual.
- $\underline{Cas\ b_p = 1}$: la nova $MU = V_{k_p}(M)$ serà $MU = V_{2^p + k_{p-1}}(M) = V_{2^p}(M)V_{k_{p-1}}(M) - V_{2^p - k_{p-1}}(M)$ on es necessitarà calcular $V_{2^p}(M)$ i $V_{2^p - k_{p-1}}(M)$.

Per aconseguir $V_{2^p}(M)$ tindrem una variable $S$ que a cada iteració doblarà la variable, $S_{\text{nova}} = S^2_{\text{anterior}} - 2$. Per l'altra part que s'ha de calcularà, s'usarà una altra variable $AD$ i també s'ha de tractar en dos casos diferents. Si ja es coneix l'anterior $AD = V_{2^p - k_{p-1}}(M)$,

- $\underline{Cas\ b_p = 0}$: la nova $AD = V_{2^{p+1} - k_p}(M)$ serà $AD = V_{2^p 2 - k_{p-1}}(M) = V_{2^p + (2^p - k_{p-1})}(M) = V_{2^p}(M)V_{2^p - k_{p-1}}(M) - V_{k_{p-1}}(M)$, que són $S$, $AD$ i $MU$ respectivament.
- $\underline{Cas\ b_p = 1}$: la nova $AD = V_{2^{p+1} - k_p}(M)$ serà $AD = V_{2^p + 2^p - (2^p + k_{p-1})}(M) = V_{2^p - k_{p-1}}(M)$ per tant es queda igual.

Per tant, es pot resumir que per cada bit $b_p$, si aquest és 1, s'actualitzarà la $MU = S * MU - AD$ i si és 0, $AD = S * AD - MU$. La $S$ s'actualitzarà cada cop.

**Exemple 5.** *Com l'exemple 3, es mostren les iteracions per calcular $V_{53}$, però usant l'algorisme* right-to-left scanning.

| | | 1 | 0 | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| $MU$ | $V_0$ | $V_1$ | $V_1$ | $V_5$ | $V_5$ | $V_{21}$ | $\mathbf{V_{53}}$ |
| $S$ | $V_1$ | $V_2$ | $V_4$ | $V_8$ | $V_{16}$ | $V_{32}$ | $V_{64}$ |
| $AD$ | $V_1$ | $V_1$ | $V_3$ | $V_3$ | $V_{11}$ | $V_{11}$ | $V_{11}$ |

**Observació 13.** *Com l'algorisme* left-to-right scanning*, aquest només fa dues multiplicacions per cada iteració, la de la variable $S$ i depenent de si el bit és 0 o 1, $AD$ o $MU$ respectivament, per tant, el cost és de $2 \log k$. La única diferència és que aquest algorisme usa una variable més que l'altre, però és menyspreable.*

**Observació 14.** *Els dos algorismes* left-to-right scanning *i* right-to-left scanning*, com hem dit, fan dues multiplicacions modulars en cada iteració. Aquestes multiplicacions tenen la peculiaritat de ser del tipus* $\{X \cdot Y_i \mid i = 1, 2, \ldots t;\ t \geq 2\}$*, i per tant se'ls hi pot aplicar el mètode CMM de [30] per reduir el temps de computació. Concretament, els productes són* $\{V_h \cdot V_l, V_h \cdot V_h\}$ *i* $\{V_l \cdot V_h, V_l \cdot V_l\}$ *en el primer, i* $\{S \cdot MU, S \cdot S\}$ *i* $\{S \cdot AD, S \cdot S\}$ *en el segon.*

Una altra eina per millorar la computació de la funció $V_k(M) \pmod n$ són les cadenes de Lucas.

**Definició 4.** *Una* cadena d'adició *d'un nombre* $n$ *és una seqüència d'enters* $1 = a_0 < a_1 < \ldots < a_r = n$ *tal que cada nombre posterior a* $a_0$ *és la suma de dos anteriors.*

**Exemple 6.** $\{1, 2, 4, 6, 8, 14\}$ *és una cadena d'adició de 14.*

**Definició 5.** *Una* cadena de Lucas *d'un nombre* $n$ *és una cadena d'adició* $a_0 < a_1 < \ldots < a_r$ *tal que*

1. $a_0 = 0$*,* $a_1 = 1$ *i* $a_r = n$
2. $a_i = a_j + a_k$ *per alguns* $j$ *i* $k$ *tals que* $i > j \geq k$ *i* $(a_j - a_k) \in \{a_0, \ldots, a_r\}$

**Exemple 7.** $\{0, 1, 2, 3, 4, 6, 7, 13\}$ *és una cadena de Lucas de 13.*

A [5], afirmen que una forma específica d'avaluar $V_k$ pot ser transformada en la construcció d'una cadena de Lucas de l'enter $k$. Per tant, el valor de $V_k$ pot ser computat amb el mínim nombre de multiplicacions si i només si, es pot trobar la cadena de Lucas de $k$ més curta (un problema molt difícil).

C.T. Wang et al. [13] van desenvolupar un mètode heurístic per generar cadenes de Lucas i poder-les usar per computar les seqüències de Lucas. Aquest mètode es basa en quatre lemes per anar construint la cadena de Lucas.

**Lema 4.** *Sigui* $k$ *un ener*

1. $V_{2k} = V_k V_k - V_0$
2. $V_{3k} = V_{2k} V_k - V_k$
3. $V_{5k} = V_{3k} V_{2k} - V_k$

**Lema 5.** *Suposant que* $k$ *és el* $k$*-èssim terme d'una seqüència de Lucas, si* $k$ *es descomposa com el lema 4, llavors la subcadena de Lucas de* $k$ *pot ser avaluada.*

**Lema 6.** *Sigui* $x$ *un enter senar,*

$$V_x = V_{\lfloor \frac{x}{2} \rfloor + 1} V_{\lfloor \frac{x}{2} \rfloor} - V_1.$$

**Lema 7.** *Sigui* $y$ *un enter senar, i* $x = \lfloor \frac{y}{2} \rfloor$*. La cadena* $(x, x+1, y)$ *forma una subcadena si els termes* $\lfloor \frac{x}{2} \rfloor$ *i* $\lfloor \frac{x}{2} \rfloor + 1$ *són inclosos en la cadena*

Per construir la cadena de Lucas, primer es comprova si $k$ es pot descomposar com en el lema 4 fins que no sigui possible i llavors es va descomposant com en el lema 6. Un cop s'obté la cadena de Lucas, s'ha d'anar construint la seqüència seguint la cadena com en l'algorisme B de [13] fins a l'element $V_k(M) \pmod n$.

**Observació 15.** *Aquest mètode és heurístic ja que si resulta que k no és múltiple de 2, de 3, ni de 5, seguirà tenint un cost* $2\log_2 k$ *(fins ara no havíem especificat que el logaritme era en base 2 perquè era obvi) com abans. Però com s'explica [13], el cost mitjà d'aquest algorisme és de* $\log_2 k + \log_3 k$ *multiplicacions modulars, és a dir, es redueixen un 19 per cent comparant amb els algorismes de [5].*

Un altre mètode per trobar la cadena de Lucas més curta va ser descrit per S.Chiou et al. [14]. Aquest consisteix en dos algorismes que calculen dues subcadenes tals que si s'uneixen formen una cadena de Lucas. Aquest mètode redueix un 13 per cent el cost dels algorismes de [5]. A diferència de l'anterior, aquest algorisme és determinista.

**Observació 16.** *Els dos algorismes esmentats que cerquen cadenes de Lucas, tenen l'inconvenient que requereixen una quantitat important de memòria per emmagatzemar la cadena de Lucas. L'avantatge dels algorismes de [5], és que ja van calculant el valor de la funció de Lucas durant la primera i única iteració.*

També cal comentar que, igual que amb l'RSA, amb les funcions de Lucas també es pot usar el teorema dels residus xinesos per millorar la computació, si s'escau.

Finalment, hi ha d'altres intents de millorar en la computació de la funció de Lucas com en [21] o [15].

## 4.6 Aplicacions de les funcions de Lucas

Com s'ha vist fins ara, la funció $V_k$ és una bona candidata per fer criptografia de clau pública degut a la seva seguretat i computació eficient. En aquesta secció hi ha un recull de totes les aplicacions on es pot usar la funció $V_k$, començant pel mètode de Diffie-Hellman fins a sistemes de xifrat i tot tipus d'esquemes de signatures digitals.

### 4.6.1 Versió Lucas del mètode Diffie-Hellman

Aquest mètode, descobert el 1976 per W. Diffie i M. Hellman, és un protocol criptogràfic que permet cercar una clau secreta compartida per dues parts que només es poden comunicar en un canal insegur. La descripció del protocol és la següent.

1. Els dos usuaris es posen d'acord en un grup cíclic i finit G i un element generador g de G. Els paràmetres g i G són públics, és a dir, qualsevol possible atacant pot accedir-hi.
2. El primer usuari escull a l'atzar un $a \in \mathbb{N}$ i envia al segon usuari $g^a$.
3. El segon usuari escull a l'atzar un $b \in \mathbb{N}$ i envia al primer usuari $g^b$.
4. El primer usuari computa $(g^b)^a$.
5. El segon usuari computa $(g^a)^b$.
6. Els dos usuaris són els únics coneixedors de la clau $g^{ab}$.

Les funcions de Lucas també ens permeten dissenyar un esquema semblant al mètode Diffie-Hellman perquè dos usuaris puguin trobar públicament una clau privada.

Primer de tot, s'hauran de posar d'acord en la tria de $p$ i $P$ que cumpleixin les condicions del problema (2) per tal de que l'esquema sigui segur. Llavors publicaran $V_x(P)$ i $V_y(P) \mod p$ on $x$ i $y$ són les respectives claus privades. Finalment, gràcies a la obervació (8) la clau privada comú serà

$$V_{xy}(P) \mod p$$

i els dos usuaris la podran aconseguir.

### 4.6.2 Sistemes de xifrat de clau pública

Els criptosistemes de clau pública tenen com a objectiu principal enviar un missatge M des d'un emissor cap a un receptor, els quals comparteixen una clau pública. En aquests criptosistemes, el receptor tria la clau pública $p_k$ i la privada $s_k$, i llavors es duen a terme els dos algorismes següents:

- *Encriptació*: L'emissor computa $c = Enc_{p_k}(M)$, i envia $c$ al receptor per un canal públic.
- *Desencriptació*: El receptor rep $c$ i computa $M = Dec_{s_k}(c)$.

Aquests criptosistemes poden ser deterministes o probabilístics. En el cas dels deterministes, per cada missatge $M$ sempre s'obté el mateix criptograma com per exemple l'RSA. Els probabilístics, en canvi, usen l'aleatorietat a l'hora d'encriptar, és a dir, que quan encriptes un mateix missatge varis cops, generalment s'obtenen diferents xifrats.

El primer sistema probabilistic de clau pública segur va ser proposat per S.Goldwasser i S.Micali, i estava basat en la dificultat del problema dels residus quadràtics. Després se'n van proposar molts més, els més coneguts dels quals són ElGamal, Paillier, Catalano i vàries construccions sota el *Random Oracle Model*. En aquesta secció veurem una versió de l'esquema ElGamal i una altra del de Catalano et al. usant funcions de Lucas.

#### Sistema de clau pública LUC

En la criptografia de clau pública, actualment l'RSA és el sistema determinista més usat en tot tipus de protocols. Aquest, dissenyat per Rivest, Shamir i Adleman, de moment està considerat segur per paràmetres de l'ordre de 512 i 1024 bits, a part de ser computablement bastant eficient.

Primerament l'usuari que ha de rebre el missatge M escull dos primers senars $p$ i $q$ (que mantindrà en secret) i publicarà $n = pq$. Llavors tria la clau pública $e$ tal que $1 < e < \varphi(n)$ amb $e$ i $\varphi(n)$ coprimers entre si. La clau privada $d$ serà l'invers de $e$ mòdul $\varphi(n)$.

- *Encriptació*: $c \equiv M^e \pmod{n}$
- *Desencriptació*: $M \equiv c^d \pmod{n}$

Amb la funció $V_k(P, Q)$ definida en els apartats anteriors, es pot desenvolupar, tal com es detalla en [3], [9] i [10] un sistema criptogràfic de clau pública diferent a l'RSA, però semblant alhora, que es denota com a LUC.

Siguin $n = pq$ amb p i q primers senars, $\Delta$ un residu no quadràtic i $Q = 1$, el missatge M que es vol enviar serà del conjunt $\Lambda'_n$ definit anteriorment. Com a clau pública s'haurà de prendre un enter $e$ relativament primer amb $(p^2-1)(q^2-1)$ i gràcies a la proposició (4), $V_e(M)$ serà la funció unidireccional de l'espai $\Lambda'_n$. Per triar la clau privada, repetirem el raonament de la demostració de la proposició (4), de manera que $d$ serà l'invers de $e$ mòdul l'ordre del grup $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$, és a dir, $d$ tal que $de \equiv 1 \ (\varphi_\Delta(n))$.

- *Encriptació*: $c \equiv LUC_e(M) \equiv V_e(M) \pmod{n}$.
- *Desencriptació*: $M \equiv LUC_d(c) \equiv V_d(c) \pmod{n}$.

Usant l'observació 8, l'equació (2) de la demostració de la proposició (4) i combinant la part pública i la privada veiem que el sistema funciona correctament.

$$LUC_d(LUC_e(M)) = V_d(V_e(M)) \equiv V_{de}(M) \equiv V_1(M) \equiv M \pmod{n}$$

**Observació 17.** *D'aquest sistema cal remarcar que a l'hora de triar el missatge que volem enviar ens trobem que cada cop s'ha de recalcular la clau privada. Com bé sabem, d es calcula imposant que* $de \equiv 1 \ (\varphi_\Delta(n))$ *i* $\varphi_\Delta(n) = (p - \left(\frac{\Delta}{p}\right))(q - \left(\frac{\Delta}{q}\right))$ *on* $\Delta = M^2 - 4$*, per tant* $\varphi_\Delta(n)$ *depèn de M. Encara que també cal dir a favor del sistema que en realitat, per cada missatge només s'ha de triar una d entre quatre possibles ja que* $\varphi_\Delta(n)$ *només pot assolir els valors* $(p-1)(q-1)$*,* $(p-1)(q+1)$*,* $(p+1)(q+1)$ *i* $(p+1)(q-1)$*.*

A l'article [9] es presenta una proposta per evitar el problema explicat en l'observació (17). Amb totes les condicions anteriors, a l'hora de calcular la clau privada, s'imposa que $d$ cumpleixi $de \equiv 1 \pmod{(p^2-1)(q^2-1)}$. Per veure que el sistema segueix funcionant correctament només cal comprovar que $V_{de}(M) \equiv M \pmod{n}$ amb les noves claus que s'han triat. Per això usarem les relacions (4.2) i (4.9) i el corol·lari (3)

$$\begin{aligned}
V_{de}(M) &= V_{m(p^2-1)(q^2-1)+1}(M) \\
&= MV_{m(p^2-1)(q^2-1)}(M) - V_{m(p^2-1)(q^2-1)-1}(M) \\
&= MV_{m(p^2-1)(q^2-1)}(M) - \frac{1}{2}(MV_{m(p^2-1)(q^2-1)}(M) - \Delta U_{m(p^2-1)(q^2-1)}(M)U_1(M)) \\
&\equiv (2M - \frac{1}{2}(2M - 0)) \\
&= M \pmod{n}
\end{aligned}$$

És important comentar que usant qualsevol de les dues claus privades que hem vist, la funció segueix sent la mateixa ja que en ambdós casos la inversa de $V_e(M)$ és calculada. I a la pràctica, segons [9] usar la primera manera resulta ser més la eficient tot i haver de calcular en cada missatge escollit, quina $d$ agafem.

A primera vista, el LUC sembla criptogràficament més segur que l'RSA degut a la seva definició i a alguna de les seves propietats tal com es comenta a [3] i [9]. L'avantatge en la definició és que les seqüències de Lucas són seqüències recursives d'ordre 2, en canvi, l'exponenciació es pot entendre com una recursió d'ordre 1. Gràcies a la relació (4.12), es pot afirmar que $U_d(P)$ es pot computar sempre que es conegui $V_d(d)$. Llavors, per la següent congruència

$$2P^k \equiv V_k(P + P^{-1}) + (P - P{-}1)U_k(P + P^{-1}) \pmod{n},$$

fàcilment provable per inducció, un criptograma $c$ de l'RSA pot ser desxifrat, si el criptograma $c+c^{-1} \pmod n$ del LUC també es pot desxifrar. Però això, com bé va comentar D. Bleichenbacher a [9] no significa que el LUC sigui més fort que l'RSA, ja que es pot donar el cas en el que només hi hagi un particular conjunt de criptogrames que es puguin desxifrar amb LUC i que alhora, l'RSA sigui segur per qualsevol missatge. Per exemple, suposem que del LUC es poden desencriptar tots els criptogrames $c$ tals que $\left(\frac{c^2-4}{p}\right) = \left(\frac{c^2-4}{q}\right) = -1$, que són el 25% dels criptogrames de LUC. Però com que, $\left(\frac{(c+c^{-1})^2-4}{p}\right) = \left(\frac{(c-c^{-1})^2}{p}\right) = 1$, per desencriptar qualsevol criptograma del RSA no en serveix cap dels de LUC que sí que podem desencriptar. Per tant, l'RSA és totalment segur a diferència del LUC, del qual se'n poden desencriptar el 25% dels criptogrames.

La seguretat de l'RSA es basa en el problema (3), en canvi, la seguretat del LUC es basa en el problema (4). D'aquí en podem concloure que aquests dos esquemes criptogràfics són polinòmicament equivalents en quant a seguretat ja que com s'ha comentat anteriorment, els problemes (3) i (4) ho demostren.

### Versió de ElGamal

Aquest criptosistema probabilístic dissenyat per T. ElGamal l'any 1985 basa la seva seguretat en el problema del logaritme discret a $\mathbb{Z}_p$, donat un $p$ bastant gran.

L'usuari a qui es vol enviar el missatge, escull un $p$ primer gran i un generador $g$ del grup multiplicatiu $\mathbb{Z}_p^*$. Llavors tria un $0 \leq x < p-1$ que serà la clau privada i publica $y \equiv g^x \pmod p$.

- *Encriptació*: El que envia el missatge M, tria aleatòriament $0 \leq k < p-1$ i es calcula $c_1 \equiv g^k \pmod p$ i $c_2 \equiv y^k M \pmod p$. $(c_1, c_2)$ és el missatge encriptat.
- *Desencriptació*: Primer es calcula $s \equiv c_1^x \pmod p$ per acabar trobant el missatge original $c_2 s^{-1} \equiv y^k M s^{-1} \equiv g^{xk} M g^{-xk} \equiv M \pmod p$.

La versió d'aquest esquema amb funcions de Lucas descrita a [4] és pràcticament igual, però usant $V_k$ enlloc de l'exponenciació a $\mathbb{Z}_p^*$.

Com abans, el receptor tria un primer $p$ prou gran, i en aquest cas un $P \in \mathbb{Z}_p$ tal que $\left(\frac{P^2-4}{p}\right) = -1$ i que 1 i $p+1$ siguin els únics divisors $k$ de $p+1$ tals que $V_k(P) \equiv 2 \mod p$. També tria un $0 \leq x < p-1$ que serà la clau privada i publica $y \equiv V_x(P) \pmod p$.

- *Encriptació*: Per enviar el missatge M, es tria aleatòriament $0 \leq k < p-1$ i es calcula $c_1 \equiv V_k(P) \pmod p$ i $c_2 \equiv V_k(y) M \pmod p$. $(c_1, c_2)$ és el missatge encriptat.
- *Desencriptació*: Primer es calcula $s \equiv V_x(c_1) \pmod p$ per acabar trobant el missatge original $c_2 s^{-1} \equiv V_k(y) M s^{-1} \equiv V_{xk} M V_{xk}^{-1} \equiv M \pmod p$.

### Criptosistema de G. Castagnos sobre quocients de cossos quadràtics

Aquest tipus de criptosistemes probabilístics es van començar a descriure a partir de l'esquema de Paillier del 1999 (veure [27]). Dos anys més tard, Catalano, Gennaro et al. van proposar a [26] una variant més ràpida, que més endavant va ser adaptada sobre el grup de punts d'una corba elíptica per Galindo et al. (veure [28]).

Ara descriurem breument el criptosistema de Catalano i Gennaro. Prenem $n = pq$ un enter amb p i q primer grans i $e$ un enter coprimer amb $\varphi(n) = (p-1)(q-1)$, per així tenir el parell $(n, e)$ com clau pública. Definim un conjunt i una funció que necessitem per construir el criptosistema.

**Definició 6.**

$$R_n = \{x \in \mathbb{N},\ 0 < x < n,\ mcd(x, n) = 1\}.$$

**Definició 7.**

$$\varepsilon_e : \begin{cases} \mathbb{Z}/n\mathbb{Z} \times R_n & \to & (\mathbb{Z}/n^2\mathbb{Z})^* \\ (M, r) & \mapsto & (1+Mn)r^e \end{cases}$$

Sabent que $r^e$ és una permutació del conjunt $R_n$ és senzill demostrar que la funció $\varepsilon_e$ és bijectiva. Per això és una bona candidata a funció unidireccional per construir un sistema criptogràfic.

- *Encriptació*: Si volem enviar el missatge M, es tria $r \in R_n$ i s'envia $c \equiv \varepsilon_e(M, r) \pmod{n^2}$
- *Desencriptació*: El receptor, que té la clau privada $d$ (l'invers de $e$ mòdul $\varphi(n)$), primer redueix $c \in (\mathbb{Z}/n^2\mathbb{Z})^*$ mòdul n i fa una desencriptació com en l'RSA per aconseguir $r$. Llavors calcula $c/r^e \pmod{n^2}$ per aconseguir $1 + Mn$ i finalment aïlla M.

**Observació 18.** *Es pot observar que si hom pot desencriptar l'RSA, llavors també pot desencriptar aquest criptosistema. De fet, està provat que invertir la funció RSA és polinòmicament equivalent a invertir la funció $\varepsilon_e$. També s'ha provat que aquest criptosistema és semànticament segur si i només si donat un element arbitrari $c \in (\mathbb{Z}/n^2\mathbb{Z})^*$, és difícil dir si existeix un element $r \in R_n$ tal que $c \equiv r^e \pmod{n^2}$.*

Per adaptar aquest criptosistema que acabem de veure al grup $(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta)^\wedge$ que hem estudiat abans usant les funcions de Lucas, també necessitem definir un conjunt i una funció adients. D'aquesta manera obtindrem el nou criptosistema descrit a [10] i partirem de les mateixes condicions que en el cas anterior prenent (n,e) com a les claus públiques.

**Definició 8.**

$$\Omega_n = \left\{x \in \mathbb{N},\ 0 < x < n^2,\ mcd(x^2 - 4, n) = 1,\ mcd(x, n) = 1\right\}.$$

**Definició 9.**

$$\varepsilon'_e : \begin{cases} \mathbb{Z}/n\mathbb{Z} \times \Lambda'_n & \to & \Omega_n \\ (M, r) & \mapsto & (1+n)^M V_e(r) \pmod{n^2} \end{cases}$$

Com abans, necessitem que $\varepsilon'_e$ estigui ben definida i sigui bijectiva, per això ho mostra el següent teorema.

**Teorema 4.** *Segons la notació de les definicions anteriors, la funció $\varepsilon'_e$ està ben definida i és bijectiva.*

**Dem.** Primer de tot cal comentar que $(1+n)^M \equiv 1 + Mn \pmod{n^2}$. Per veure que està ben definida, s'ha de mostrar que $\varepsilon'_e(M, r) \in \Omega_n$. Per això, si prenem mòdul n obtenim que $\varepsilon'_e(M, r) \pmod{n} \equiv V_e(r)$, i com que $r \in \Lambda'_n$, per la proposició (4) $\varepsilon'_e(M, r) \pmod{n} \in \Lambda'_n$. D'aquí ja és senzill veure el que volíem.
Per veure que és bijectiva, només cal veure que la funció és injectiva ja que $\mathbb{Z}/n\mathbb{Z} \times \Lambda'_n$ i $\Omega_n$ tenen el mateix nombre d'elements. Per tant, prenem $(M_1, r_1)$ i $(M_2, r_2)$ tal que les imatges siguin iguals. Si prenem mòdul $n$, com que $V_e(r)$ és una permutació de $\Lambda'_n$, $r_1 = r_2$. Llavors, com que $V_e(r_1)$ és primer amb $n$, $M_1 = M_2$. □

- *Encriptació*: Aquest procés és exactament igual que amb l'esquema d'abans, però usant la funció $\varepsilon_e(M, r)$.
- *Desencriptació*: Com abans, es redueix $c \pmod n$ i com que $r$ és menor que n, es pot recuperar fent una desencriptació com en el LUC usant $d$ (l'invers de $e$ mòdul $(p - \left(\frac{\Delta}{p}\right))(q - \left(\frac{\Delta}{q}\right))$). Un cop trobat $r$, s'aïlla $M$ com abans.

**Observació 19.** *Amb aquest algorisme de desencriptació i amb el de Catalano, es pot usar el Teorema dels residus xinesos per incrementar-ne la velocitat. El cas de l'algorisme del G. Castagnos el podem trobar detallat en l'apartat 6 de l'article [10].*

**Observació 20.** *És clar que si es pot desencriptar el LUC, llavors també es pot desencriptar aquest esquema de G. Castagnos, però a diferència de l'esquema de Catalano et al. i l'RSA, no és sabut si aquests dos esquemes són equivalents. Fins ara es creu que la millor manera d'invertir $\varepsilon'_e$ és factoritzar $n$. A [10] presenten un teorema que afirma que la seguritat semàntica de l'esquema del G.Castagnos recau en la dificultat del següent problema decisional: Sigui $n = pq$ i $e$ coprimer amb $(p^2 - 1)(q^2 - 1)$, donat un element $c$ del conjunt $\Omega_n$, trobar si existeix $r \in \Lambda'_n$ tal que $c = V_e(r) \pmod{n^2}$.*

### 4.6.3 Esquemes de signatura digital

Els esquemes de signatura digital són criptosistemes de clau pública que consisteixen en tres algorismes clarament diferenciats.

- *Generació de claus*: L'algoritme de generació de claus, amb l'entrada $1^k$ (sent $k$ el paràmetre de seguretat), produeix un parell $(p_k, s_k)$ que són la clau pública i la clau privada respectivament.
- *Signatura*: L'algoritme de signatura, que té com entrada la clau privada $s_k$ i el missatge $M$ que es vol signar, genera una signatura $s$ per $M$.
- *Verificació*: L'algorisme de verificació, d'entrada la clau pública $p_k$, el missatge $M$ i la signatura $s$, comprova si $s$ és una signatura vàlida per $M$.

Hi ha diferents tipus d'atacs a aquests esquemes de signatures digitals, que es poden diferenciar entre:

- *Key-only attack.* Només se sap la clau pública.
- *Known-signature attack.* L'atacant només coneix parelles de missatges i signatures corresponents triades per l'usuari que signa.
- *Chosen-message attack.* L'atacant pot fer servir a l'usuari que signa com un oracle a una llista de missatges i només una vegada.
- A*daptively-chosen-message attack.* A diferència de l'anterior, l'atacant pot fer servir l'usuari que signa com un oracle tantes vegades com vulgui.

Els nivells d'èxit de l'atac poden ser descrits en ordre creixent com:

- *Existential forgery.* L'atacant pot signar almenys un missatge, no necessàriament un que esculli ell.
- *Selective forgery.* L'atacant pot signar uns quants missatges que tria.
- *Universal forgery.* L'atacant signa els missatges que es proposa.
- *Retrieval of secret keys.* L'atacant descobreix la clau secreta.

**Derivats de ElGamal i RSA**

Des de 1993, la NSA (*National Security Agency*) va adoptar el DSA (*Digital Signature Algorithm*) com signatura digital estàndard, que es basa en l'esquema de T. ElGamal. Amb les funcions de Lucas també s'han dissenyat signatures basades en aquest esquema. Els paràmetres d'aquest són un $p$ primer suficientment gran, un generador $g$ del grup multiplicatiu $\mathbb{Z}_p^*$ i una funció hash $H$. A continuació veiem l'esquema de signatura d'ElGamal.

- *Generació de claus*: El signant sel·lecciona $x$ tal que $0 < x < p-1$ que serà la clau privada i calcula $y \equiv g^x \pmod{p}$, la clau pública.
- *Signatura*: La signatura d'un missatge $M$ serà un parell $(r, s)$ que es calcularà a partir d'un nombre aleatori $k$ tal que $0 < k < p-1$ i $mcd(k, p-1) = 1$. Primer calculem $r \equiv g^k \pmod{p}$ i posteriorment $s \equiv (H(M) - xr)k^{-1} \pmod{p-1}$. Cal comentar que si $s = 0$, es tria una nova $k$ i es torna a calcular $r$ i $s$.
- *Verificació*: Només cal comprovar que $g^{H(M)} \equiv y^r r^s \pmod{p}$.

Com hem dit, hi ha vàries signatures que usen les funcions de Lucas $V_k$ i $U_k$ i les seves propietats. La primera, denotada com LUCELG DS, va ser dissenyada per P.Smith i C.Skinner a [3], i és molt semblant a la que hem descrit. Posteriorment, a [8], es va veure que aquesta podia ser atacada fàcilment. Els paràmetres de LUCELG DS són una funció hash $H$, un primer $p$ prou gran tal que $p+1$ no estigui composat per primers petits i un generador $P \in \mathbb{Z}_p$ tal que $\left(\frac{P^2-4}{p}\right) = -1$ i que 1 i $p+1$ siguin els únics divisors $k$ de $p+1$ tals que $V_k(P) \equiv 2 \mod p$.

- *Generació de claus*: A diferència d'abans, el signant calcula dues claus públiques a partir de la clau privada $x$ que prèviament a escollit. Seran $y_V \equiv V_x(P) \pmod{p}$ i $y_U \equiv U_x(P) \pmod{p}$.
- *Signatura*: Per signar el missatge en aquest cas, a partir d'un nombre aleatori $k \in \mathbb{Z}_{p+1}^*$, es calcula $r_V \equiv V_k(P) \pmod{p}$, $r_U \equiv U_k(P) \pmod{p}$ i $s \equiv (H(M) - xr)k^{-1} \pmod{p+1}$. La signatura serà la tupla $r_V, r_U, s$.
- *Verificació*: Cal veure que gràcies a la relació (4.8), es compleix

$$2V_{H(M)}(P) \equiv V_{r_V}(y_V)V_s(r_V) + \Delta y_U U_{r_V}(y_V) r_U U_s(r_V) \pmod{p}$$

Desafortunadament, aquesta signatura no és vàlida ja que pot ser *universally forged* amb un *key-only attack*. Com podem veure, $r_U$ no està lligada amb $r_V$ i $s$ durant el procés de la signatura i per això, independentment de quines $r_V$ i $s$ es trïin, qualsevol persona pot signar el missatge M usant

$$r_U \equiv (2V_{H(M)}(P) - V_{r_V}(y_V)V_s(r_V))(\Delta y_U U_{r_V}(y_V)U_s(r_V))^{-1} \pmod{p}$$

Una altra signatura amb l'esquema de ElGamal és la que proposen a [8], on aparentment es sol·luciona el problema que hi havia en la LUCELG DS, ja que només usa la funció $V_k$ i una clau pública. Els paràmetres són els mateixos que abans amb la diferència que $\left(\frac{P^2-4}{p}\right) = 1$ i per això, quan treballem amb els subíndex de $V_k$, enlloc de mòdul $p+1$, usarem mòdul $p-1$ degut al teorema (3). Els algorismes de la signatura són molt semblants als de les anteriors, tot i que en aquest cas s'usarà la relació (4.16) i el següent lema.

**Lema 8.** *Amb les condicions anteriors i les claus que es defineixen en els següents algorismes,*

$$2V_{H(M)}(P) \equiv V_r(y)V_s(r) \pm \sqrt{V_r^2(y)-4}\sqrt{V_s^2(r)-4} \pmod{p}.$$

- *Generació de claus*: Primer es tria la clau privada $x \in \mathbb{Z}_p$ i llavors es calcula la clau pública $y \equiv V_x(P) \pmod{p}$.
- *Signatura*: La signatura serà la tupla $(r, s)$. S'escull un nombre aleatori $k \in \mathbb{Z}_{p-1}^*$, per calcular $r \equiv V_k(P) \pmod{p}$ i $s \equiv (H(M) - xr)k^{-1} \pmod{p-1}$.
- *Verificació*: El receptor només haurà de comprovar que es satisfà la següent congrüencia.

$$V_{H(M)}^2(P) + V_s^2(r) + V_r^2(y) \equiv V_{H(M)}(P)V_r(y)V_s(r) + 4 \pmod{p}.$$

**Observació 21.** *La seguretat d'aquesta signatura digital, com es veu en l'observació (9), es basa en el logaritme discret sobre $GF(p)$ ja que s'imposa la condició $\left(\frac{P^2-4}{p}\right) = 1$.*

A [8], P. Horster et al. demostren que aquesta signatura digital és *universally forged* si i només la signatura d'ElGamal descrita anteriorment és *universally forged*. A part, també mostren que un atacant pot assolir una *existential forgery* amb un *key-only attack*. L'atacant primer escull $a \in \mathbb{Z}_{p-1}$ i $b \in \mathbb{Z}_{p-1}^*$ aleatòriament i computa

$$r \equiv V_k(P) \equiv 2^{-1}\left(V_a(P)V_b(y) \pm \sqrt{(V_a^2(P)-4)(V_b^2(y)-4)}\right) \pmod{p}.$$

Per una altra banda, se sap que

$$\begin{aligned} r &\equiv V_k(P) \equiv V_{Ms^{-1}-xrs^{-1}}(P) \equiv \\ &\equiv 2^{-1}\left(V_{Ms^{-1}}(P)V_{-rs^{-1}}(y) \pm \sqrt{(V_{Ms^{-1}}^2(P)-4)(V_{-rs^{-1}}^2(y)-4)}\right) \pmod{p}, \end{aligned}$$

d'on obtenim que $a \equiv Ms^{-1} \pmod{p-1}$ i $b \equiv -rs^{-1} \pmod{p-1}$. Per tant, l'atacant ja té $r$ i pot calcular fàcilment $s$ i $M$ aïllant perquè la signatura sigui vàlida. Cal comentar que fent servir una funció de Hash es pot prevenir aquest atac.

Fins ara ens hem basat en l'esquema de la signatura de T. ElGamal, però hi ha d'altres signatures interessants les quals també es poden aplicar a les funcions de Lucas com les signatures RSA, que van ser les primeres en ser dissenyades en el món de la signatura digital.

Els paràmetres d'aquestes són una funció hash $H$ i un enter n prou gran de la forma $n = pq$.

- *Generació de claus*: L'usuari que vol signar el missatge, generarà les mateixes claus que en el sistema RSA abans explicat. La privada serà $d$ i la pública $e$.
- *Signatura*: Per signar es farà el procés invers que en l'RSA amb la imatge hash del missatge. La signatura serà $H(M)^d \pmod{n}$.

- *Verificació*: El receptor usarà la clau pública $e$ i verificarà que $H(M) \equiv (H(M)^d)^e \pmod{n}$.

Aquesta signatura pot ser *universally forged* amb un *chosen-message attack* degut a la propietat multiplicativa de l'exponenciació.

De la mateixa manera, usant la funció del sistema criptogràfic LUC amb les mateixes condicions i paràmetres, es pot signar un missatge M com $V_d(H(M)) \pmod{n}$ i verificar-ho comprovant que $V_e(V_d(H(M))) \equiv H(M) \pmod{n}$.

Com que les funcions de Lucas no tenen la propietat multiplicativa, els autors de [3] creien que aquest esquema de Lucas podia evitar un chosen-message attack. Però malauradament, D. Bleichenbacher et al. van trobar un chosen-message attack contra la signatura amb funcions de Lucas. En aquest atac primer es seleccionen $a$, $b$, $c$, $s$ i $t$ tal que $bs - ct = 1$ i $bs + ct = ae$. Llavors es calcula $M_s \equiv V_s(M) \pmod{n}$ i $M_t \equiv V_t(M) \pmod{n}$ i es demana a l'oracle que signi els missatges $V_d(M_s \pmod{n})$ i $V_d(M_t \pmod{n})$. Finalment, $V_d(M)$ es calcula com

$$V_d(M) \equiv V_b(V_d(M_s))V_c(V_d(M_t)) - V_a(M) \pmod{n}.$$

La correctesa d'aquest atac es demostra a partir de les equacions (4.11) i (4.13).

$$\begin{aligned} V_b(V_d(M_s))V_c(V_d(M_t)) &\equiv V_{dbs}(M)V_{dct}(M) \\ &\equiv V_{d(bs+ct)}(M) + V_{d(bs-ct)}(M) \\ &\equiv V_{dae}(M) + V_d(M) \\ &\equiv V_a(M) + V_d(M) \pmod{n}. \end{aligned}$$

**Aplicació de les signatures digitals a canals subliminals**

Un canal *covert* o *paràsit* és un canal de comunicació que es fa servir sense l'autorització del propietari de tal canal de manera que no es pugui detectar el seu ús per cap entitat que no estigui involucrada en la comunicació.

Un canal *subliminal* es un canal de comunicació *covert* usat per enviar un missatge a un receptor autoritzat tal que aquest missatge no pugui ser descobert per cap receptor no autoritzat. Aquests canals subliminals es van començar a usar quan, en el 1984, G. Simmons se'n va adonar de que les firmes digitals tipus ElGamal podien ser usades per establir un canal subliminal.

Per exemple, en un passaport, la signatura digital és usada per autentificar al seu propietari davant un agent. A més, el missatge en el canal subliminal pot dir-li a l'agent que aquest propietari té antecedents, o qualsevol altre informació rellevant. Pot haver-hi moltes més aplicacions dels canals subliminals en la vida quotidiana com l'historial d'un conductor amb el seu carnet de conduir o els pagaments que ha efectuat un client d'un banc amb una targeta de crèdit.

Els canals subliminals poden ser classificats en canals *broadband* i *narrowband*. Els broadband són els que usen quasi tots els bits disponibles i els narrowbans els que n'usen pocs.

A l'hora de dissenyar aquests tipus d'esquemes, l'objectiu és aconseguir que l'usuari que signa no hagi de compartir la seva clau secreta amb els receptors i poder usar canals *broadband*. G.Simmons va començar descrivint canals *narrowband* que no requerien compartir la clau secreta, fins que va aconseguir trobar un canal *broadband*, però que tenia el problema d'haver de compartir la clau secreta. A [16], L. Harn et al. van dissenyar un esquema de signatura amb canals subliminal broadband que no requerien compartir la clau secreta de la signatura (que el veurem a continuació). A més a més, també van presentar un esquema semblant basat en les funcions de Lucas on s'aconseguia una major eficiència.

**Notació 1.** *Siguin $p$ i $q$ enters primers senars i $y_p$ i $y_q$ enters, la sol·lució de les equacions*

$$\begin{aligned} y &\equiv y_p \pmod{p} \\ y &\equiv y_q \pmod{q} \end{aligned}$$

*la denotem com a $y = CRT(y_p, y_q; p, q)$.*

Els paràmetres que primer s'escullen en l'esquema de [16] són $p$, $q$ primers senars, $n = pq$, $p' = \frac{p-1}{2}$, $q' = \frac{q-1}{2}$ i $\alpha \in [1, n-1]$ d'ordre $(p-1)(q-1)$. En aquesta signatura hi haurà dos canals subliminals, un que usarà el primer $p$ i l'altre que usarà $q$, però per la signatura treballarem amb mòdul n. Degut al disseny d'aquest esquema, ha d'haver-hi óbligatòriament dos canals, però això no significa que hi hagi dos receptors. Lo important és que els dos canals són disjunts i el receptor només conegui la clau secreta d'un dels canals. Si només hi ha un receptor, l'usuari que signa serà el coneixedor de l'altre clau secreta.

- *Claus privades*: Cada receptor té la seva clau privada $x_p \in [1, p-1]$ i $x_q \in [1, q-1]$, que únicament la coneix ell mateix i l'usuari que signa. Llavors, aquest calcula la seva clau privada $x = CRT(x_p, x_q, 2p', 2q')$ agafant la més petita de les sol·lucions i no la comparteix amb ningú.
- *Clau pública*: Hi haurà una clau pública que es calcularà a partir de $y_p \equiv \alpha^{x_p} \pmod{p}$ i $y_q \equiv \alpha^{x_q} \pmod{q}$. Aquesta serà $y = CRT(y_p, y_q, p, q)$.
- *Signatura*: Per signar $M$, i enviar subliminalment $M_p$ pel canal $p$ i $M_q$ pel canal $q$, primer es calcula $r_p \equiv \alpha^{M_p} \pmod{p}$ i $r_q \equiv \alpha^{M_q} \pmod{q}$ per obtenir $r = CRT(r_p, r_q; p, q)$. En segon lloc, necessitem $M_{pq} = CRT(M_p, M_q; 2p', 2q')$, agafant la sol·lució més petita, per obtenir $s \equiv rMx - M_{pq} \pmod{4p'q'}$. La signatura és (r,s).
- *Verificació*: Només cal comprovar que $y^{rM} \equiv r\alpha^s \pmod{n}$
- *Recuperació dels missatges subliminals*: El receptor que coneix $p$ i la clau secreta $x_p$, calcula $M_p \equiv rMx_p - s_p \mod 2p'$ on $s_p \equiv s \pmod{2p'}$. L'altre receptor fa el mateix amb la seva clau secreta $(q, x_q)$.

**Observació 22.** *Per a qualsevol adversari extern, la seguretat de l'esquema recau en dos problemes, factorització (problema 3) i logaritme discret (problema 1). Per això, aquesta signatura és més segura que molts d'altres esquemes coneguts. Però cal comentar que en quant als receptors autoritzats la seguretat només recau en un dels dos problemes, el logaritme discret, degut a que ja coneixen la factorització de n. Tot i això a [16] L.Harn et al. detallen una proposta per a sol·lucionar aquest inconvenient.*

**Observació 23.** *Com hem comentat en l'observació anterior, la seguretat de l'esquema recau en els dos problemes, i per això, necessitem que la p i q siguin suficientment grans perquè el logaritme discret mantingui el mínim nivell de seguretat necessari. En conseqüència, n = pq serà massa llarga en quant a computació.*

Per sol·lucionar el problema de l'observació (23), com hem comentat anteriorment, els autors de [16] van presentar un nou esquema basat en les funcions de Lucas, que basa la seva seguretat en la factorització (problema 4) i el logaritme discret a $\mathbb{Z}_{p^2}$ enlloc de $\mathbb{Z}_p$ (problema 2). Per tant, gràcies a que resoldre el problema (2) és més costós, permet agafar els paràmetres $p$ i $q$ de menys longitud i conseqüentment obtenir una $n$ de longitud binària més curta.

Els paràmetres de l'esquema amb les funcions de Lucas són $p$, $q$ primers senars, $n = pq$, $p' = \frac{p+1}{2}$, $q' = \frac{q+1}{2}$ i $P \in \mathbb{Z}$ tal que el polinomi $f(x) = x^2 - Px + 1$ sigui irreductible sobre $\mathbb{Z}_p$ i $\mathbb{Z}_q$.

- *Claus privades*: $x_p$ i $x_q$ són exactament les mateixes que en l'esquema anterior i $x = CRT(x_p, x_q; p+1, q+1)$.
- *Clau pública*: La clau $y = CRT(y_p, y_q, p, q)$ es calcula a partir de $y_p \equiv V_{x_p}(P) \pmod{p}$ i $y_q \equiv V_{x_q}(P) \pmod{q}$.
- *Signatura*: Per signar $M$, i enviar subliminalment $M_p$ pel canal $p$ i $M_q$ pel canal $q$, primer es calcula $r_p \equiv V_{M_p}(P) \pmod{p}$ i $r_q \equiv V_{M_q}(P) \pmod{q}$ per obtenir $r = CRT(r_p, r_q; p+1, q+1)$. En segon lloc, necessitem $M_{pq} = CRT(M_p, M_q; p+1, q+1)$, agafant la sol·lució més petita, per obtenir $s \equiv rMx - M_{pq} \pmod{(p+1)(q+1)}$. La signatura és (r,s).
- *Verificació*: Només cal comprovar que

$$r^2 + V_{Mr}^2(y) + V_s^2(P) \equiv rV_{Mr}(y)V_s(P) + 4 \pmod{n}.$$

- *Recuperació dels missatges subliminals*: El receptor que coneix $p$ i la clau secreta $x_p$, calcula $M_p \equiv rMx_p - s_p \mod p+1$ on $s_p \equiv s \pmod{p+1}$. L'altre receptor fa el mateix amb la seva clau secreta $(q, x_q)$.

# Capítol 5

# Sistema de clau pública XTR

Els elements de les funcions de Lucas estudiades anteriorment pertanyen a un subgrup de $p+1$ elements de $GF(p^2)$, però hem vist a l'equació (4.5) que usant la traça d'aquests s'aconsegueix representar-los amb elements de $GF(p)$ sense perdre la seguretat que garanteix el logaritme discret a $GF(p^2)$.

El sistema XTR és un nou mètode que es basa en la mateixa idea que les funcions de Lucas, representant elements de $GF(p^6)$ sobre $GF(p^2)$ usant la traça $Tr_{GF(p^6)/GF(p^2)}$. En conseqüència, el nom XTR és una abreviació de *Efficient and Compact Subgroup Trace Representation*.

Aquest nou mètode té noves propietats que veurem a continuació, i que proporcionen interessants aplicacions per la criptografia.

## 5.1 Mètode XTR

Siguin $p$ i $q$ enters primers tals que $p \equiv 2 \pmod 3$, $q > 3$ i $q$ divideix $p^2-p+1$, considerem el cos finit $GF(p^6)$. Dintre del grup multiplicatiu de $GF(p^6)$, que denotem com $GF(p^6)^*$, prenem un element $g$ d'orgre $q$.

El subgrup d'ordre $p^2-p+1$ de $GF(p^6)^*$ es coneix com al *supergrup XTR*, en canvi, el subgrup de $q$ elements generat per $g$ és el *subgrup XTR*. El fet d'agafar aquest *supergrup XTR* és degut a que no està contingut en cap subgrup propi de $GF(p^6)$ i d'aquesta manera, la computació del logaritme discret és tan complicada com en $GF(p^6)^*$. Però aquesta elecció no és important només per raons de seguretat, sinó que també permet una representació eficient dels elements del *supergrup XTR*, i per tant també dels del *subgrup XTR*, que consisteix en expressar-los amb elements de $GF(p^2)$ enlloc de $GF(p^6)$.

Com que es treballarà amb elements de $GF(p^2)$, és necessari que les operacions aritmètiques en aquest cos finit siguin eficients. Si prenem el polinomi irreductible $(x^3-1)/(x-1) = x^2+x+1$, les seves arrels $\alpha$ i $\alpha^p$ formen una base de $GF(p^2)$ sobre $GF(p)$, és a dir,

$$GF(p^2) \cong \{x_1\alpha + x_2\alpha^p : x_1, x_2 \in GF(p)\}.$$

Degut a que $p \equiv 2 \pmod 3$ i que $\alpha^3 = 1$, es compleix que $\alpha^i = \alpha^{i \bmod 3}$ i $\alpha^p = \alpha^2$, per tant podem expressar $GF(p^2)$ de la següent forma.

$$GF(p^2) \cong \left\{x_1\alpha + x_2\alpha^2 \mid \alpha^2+\alpha+1 = 0 \text{ i } x_1, x_2 \in GF(p)\right\}.$$

D'aquesta forma, es pot calcular el cost de les operacions a $GF(p^2)$ en funció de multiplicacions a GF(p).

**Lema 9.** *Sigui $x, y, z \in GF(p^2)$ amb $p \equiv 2 \pmod 3$,*

1. *computar $x^p$ no costa res,*
2. *per computar $x^2$ s'han de fer dues multiplicacions a $GF(p)$,*
3. *per computar $xy$ calen tres multiplicacions a $GF(p)$,*
4. *per computar $xz - yz^p$ són necessàries quatre multiplicacions a GF(p).*

**Dem.** Siguin $x, y, z \in GF(p^2)$ tals que $x = x_1\alpha + x_2\alpha^2$, $y = y_1\alpha + y_2\alpha^2$ i $z = z_1\alpha + z_2\alpha^2$, i tenint en compte que un element $t \in GF(p)$ es pot representar de la forma $t = -t\alpha - t\alpha^2$.

1. $x^p = x_2\alpha + x_1\alpha^2$.
2. $x^2 = x_2(x_2 - 2x_1)\alpha + x_1(x_1 - 2x_2)\alpha^2$
3. $xy = (x_2y_2 - x_1y_2 - x_2y_1)\alpha + (x_1y_1 - x_1y_2 - x_2y_1)\alpha^2$. S'han de computar les parelles $x_1y_1$, $x_2y_2$ i $(x_1 + x_2)(y_1 + y_2)$.
4. Operant resulta

$$\begin{aligned} xz - yz^p &= (z_1(y_1 - x_2 - y_2) + z_2(x_2 - x_1 + y_2))\alpha + \\ &\quad ((z_1(x_1 - x_2 + y_1) + z_2(y_2 - x_1 - y_1)))\alpha^2. \end{aligned}$$

□

M. Stam et al. [21] van proposar una lleugera millora en l'aritmètica presentada en el lema anterior. La multiplicació a $GF(p)$ consta de dos passos amb costos significatius, el pas de la multiplicació i el pas de la reducció. Per exemple, per multiplicar $3 \cdot 4$ a $GF(5)$, el pas de la multiplicació és $3 \cdot 4 = 12$ i el pas de la reducció és $12 \mod 5 = 2$. L'observació clau és que el pas de la reducció és més costós que el de la multiplicació, i per això, en els casos (3) i (4) del lema anterior ens podem estalviar algun pas de reducció fent primer totes les operacions i no reduir fins al final, aconseguint així els següents costos.

**Lema 10.** *Sigui $x, y, z \in GF(p^2)$ amb $p \equiv 2 \pmod 3$,*

1. *computar $x^p$ no costa res,*
2. *per computar $x^2$ s'han de fer dues multiplicacions a $GF(p)$,*
3. *per computar $xy$ calen dues multiplicacions i mitja a $GF(p)$,*
4. *per computar $xz - yz^p$ són necessàries tres multiplicacions a GF(p).*

Per representar l'element $g \in GF(p^6)$ amb elements de $GF(p^2)$ es necessiten tenir en compte els seus conjugats $g^{p^2}$ i $g^{p^4}$. Com que $g$ és un element d'ordre $q$ tal que $q \mid p^2 - p + 1$, podem afirmar que $g^{p^2-p+1} = 1$; per tant, $g^{p^2} = g^{p-1}$ i $g^{p^4} = g^{-p}$. Si considerem el polinomi que té $g$ i els seus conjugats com arrels, obtenim un polinomi de tercer grau amb coeficients a $GF(p^2)$ de la següent forma.

$$(x - g)(x - g^{p-1})(x - g^{-p}) = x^3 - Tr(g)x^2 + Tr(g)^p x - 1 \in GF(p^2)[x] \qquad (5.1)$$

Com que aquest polinomi està completament determinat per la $Tr(g)$, podem identificar els elements de $GF(p^6)$ (sense distingir conjugats) amb la seva traça. Aquesta observació també es pot aplicar a qualsevol potència de $g$, és a dir, el polinomi $x^3 - Tr(g^n)x^2 + Tr(g^n)^p x - 1$ i les seves arrels queden determidats per $Tr(g^n)$.

Si hi hagués una forma prou ràpida de calcular $Tr(g^n)$ donat $Tr(g)$, seria la manera de calcular $g^n$ donat $g$ però usant elements de $GF(p^2)$, és a dir, elements amb un tamany menys costós de representar que els de $GF(p^6)$. Aquesta és la idea clau del mètode XTR, en els diferents protocols criptogràfics, usar la traça enlloc de $g$.

## 5.2 Computació de traces

Com hem dit, s'ha de computar $Tr(g^n)$ donat $Tr(g)$, i per això són necessaris una sèrie de conceptes previs.

**Definició 10.** *Sigui $c \in GF(p^2)$ es defineix*

$$F(c, x) = x^3 - cx^2 + c^p x - 1 \in GF(p^2)[x],$$

*i $c_n = h_0^n + h_1^n + h_2^n$ per $n \in \mathbb{Z}$ on $h_0, h_1, h_2 \in GF(p^6)$ són les arrels de $F(c, X)$.*

**Observació 24.** *La definició de $F(c, x)$ per tot $c \in GF(p^2)$ és més general que el polinomi que hem definit a l'equació (5.1), on només es consideraven $c$ de la forma $Tr(g)$ per $g$ d'ordre $> 3$ i que divideixi $p^2 - p + 1$. L'aplicació d'aquesta forma general s'usarà en la selecció de paràmetres a l'hora d'escollir el $g$ adient. Per la computació, si $c = Tr(g)$, llavors es calcula $c_n = Tr(g^n)$ usant les següents propietats.*

**Lema 11.** *1. $c = c_1$.*

2. *$c_{-n} = c_{np} = c_n^p$ per $n \in \mathbb{Z}$.*
3. *$c_n \in GF(p^2)$ per $n \in \mathbb{Z}$.*
4. *$c_{u+v} = c_u c_v - c_v^p c_{u-v} + c_{u-2v}$ per $u, v \in \mathbb{Z}$.*
5. *$F(c_n, h_j^n) = 0$ per $j = 0, 1, 2$ i $n \in \mathbb{Z}$.*
6. *Les arrels $h_j$, o tenen ordre dividint a $p^2 - p + 1$ i major que 3 o pertanyen a $GF(p^2)$. En particular, $F(c, x)$ és irreductible a $GF(p^2)$ si i només si l'ordre de les seves arrels divideix $p^2 - p + 1$ i major que 3.*
7. *$F(c, x)$ és reductible a $GF(p^2)$ si i només si $c_{p+1} \in GF(p)$.*

**Dem.** Aquestes propietats es demostren a partir de la definició de traça i de la definició 10. Es pot trobar detallada en [18].

Gràcies a aquest lema, donats $c, c_{k-1}, c_k$ i $c_{k+1}$, es poden computar $c_{k+2}, c_{2k-1}, c_{2k}$ i $c_{2k+1}$.

- $c_{2k} = c_k^2 - 2c_k^p$.
- $c_{k+2} = c \cdot c_{k+1} - c^p \cdot c_k + c_{k-1}$.
- $c_{2k-1} = c_{k-1} \cdot c_k - c^p \cdot c_k^p + c_{k+1}^p$.

- $c_{2k+1} = c_{k+1} \cdot c_k - c \cdot c_k^p + c_{k-1}^p$.

**Definició 11.** $S_n(c) = (c_{n-1}, c_n, c_{n+1}) \in GF(p^2)^3$.

Com que per calcular $c_n$ iterativament, es necessiten trios consecutius, a la pràctica, l'algorisme que necessitem calcularà $S_n(c)$ donats $c$ i $n$.

La idea d'aquest algorisme (detallat en [18] i [20]) és inicialitzar $S_0$, $S_1$, $S_2$ i $S_3$, i llavors recórrer els bits de $k$ (essent $n = 2k + 1$) fent les següents operacions.

- Si el bit és 0, usa $(c_{2i}, c_{2i+1}, c_{2i+2})$ per calcular $(c_{4i}, c_{4i+1}, c_{4i+2})$.
- Si el bit és 1, usa $(c_{2i}, c_{2i+1}, c_{2i+2})$ per calcular $(c_{4i+2}, c_{4i+3}, c_{4i+4})$.

**Exemple 8.** *Es vol calcular $S_{24}(c)$ segons l'algorisme anterior. Com 24 no es pot expressar de la forma $n = 2k + 1$, calcularem $S_{23}(c)$ i llavors, mitjançant $c_{n+2} = c \cdot c_{n+1} - c^p \cdot c_n + c_{n-1}$ obtindrem $S_{24}(c)$. Per tant, si expressem $23 = 2 \cdot 11 + 1$ ens queda que $k = 11$, que la seva representació binària és $1011$ $(b_0, b_1, b_2, b_3)$. A partir d'aquí es recorre la representació binària d'esquerra a dreta, tenint en compte que $S_0$, $S_1$, $S_2$ i $S_3$ estan calculats i els casos explicats anteriorment.*

1. *Comencem pel segon bit ja que $b_0$ és sempre és 1. Partim de $S_3(c) = (c_2, c_3.c_4)$. Com que $b_1 = 0$, tenim el primer cas amb $i = 1$ i obtenim $(c_4, c_5, c_6)$, que coincideix amb $S_5(c)$.*
2. *El següent bit és $b_2 = 1$, per això usem la fórmula del segon cas amb $i = 2$ i es computa $(c_{10}, c_{11}, c_{12})$, que coincideix amb $S_{11}(c)$.*
3. *En l'últim bit $b_3 = 1$ es repeteix el pas anterior amb $i = 5$ obtenint així $(c_{22}, c_{23}, c_{24})$, que coincideix amb $S_{23}(c)$, que és el que volíem.*

**Teorema 5.** *Donada la suma $c$ de les arrels de $F(c, x)$, la suma $c_n$ de les potències $n$-èssimes de les arrels de $F(c, x)$ pot ser computada amb $8\log_2(n)$ multiplicacions a $GF(p)$ ($7\log_2(n)$ multiplicacions a $GF(p)$ amb l'aritmètica millorada).*

**Dem.** Usant el lema 9 es prova que $c_{2n}$ es calcula amb dues multiplicacions a GF(p) i $c_{n+2}$, $c_{2n-1}$ i $c_{2n+1}$ amb quatre (amb l'aritmètica millorada, els tres últims resultats es calculen amb 3 multiplicacions enlloc de quatre). Llavors, amb l'algorisme de [18] i [20] és immediat.

**Notació 2.** *La computació de $Tr(g^k)$ donat $Tr(g)$ s'anomena exponenciació simple.*

Per diverses aplicacions, aquest tipus d'exponenciació no serà suficient ja que haurem de computar $Tr(g^a g^{bk})$ donats $a, b \in \mathbb{Z}$ i $S_k(Tr(g))$. Aquesta operació se la denotarà com l'exponenciació doble. En [20] es descriu un algorisme complex usant operacions entre matrius que necessita $8\log_2(a/b \mod q) + 8\log_2(b) + 34$ multipicacions a $GF(p)$.

En l'article [21] hi ha diversos algorismes interessants que milloren tant l'exponenciació simple com la doble. El més important és una generalització de l'exponenciació doble, on es calcula $c_{bk+al}$, donats $0 < a, b < q$, $c_k$, $c_l$, $c_{k-l}$ i $c_{k-2l}$. Aquest algorisme inicialitza

una variable nova per cada paràmetre inicial, $u_0 = k$, $v_0 = l$, $d_0 = b$ i $e_0 = a$, de manera que queda $u_0d_0 + v_0e_0 = bk + al$ i $c_{u_0}$, $c_{v_0}$, $c_{u_0-v_0}$ i $c_{u_0-2v_0}$. A continuació, a partir d'una casuística detallada en [21], s'actualitzen totes les variables de manera que en cada iteració es cumpleixi $u_id_i + v_ie_i = bk + al$ i calculi $c_{u_i}$, $c_{v_i}$, $c_{u_i-v_i}$ i $c_{u_i-2v_i}$, fins que $d_i = e_i$. Quan s'hagi assolit aquesta igualtat, com que $(u_i+v_i)d_i = bk+al$, primer es podrà calcular $c_{u_i+v_i}$ per la quarta propietat del lema 11, i finalment es calcularà $c_{(u_i+v_i)d_i} = c_{bk+al}$ fent una crida a l'algorisme d'exponenciació simple. Els autors de [21], gràcies a aquest algorisme conjecturen el següent.

**Conjectura 1.** *Donats els enters $a, b$ tal que $0 < a, b < q$ i els valors $c_k$, $c_l$, $c_{k-l}$ i $c_{k-2l}$, el valor de $c_{bk+al}$ pot ser computat en una mitja de $6\log_2(max(a, b))$ multiplicacions a $GF(p)$ utilitzant l'aritmètica millorada.*

Això significa un gran avenç en la computació del mètode XTR, degut a que millora la computació de l'exponenciació simple i redueix a més de la meitat el nombre de multiplicacions de l'algorime per computar l'exponenciació doble amb matrius.

**Observació 25.** *A partir d'aquest algorisme també se'n pot extreure un de molt eficient per exponenciació simple. Si es vol computar $c_u$, s'escull $a = round(\frac{3-\sqrt{5}}{2}u)$, $b = u - a$, $k = l = 1$ (on round(x) és l'enter més proper a x) i s'aplica l'algorisme generalitzat. Tenint en compte la conjectura, el cost d'aquest algorisme seria $5.2\log_2 u$ multiplicacions a $GF(p)$ de mitja, el que significa una millora substancial.*

Més endavant, degut a la necessitat de fer càlculs amb traces per poder dissenyar diferents aplicacions, Z. Jia et al. [37] van descriure la manera de computar el que es podria dir l'exponenciació triple en termes de la simple i la doble. Aquesta consisteix en computar $Tr(g^a g^{bk} g^r)$ donats $Tr(g)$, $S_k(Tr(g))$, $S_r(Tr(g))$ i $a, b, c \in GF(q)^*$

Una última operació definida és la multiplicació, és a dir, donats $a, b \in \mathbb{Z}$, $S_a(Tr(g))$ i $S_b(Tr(g))$, computar $Tr(g^{a+b})$. L'algorisme encarregat de trobar aquesta suma està descrit en [40] on s'usa en una aplicació del mètode XTR que s'explicarà més endavant.

## 5.3 Selecció de paràmetres

Els paràmetres del mètode XTR són els primers $p$ i $q$, i l'element $g$ de $GF(p^6)$.

La selecció dels paràmetres $p$ i $q$ han de complir que $q \mid p^2 - p + 1$ perquè el *subgrup XTR* que construïm no pertanyi a cap subgrup propi de $GF(p^6)$ i resisteixi l'atac de [34]. També s'ha de garantir que $p \equiv 2 \pmod 3$ per assegurar la velocitat de les operacions amb elements de $GF(p^p)$ com hem vist en el lema 9. Per assegurar la seguretat, almenys equivalent a l'RSA de 1024 bits, el tamany de $p$ ha de ser 1024/6, per tant, es cercarà $p$ d'aproximadament 170 bits i $q$ de 160 bits. Per això, a [20], A.K. Lenstra et al. descriu alguns algorismes per trobar $p$ i $q$ tal que compleixin totes aquestes premises.

El paràmetre més complicat de trobar és l'element del *subgrup XTR*. Havent trobat $p$ i $q$, volem calcular un element $c \in GF(p^2)$ tal que $c = Tr(g)$ per un element $g \in GF(p^6)$ d'ordre $q \mid p^2 - p + 1$. Com que $g$ ja està determinat per les arrels de $F(Tr(g), x)$, només caldrà trobar $Tr(g)$. Per això, si trobem $c \in GF(p^2)$ tal que $F(c, x)$ sigui irreductible, degut al punt 6 del lema 11, $c$ és la traça d'un element $h \in GF(p^6)^*$ d'ordre $> 3$ i que

divideix a $p^2-p+1$. Per tant, si $c_{(p^2-p+1)/q}(= h_0^{p^2-p+1)/q} + h_1^{p^2-p+1)/q} + h_2^{p^2-p+1)/q}) \neq 3$, la $Tr(g)$ pot ser definida com $c_{(p^2-p+1)/q}$ ja que compleix les condicions necessàries. Resumint, es redueix a buscar $c \in GF(p^2)$ tal que $F(c,x)$ sigui irreductible i $c_{(p^2-p+1)/q} \neq 3$

Per trobar un polinomi $F(c,x)$ irreductible, A.K. Lenstra descriu a [20] dos algorismes diferents, un basat en el *mètode de Scipione Del Ferro* per trobar arrels d'un polinomi de grau tres, i l'altre en un test d'rreductibilitat que trasllada el problema actual a trobar arrels d'un polinomi concret a $GF(p)$. El més eficient és el segon, que permet trobar la $Tr(q)$ amb $\frac{q}{q-1}(2.7\log_2(p) + 8\log_2(p^p-p+1)/q)$ multiplicacions a GF(p).

A part dels paràmetres públics necessaris pel mètode XTR que hem especificat, també acostuma a haver-hi una clau privada i una pública que es repeteixen en les aplicacions que s'explicaran més endavant. La clau privada és un enter aleatòri $k \in [2, q-3]$ i la pública $Tr(g^k)$, que es calcula de

$$S_k(Tr(g)) = (Tr(g^{k-1}), Tr(g^k), Tr(g^{k+1})).$$

Però hi ha certes aplicacions, que degut a que necessiten calcular exponenciacions dobles i triples, la clau pública ha de ser la tupla $S_k(Tr(g))$ enlloc de només l'element d'enmig.

Degut a que $S_k(c) \in GF(p^2)^3$, el tamany de la clau pública es multiplica per tres. Aquesta observació es pot millorar considerablement ja que donats $Tr(g)$, $Tr(g^k)$ i $Tr(g^{k-1})$ (o $Tr(g^{k+1})$) es pot aconseguir $Tr(g^{k+1})$ (o $Tr(g^{k-1})$). Fins i tot hi ha algun mètode, que sota certes condicions, aconsegueix $Tr(g^{k-1})$ i $Tr(g^{k+1})$, donats $Tr(g)$ i $Tr(g^k)$.

A [20], s'hi descriu una forma de calcular $Tr(g^{k-1})$ (i $Tr(g^{k+1})$) donats $Tr(g)$ i $Tr(g^{k+1})$ (i $Tr(g^{k-1})$), que necessita un algorisme d'inversió a $GF(p^2)$. Si tenim $x = x_1\alpha + x_2\alpha^2$,

$$\frac{1}{x} = \frac{x_2\alpha + x_1\alpha^2}{x_1x_2 + (x_1-x_2)^2}.$$

Si denotem $c = Tr(g)$ i $c_k = Tr(g^k)$, el teorema següent ens mostra el mètode desitjat.

**Teorema 6.**

- *Si* $k \neq p, 1-p \mod p^2-p+1$, *llavors* $c^p c_{k-1} - cc_k \neq 0$ *i*

$$c_{k+1} = \frac{c_k^p(c^2-3c^p) - c_{k-1}^p(c^{2p}-3c) - c_{k-1}^2 c + c_k^2(c^p-c^2) + c_k c_{k-1} c^{p+1}}{c^p c_{k-1} - cc_k}$$

- *Si* $k \neq -p, p-1 \mod p^2-p+1$, *llavors* $cc_{k+1} - c^p c_k \neq 0$ *i*

$$c_{k-1} = \frac{c_k^p(c^{2p}-3c) - c_{k+1}^p(c^2-3c^p) - c_{k+1}^2 c^p + c_k^2(c-c^{2p}) + c_k c_{k+1} c^{p+1}}{cc_{k+1} - c^p c_k}$$

**Dem.** La demostració d'aquest teorema es troba en l'article [19].

Aquest teorema, però, es pot millorar amb unes altres fórmules descrites per X.Chen et al. [40]. A part de ser més eficients, tenen l'avantatge que la condició sobre $k$ és més senzilla. Sigui $k \neq 1$, $c = Tr(g)$ i $c_k = Tr(g^k)$,

$$c_{k+1} = \frac{c^p c_{k-1}^{2p} + (3-c^{p+1})c_k^p c_{k-1}^p - c_k^p c_{k-1}^2 + (c_k^{p+1}-3)c^p c_{k-1} + c(c_k^{2p} - c_k^{p+2}) + (c^{p+2}-c^{2p})c_k}{c^{p+1} - c_k^{p+1}}$$

$$c_{k-1} = \frac{cc_{k+1}^{2p} + (3 - c^{p+1})c_k^p c_{k+1}^p - c_k^p c_{k+1}^2 + (c_k^{p+1} - 3)cc_{k+1} + c^p(c_k^{2p} - c_k^{p+2}) + (c^{2p+1} - c^2)c_k}{c^{p+1} - c_k^{p+1}}.$$

A priori, aquestes fórmules semblen més complicades, però degut a que elevar un element de $GF(p^2)$ a $p$ no costa res, són més eficients tal i com es detalla a [40]. Cal dir que aquesta millora és poc significativa ja que els dos mètodes tenen un petit nombre constant d'operacions a GF(p).

Finalment, imposant la condició $p \equiv 8 \pmod 9$, Lenstra et al. [20] presenten un algorisme per calcular $Tr(g^{k+1})$ donats $Tr(g)$ i $Tr(g^k)$, que necessita $10.6 \log_2(p)$ multiplicacions a $GF(p)$ usant l'aritmètica de [20].

## 5.4 Seguretat

Amb aquest nou mètode, es poden definir tres noves versions de problemes coneguts en grups multiplicatius de cossos finits. La primera versió XTR-DH és del problema Diffie-Hellman (DH).

**Problema 5.** *Problema DH Sigui G un grup cíclic finit de p elements (amb p primer) i g un generador d'aquest grup. Donats $g^a, g^b \in G$, trobar $g^{ab} \in G$.*

**Problema 6.** *Problema XTR-DH Sigui G el* subgrup XTR *de q elements (amb q primer) i g un generador d'aquest grup. Donats $Tr(g^a)$ i $Tr(g^b)$ del* subgrup XTR*, trobar $Tr(g^{ab})$.*

La versió XTR-DHD és del problema decisional de Diffie-Hellman (DHD) tal i com podem veure a continuació. Denotem $DH(g^a, g^b) = g^{ab}$ i $XDH(Tr(g^a), Tr(g^b)) = Tr(g^{ab})$.

**Problema 7.** *Problema DHD Sigui G un grup cíclic finit de p elements (amb p primer) i g un generador d'aquest grup. Donats $a, b, c \in G$, cal decidir si $c = DH(a, b)$.*

**Problema 8.** *Problema XTR-DHD Sigui G el* subgrup XTR *de q elements (amb q primer) i g un generador d'aquest grup. Donats a, b i c del* subgrup XTR*, trobar $c = XDH(a, b)$.*

Finalment, la versió XTR-DL és del logaritme discret (DL) (descrit a l'apartat de seguretat de Lucas), el problema més intractable d'aquests tres.

**Problema 9.** *Problema XTR-DL Sigui G el* subgrup XTR *de q elements (amb q primer) i g un generador d'aquest grup. Donat a del* subgrup XTR*, trobar $0 < x < q$ tal que $a = Tr(g^x)$.*

La clau per poder comparar la seguretat dels sistemes que usen el mètode XTR amb la d'altres sistemes criptogràfics és comprovar la relació entre aquests diferents problemes. Lenstra et al. [18] demostren un teorema que especifica aquestes relacions i que detallarem a continuació.

**Notació 3.** *Diem que un problema A és (a,b)-equivalent a un problema B, si qualsevol cas del problema A (o B) es pot resoldre, com a molt, amb 'a' (o 'b') crides d'un algorisme que resolgui B (o A).*

**Teorema 7.** *Sigui G el subgrup XTR definit anteriorment,*

1. *El problema XTR-DL és (1,1)-equivalent al problema DL en G.*
2. *El problema XTR-DH és (1,2)-equivalent al problema DH en G.*

*3. El problema XTR-DHD és (3,2)-equivalent al problema DHD en G.*

**Dem.** Només es detallarà la demostració del primer punt ja que serà l'únic que s'usarà en aquesta secció. Els altres punts estan demostrats a [18].
Per computar el $DL(y)$, es calcula el $XTR-DL$ de la traça de $y$ (suposem que és $x$). Llavors el $DL(y)$ serà $x \cdot p^{2j} \pmod q$ amb $j = 0$, $j = 1$ o $j = 2$. Per computar el $XTR-DL$ de $a$ només cal computar $DL(b)$ on b és una arrel de $F(a, x)$. □

**Remarca 2.** *Com es pot apreciar en el teorema anterior, un algorisme que resol els problemes usuals, pot ser transformat en un altre que resol les versions corresponents del mètode XTR, i viceversa.*

A partir d'aquí, ens centrarem en el problema del logaritme discret, que es en el que es basen la majoria de les aplicacions del mètode XTR. Segons la remarca anterior, resoldre el poblema XTR-DL és el mateix que resoldre el logaritme discret en un subgrup d'ordre $q$ del grup multiplicatiu $GF(p^6)$. Com s'explica a [18], la dificultat del logaritme discret en un subgrup de $GF(p^6)$, depèn del tamany del mínim subcòs que evolta aquest subgrup i del tamany de $q$.

Els paràmetres del mètode XTR es prenen de tal manera que el mínim subcòs que conté el *subgrup XTR* sigui $GF(p^6)$ i que aquest tingui un divisor $q$ prou gran. D'aquesta forma, el problema del logaritme discret en el *subgrup XTR*, és tan fort com al grup multiplicatiu $GF(p^6)^*$. Per tant, el problema XTR-DL garanteix la seguretat del problema del logaritme discret a $GF(p^6)^*$, però amb l'avantatge de que, usant traces, les operacions dins el *subgrup XTR* són més eficients que a $GF(p^6)^*$. En el cas de l'exponenciació simple, s'aconsegueix millorar 4.5 vegades la velocitat, en canvi, l'exponenciació doble es computa 4.65 vegades més ràpid gràcies als algorismes descrits en la secció 5.2.

Com es considera a [18], el logaritme discret en un grup multiplicatiu $GF(p^t)$ és més difícil de resoldre que la factorització del problema RSA amb $t \cdot \log_2(p)$ bits. En conseqüència, per aconseguir un sistema tan segur com l'RSA usant el mètode XTR, s'ha de prendre els paràmetres $p$ i $q$ de 170 i 160 bits respectivament.

Finalment, cal remarcar que, com el problema del logaritme discret amb les funcions de Lucas, el problema XTR-DL és vulnerable a atacs subexponencials, en aquest cas de complexitat $L_{p^6}\left[\frac{1}{3}, (\frac{64}{9})^{\frac{1}{3}} + o(1)\right]$.

## 5.5 Aplicacions del mètode XTR

Com hem vist que el mètode XTR té unes bones propietats criptogràfiques, a partir de [18], s'han publicat diferentes aplicacions on s'usa aquest mètode.

### 5.5.1 Versió XTR del mètode Diffie-Hellman

El mètode Diffie-Hellman, com s'ha explicat en l'apartat 4.6.1, és un mètode per acordar una clau secreta entre dos usuaris que només poden usar canals públics.

Els dos usuaris que volen trobar una clau comuna $K$ en secret, comparteixen els paràmetres $p$, $q$ i $Tr(g)$ escollits tal i com s'ha comentat en l'apartat 5.3. Llavors aquests usuaris han de seguir el següent procés.

1. L'usuari 1 escull una clau privada $a \in [2, q-3]$ i calcula

$$S_a(Tr(g)) = (Tr(g^{a-1}), Tr(g^a), Tr(g^{a+1})).$$

   Llavors envia $Tr(g^a) \in GF(p^2)$ a l'usuari 2.

2. L'usuari 2 escull una clau privada $b \in [2, q-3]$ i calcula

$$S_b(Tr(g)) = (Tr(g^{b-1}), Tr(g^b), Tr(g^{b+1})).$$

   Llavors envia $Tr(g^b) \in GF(p^2)$ a l'usuari 2.

3. L'usuari 1, que rep $Tr(g^b)$ i coneix $a$, calcula

$$S_a(Tr(g^b)) = (Tr(g^{(a-1)b}), Tr(g^{ab}), Tr(g^{(a+1)b})) \in GF(p^2)^3.$$

4. L'usuari 1, que rep $Tr(g^a)$ i coneix $b$, calcula

$$S_b(Tr(g^a)) = (Tr(g^{(b-1)a}), Tr(g^{ab}), Tr(g^{(b+1)a})) \in GF(p^2)^3.$$

El dos usuaris ja tenen un element en comú pertanyent a $GF(p^2)$. Per tant els dos poden determinar $K = Tr(g^{ab})$.

### 5.5.2 Criptosistema de clau pública XTR-ElGamal

Aquest criptosistema té la mateixa estructura que l'ElGamal, vist en la secció 4.6.2. Hi ha un receptor, i un usuari que li vol enviar un missatge $M$.

El receptor publica els paràmetres XTR, $p$, $q$ i $Tr(g)$ i tria una clau privada $k \in [2, q-3]$. Llavors calcula $S_k(Tr(g)) = (Tr(g^{k-1}), Tr(g^k), Tr(g^{k+1}))$, d'on tria $Tr(g^k)$ com a clau pública que serà coneguda per qui vulgui enviar el missatge.

- *Encriptació*: L'usuari que envia el missatge M, tria aleatòriament $b \in [2, q-3]$ i calcula

$$S_b(Tr(g)) = (Tr(g^{b-1}), Tr(g^b), Tr(g^{b+1})) \text{ i}$$

$$S_b(Tr(g^k)) = (Tr(g^{(b-1)k}), Tr(g^{bk}), Tr(g^{(b+1)k})).$$

  Després encripta el missatge $M$ amb un mètode de criptografia simètrica (són mètodes que usen la mateixa clau tan per xifrar com per desxifrar) usant com a clau privada $Tr(g^{bk}) \in GF(p^2)$. Finalment, envia $Tr(g^b, E)$ on $E$ és el missatge encriptat pel mètode simètric.

- *Desencriptació*: Com que el receptor coneix $k$ i $Tr(g^b)$, calcula

$$S_k(Tr(g^b)) = (Tr(g^{(k-1)b}), Tr(g^{bk}), Tr(g^{(k+1)b}))$$

  per trobar $Tr(g^{bk})$, que és la clau secreta del mètode simètric escollit per l'usuari que ha enviat el missatge, i així poder trobar $M$ a partir de $E$.

### 5.5.3 Signatura de XTR-Nyberg-Rueppel

Els paràmetres segueixen sent els mateixos $p$, $q$ i $Tr(g)$, i se li afegirà una funció hash $H$.

- *Generació de claus*: La clau privada serà un enter aleatori $k \in [2, q-3]$ i la clau pública serà la terna

$$S_k(Tr(g)) = (Tr(g^{k-1}), Tr(g^k), Tr(g^{k+1})).$$

- *Signatura*: Per signar el missatge $M$, a partir d'un nombre aleatori $u \in [2, q-3]$, es calcula

$$S_u(Tr(g)) = (Tr(g^{u-1}), Tr(g^u), Tr(g^{u+1})).$$

  D'aquest càlcul en surt la clau privada $Tr(g^u)$ que s'usarà per l'encriptació simètrica que s'aplicarà a $M$ i d'on obtindrem $E$. La signatura de $M$ serà la tupla $(E, s)$, on $s = k \cdot H(E) + u \pmod q$

- *Verificació*: Primer cal comprovar que $s \in \{0, 1, \ldots, q-1\}$ i es pren l'invers de H(E) respecte la suma mòdul $q$. Usant la clau pública i $Tr(g)$, es computa

$$Tr(g^s \cdot g^{-H(E)k}) = Tr(g^{s-H(E)k}) = Tr(g^u).$$

  Com que el mètode d'encriptació simètric és públic i ja s'ha obtingut la clau privada d'aquest, es comprova si la $E$ de la signatura és correcte.

A [38], C.Hu et al. presenten un context on seria interessant usar aquesta signatura. Dissenyen un Certificate authority system usant la signatura XTR-Nyberg-Rueppel, amb la qual s'estalvia temps i espai sense perdre seguretat.

Una *Public Key Infrastucture* (PKI) és una combinació de hardware i software, polítiques i procediments de seguretat que permeten l'execució amb garanties d'operacions criptogràfiques com el xifrat o les signatures digitals. El Certification Authority és una entitat, que normalment actua com a nucli de diversos PKI. La seva missió és repartir certificats digitals i poder-los usar amb les altres parts. La signatura XTR-Nyberg-Rueppel intervé a l'hora de crear aquests certificats digitals i compartir-los amb les altres parts.

### 5.5.4 Signatura XTR-DSA

La signatura DSA [35] és un estàndard del Govern Federal dels Estats Units per firmes digitals, és a dir, una referència en la criptografia actual.

Els paràmetres que s'han d'escollir per poder signar amb el DSA són una funció hash $H$ que normalment és el SHA-2, dos enters primers $p$ i $q$ (de 1024 i 160 bits respectivament) tals que $q \mid (p-1)$ i $g \in GF(p)^*$ d'ordre $q$.

- *Generació de claus*: Es tria $x \in [1, q-1]$ aleatòriament com a clau privada, i es calcula la clau privada $y$ de manera que $y \equiv g^x \pmod p$.

- *Signatura*: Per signar un missatge $M$, es tria aleatòriament $k \in [1, q-1]$ per calcular $r = (g^k \pmod p) \pmod q$. Seguidament es computa $s = k^{-1}(H(M) + xr) \pmod q$. La signatura del missatge $M$ que s'envia és $(r, s)$.

- *Verificació*: Primer es computa l'invers multiplicatiu de $s$, és a dir, $s^{-1} \pmod q$. En segon lloc es calcula $u_1 \equiv H(M)s^{-1} \pmod q$ i $u_2 \equiv rs^{-1} \pmod q$. Finalment es comprova que $(g^{u_1}y^{u_2} \pmod p) \pmod q$ és igual a terme $r$ de la signatura.

A [18] es presenta una versió d'aquesta signatura a la que es denomina com signatura XTR-DSA. Com a paràmetres es prenen els necessaris pel mètode XTR $p$, $q$ i $Tr(g)$, i a més a més, la mateixa funció hash $H$ que en la signatura DSA.

- *Generació de claus*: La clau privada serà un enter aleatori $k \in [2, q-3]$ i la clau pública serà la terna

$$S_k(Tr(g)) = (Tr(g^{k-1}), Tr(g^k), Tr(g^{k+1})).$$

- *Signatura*: Per signar el missatge $M$, a partir d'un nombre aleatori $u \in [2, q-3]$, es calcula

$$S_u(Tr(g)) = (Tr(g^{u-1}), Tr(g^u), Tr(g^{u+1})).$$

  Com que $Tr(g^u) \in GF(p^2)$ el podem expressar com $Tr(g^u) = x_1\alpha + x_2\alpha^2$. Llavors es computa $r = x_1 + px_2 \pmod q$ i $s = u^{-1}(H(M) + kr) \pmod q$, on $r$ i $s$ seran els elements de la signatura.

- *Verificació*: Abans de tot s'ha de comprovar que $r, s \in [1, q-1]$ i computar $s^{-1} \pmod q$. Després es calcula $u_1 \equiv H(M)s^{-1} \pmod q$ i $u_2 \equiv rs^{-1} \pmod q$. Com que tenim $Tr(g)$ i la clau pública es pot calcular

$$Tr(g^{u_1} \cdot g^{u_2 k}) = Tr(g^{s^{-1}(H(M)+kr)}) = Tr(g^u).$$

  A partir d'aquí es repeteix el procés de signatura amb $Tr(g^u)$ i es comprova que el valor resultant és igual a $r$.

### 5.5.5 Signatures Blind basades en XTR

Aquest tipus de signatures, que juguen un paper important en el comerç electronic van ser introduïdes per D. Chaum [39]. Més endavant, es van presentar esquemes de signatures *blind* basades en el logaritme discret. A continuació, veurem dues variants basades en el mètode XTR que es van presentar a [40].

Una signatura *Blind* és una forma de signatura digital en la que el contingut del missatge que es vol signar és ocultat abans de ser signat. Acostumen a haver-hi dos usuaris, dels quals un amaga el missatge i l'altre el signa.

#### Signatura XTR-Blind-Schnorr

Els paràmetres són els del mètode XTR, $p$, $q$ i $Tr(g)$, i una funció hash $H$. Hi ha dos usuaris A i B que signaran el missatge $M$ entre els dos. La clau privada l'escull A aleatòriament $x \in [2, q-3]$, i la clau pública és $y = Tr(g^x)$, calculada mitjançant

$$S_x(Tr(g)) = (Tr(g^{x-1}), Tr(g^x), Tr(g^{x+1})).$$

El procés de signatura és un intercanvi d'informació entre A i B, que van ocultant el missatge.

1. A tria aleatòriament $r \in [2, q-3]$, computa $S_r(Tr(g))$ i envia $Tr(g^x) \in GF(p^2)$ a B.

2. B escull $\alpha \in GF(q)$ i $\beta \in GF(q)^*$ aleatòriament i calcula $Tr(g^{\alpha+\beta x})$ a partir de $Tr(g^x)$. Com que B coneix $Tr(g^k)$ i $Tr(g^{\alpha+\beta x})$, computa $r' = Tr(g^{k+\alpha+\beta x}) \in GF(p^2)$. Sigui $r' = (a, b)$, computa $e' = H(M\|ap + b)$ i envia $e = e' - \alpha \pmod q$ a A.

3. A torna a enviar $s = k - ex \pmod q$ a B.

4. B finalment computa $s' = s + \alpha \pmod q$. La signatura de $M$ és (e',s').

El procés de verificació consisteix en calcular $S_{s'}(Tr(g))$ i $S_{e'}(y)$, i computar $Tr(g^{s'+e+x})$. Llavors, sigui $(a', b') = Tr(g^{s'+e+x})$, s'ha de comprovar que $H(m\|a'p + b') = e'$.

**Signatura XTR-Blind-Nyberg-Rueppel**

Amb els mateixos paràmetres i claus que en la signatura anterior, es vol signar $M$. A part, també s'usa un algorisme d'encriptació simètrica $E$. L'estructura és la mateixa, on apareixen els usuaris que signen A i B.

1. A tria aleatòriament $\tilde{k} \in [2, q-3]$, computa $S_{\tilde{k}}(Tr(g))$ i envia $Tr(g^{\tilde{k}}) \in GF(p^2)$ a B.

2. B escull $\alpha \in GF(q)$ i $\beta \in GF(q)^*$ aleatòriament i calcula $Tr(g^{\alpha+\beta\tilde{k}})$ a partir de $Tr(g^{\tilde{k}})$. La clau privada de l'algorisme E es basarà en $Tr(g^{\alpha+\beta\tilde{k}})$, d'aquí es calcula $r = E(M)$ i $\tilde{M} = r\beta^{-1} \pmod q$. Llavors es verifica que $\tilde{M} \in GF(q)^*$ i l'envia a A.

3. A calcula $\tilde{s} = \tilde{M}x + \tilde{k} \pmod q$ i l'envia a B.

4. B calcula $s = \tilde{s}\beta + \alpha \pmod q$. La signatura de $M$ és $(r, s)$.

Per verificar la signatura cal calcular $S_s(Tr(g))$ i $S_{-r}(y)$ i computar $Tr(g^{s-rx})$. Llavors, com que

$$Tr(g^{s-rx}) = Tr(g^{\tilde{s}\beta+\alpha-rx}) = Tr(g^{\tilde{M}x\beta+\tilde{k}\beta+\alpha-rx}) = Tr(g^{\alpha+\tilde{k}\beta}),$$

podem obtenir la suposada clau privada de $E$ i només cal comprovar si la $r$ de la signatura és correcte.

**Observació 26.** *Aquesta signatura es pot variar per prescindir de l'algorisme de d'encriptació simètrica calculant* $r = (ap + b)M \pmod q$ *on* $(a, b) = Tr(g^{\alpha+\beta\tilde{k}})$.

### 5.5.6 XTR-Signatura digital basada en la identitat

La criptografia basada en la identitat és un tipus de criptografia de clau pública en la cual la clau pública d'un usuari és una informació personal referent a aquest com per exemple l'empremta digital, el número del DNI, l'adreça de correu, etc. A part dels usuaris que intervenen en el procés de signatura i verificació, també hi ha una altra part anomenada Generador de Claus Privades (PKG), que és l'encarregat de generar la corresponent clau privada a l'usuari que signa. Per que això sigui possible, el PKG té una clau privada i una clau pública que depèn de la privada i que tothom hi té accés.

La primera signatura basada en la identitat va ser proposada el 1984 per Shamir [36]. La majoria d'aquest tipus de signatures estan basades en aparellaments bilinears (*bilinear pairings*), fins que l'any 2008, se n'ha fet una versió usant el mètode XTR a [37].

Els paràmetres de la XTR-Signatura digital basada en la identitat són $p$, $q$ i $Tr(g)$ que es trien pel mètode XTR. El PKG tria aleatòriament $k < n$ que serà la clau privada, una funció hash $H$ i computa

$$S_k(Tr(g)) = (Tr(g^{k-1}), Tr(g^k), Tr(g^{k+1})),$$

per obtenir la clau pública que serà $Tr(g^k)$.

- *Registre de l'usuari*: En aquest tipus de signatures, l'usuari ha de rebre una clau privada de el PKG depenent de la seva identitat. Sigui $ID$ la identitat de l'usuari que vol signar el missatge $M$. El PKG verifica $ID$, tria aleatòriament $r \in GF(q)^*$, computa $s_{ID} \equiv kH(ID) + r \pmod q$ i escull $R = Tr(g^r)$ del càlcul

  $$S_r(Tr(g)) = (Tr(g^{r-1}), Tr(g^r), Tr(g^{r+1})).$$

  La clau privada de l'usuari que signa subministrada pel PKG serà $(s_{ID}, R)$. Cal remarcar que $(s_{ID}, R)$ s'envia per un canal segur, de manera que només ho conegui l'usuari que signa i el PKG.

- *Signatura*: L'usuari que signa, primer escull aleatòriament $a \in [2, q-3]$ i computa

  $$S_a(Tr(g)) = (Tr(g^{a-1}), Tr(g^a), Tr(g^{a+1})),$$

  d'on $z = Tr(g^a)$. Llavors usa la funció hash escollida per calcular $e = (H(M\|z))$ on $\|$ significa l'operació concatenar. Finalment, calcula $s = s_{ID}e + a \pmod q$ i la signatura d'un missatge $M$ és $(M, e, s, R)$.

- *Verificació*: Si es vol comprovar que la signatura és vàlida, simplement es calcula $b = -e \pmod q$ i $c = bH(ID)$. Llavors, gràcies a l'algorisme de [37], es computa $z' = Tr(g^s g^{ck} g^{br})$ i es verifica que $H = (M\|z') \equiv e$.

La verificació funciona degut a que

$$\begin{aligned} z' &= Tr(g^s g^{ck} g^{br}) \\ &= Tr(g^{s_{ID}e+a \pmod q} g^{-ekH(ID) \pmod q} g^{-er \pmod q}) \\ &= Tr(g^{ekH(ID)+er+a \pmod q} g^{-ekH(ID) \pmod q} g^{-er \pmod q}) \\ &= Tr(g^a) \\ &= z. \end{aligned}$$

# Capítol 6

# Criptografia basada en el Tor

En la criptografia de clau pública molts criptosistemes estan basats en la intractabilitat del problema del logaritme discret, en concret els que usen grups ciclics d'extensions de cossos com els sistemes de Lucas i en els que s'aplica el mètode XTR.

En aquests dos casos anteriors, com s'ha vist durant el treball, s'escull un subgrup cíclic del grup multiplicatiu del cos finit $GF(p^n)$, i mitjançant les traces dels elements s'aconsegueix una representació eficient i compacta sobre un grup multiplicatiu d'un cos finit $GF(p^t)$ de menor dimensió ($t = 1$ en el cas de Lucas i $t = 2$ en el mètode XTR). W.Bosma et al.[41] van plantejar una conjectura sobre si es podien extendre aquests mètodes (especialment l'XTR) en extensions de cossos finits de grau arbitrari, però va ésser possible fins que K.Rubin et al.[42] van introduir una generalització basada en els tors algebraics (són generalitzacions del grup multiplicatiu), donant així una nova interpretació àlgebro-geomètrica dels sistemes de Lucas i el mètode XTR.

**Definició 12.** *Un tor algebraic $T$ sobre $GF(p)$ és un grup algebraic definit sobre $GF(p)$ tal que sobre alguna extensió de cossos finita és isomorf a $(G_m)^d$, on $G_m$ és el grup multiplicatiu i $d$ és necessàriament la dimensió de $T$. Sigui la $N_{L/F}(\alpha)$ la norma d'$\alpha$ sobre $F$, el tor $T_n(GF(p))$ es defineix com*

$$T_n(GF(p)) \cong \left\{\alpha \in GF(p)^* \mid N_{GF(p^n)/F}(\alpha) = 1, \text{ on } k \subseteq F \subseteq GF(p^n)\right\}.$$

Aquesta generalització es basa en un lema en que identifica $T_n(GF(p))$ amb el subgrup cíclic $G_{p,n} \subset GF(p^n)^*$ d'ordre el polinomi n-ciclotòmic avaluat en $p$, i mostra que la seguretat dels criptosistemes basats en el logaritme discret en el grup $T_n$ (el tor) és la del grup multiplicatiu $GF(p^n)^*$. Llavors, es demostra que una parametrització racional del tor, dona una representació compacta del grup $T_n(GF(p))$.

Aquesta representació és més compacta que la dels sistemes de Lucas i XTR ja que, a diferència d'aquests, permet usar l'operació multiplicativa a part de l'exponenciació. Cal comentar que el mètode XTR sí que presenta alguns algorismes on es pot fer la multiplicació (donat $Tr(g^a)$ i $Tr(g^b)$, computar $Tr(g^{a+b})$), però la difícil computació d'aquests fa que es perdi part de la velocitat guanyada amb la representació compacta. A [42] hi ha descrit un criptosistema a $GF(p^6)$ com l'XTR anomenat CEILIDH, basat en la criptografia al tor, que té aquest avantatge que s'acaba d'esmentar.

El problema dels sistemes de Lucas i XTR, ve donat degut a que la funció traça no és una parametrització racional, i per un element i els seus conjugats la imatge és la mateixa.

# Capítol 7

# Conclusions

Els criptosistemes basats en les funcions de Lucas i el mètode XTR són una bona alternativa als protocols més usats com RSA o el DSA. L'avantatge més important és que es poden prendre paràmetres de tamany més petit que en els mètodes com RSA assolint la mateixa seguretat. Això pot ser útil en entorns on es disposi de poc espai per emmagatzemar variables i no sigui tan important l'eficiència amb que s'executi.

Tot i que aquests criptosistemes no són tan ràpids de computar com l'RSA o sistemes basats en el logaritme discret en $GF(p)^*$, s'ha demostrat que poden ser bastant eficients, i que poden tenir un marge de millora trobant un mètode que calculi cadenes de Lucas més curtes (en el cas de les funcions de Lucas).

Com hem pogut veure durant el treball, hi ha una gran varietat d'aplicacions criptogràfiques usant aquests dos mètodes. Cal remarcar-ne una en especial en que les funcions de Lucas fan factible les signatures digitals en canals subliminals, cosa que no era possible amb el mètodes tradicionals.

Finalment, en l'últim capítol es generalitzen aquests dos casos concrets, de manera que s'obren un munt de possibilitats en la criptografia sobre cossos finits.

# Bibliografia


[1] Lehmer D.H. *An extended theory of Lucas' functions*, Ann. of Math. (2), Vol.31 (1930), 419–448.

[2] Williams H.C. *A p+1 method of factoring*, Mathematics of Computation, Vol.39 (1982), 225–234.

[3] Smith P.J., Lennon M.J.J. *LUC: a new public key system*, Proceedings of the Ninth IFIP Int. Symp. on Computer Security (1993), 103–117.

[4] Smith P.J., Skinner C. *A public key cryptosystem and digital signature system based on the Lucas function analogue to discrete logarithms*, Pre-proceedings Asiacrypt (1994), 298–306.

[5] Yen S.-M., Laih C.-S., *Fast algorithm for LUC digital signature computation*, IEE Proc. Comput. Digit. Tech. Vol. 142 2 (1995), 165–169.

[6] Laith C.-S., Tu F.-K., Tai W.-C., *Remarks on LUC public key system*, Electronics Letters, Vol.30 (2) (1994), 123–4.

[7] Murphy S., *Comment: Remarks on LUC public key system*, Electronics Letters, Vol.30 (7) (1994), 558–9.

[8] Horster P., Michels M., Petersen H., *Digital signature schemes based on Lucas functions*, University of Technology Chemnitz-Zwickau, Technical Report TR-95-1, in: Communications and Multimedia Security, IT-Sicherheit '95, Joint working conference IFIP TC-6 TR-11 and Austrian Computer Society, Graz, Sept., (1995), 20–21.

[9] Bleichenbacher D., Bosma W., Lenstra A. K., *Some remarks on Lucas-based cryptosystems*, in: Proceedings of CRYPTO'95, in: Lecture Notes in Comput. Sci., vol. 963, Springer, (1995), 386–396.

[10] Castagnos G., *An efficient probabilistic public-key cryptosystem over quadratic fields quotients*, Finite Fields Appl. 13 (3) (2007), 563–576.

[11] Laith C.-S., Tu F.-K., Tai W.-C., *On the security of the Lucas function*, Information Processing Letters 53 (1995), 243–247.

[12] Joye M., Quisquater J. J., *Efficient computation of full Lucas sequences*, Electronics Lett. 32 (1996), 537–538.

[13] Wang C.-T., Chang C.,-C., Lin C.-H., *A method for computing Lucas sequences*, Computers and Mathematics with Applications 38 (1999), 187–196.

[14] Chiou S.Y., Laih C.S., *An efficient algorithm for computing the Luc chain*, IEE Proc. Comput. Digit. Tech., Vol. 147, 4 (2000), 263–265.

[15] Ali Z.M., Othman M., Said M.R.M., Sulaiman M.N., *Two Fast Computation Algorithms for LUC Cryptosystems*, Proceeding of the International Conference on Electrical Engineering and Informatics (IEEI2007), ITB, Vol. 2, (2007), 434–437.

[16] Harn L., Gong G., *Digital signature with a subliminal channel*, IEE Proc. Comput. Digit. Tech., vol. 144 (6), (1997), 387–389.

[17] Castagnos G., Vergnaud D., *Trapdoor permutation polynomials of $\mathbb{Z}/n\mathbb{Z}$ and public key cryptosystems*, ISC 2007, LNCS 4779 (2007), 33–350.

[18] Lenstra A. K., Verheul E. R., *The XTR public key system*, Proceedings of Crypto 2000, LNCS 1880, Springer-Verlag (2000), 1–19.

[19] Lenstra A. K., Verheul E. R., *Key improvement to XTR*, Proceedings of Asiacrypt 2000, LNCS 1976, Springer-Verlag (2000), 220–233.

[20] Lenstra A. K., Verheul E. R., *An Overview of the XTR public key system*, In Public-Key Cryptography and Computational Number Theory, Walter De Gruyter Inc. (2001), 151–181.

[21] Stam M., Lenstra A. K., *Speeding Up XTR*, Proceedings of Asiacrypto 2001, LNCS 2248, Springer-Verlag (2001), 125–143.

[22] Li W.-C.W., Näslund M., Shparpinski I. E., *Hidden number problem with the trace and bit security of XTR and LUC*, CRYPTO 2002, LNCS 2442 (2002), 433–448.

[23] Adleman L.M., DeMarrais J., *A subexponential algorithm for discrete logarithms over all finite fields*, Proceedings Crypto'93, Lecture Notes in Comp. Sci. 773 (1994), 147–158.

[24] Gordon D., *Discrete logarithms in $GF(p)$ using the number field sieve*, SIAM J, Disc. Math. 6 (1993), 124–138.

[25] Schirokauer O., *Using number fields to compute general discrete logarithms*, Mathematics of Computation, Vol.69 (231) (2000), 1267–1283.

[26] Catalano D., Gennaro R., Howgrave-Graham N., NguyenP.Q., *Paillier's cryptosystem revisited*, CCS'01, Proceedings if the 8th ACM Conference on Computer and Communications Security, ACM (2001), 206–214.

[27] Paillier P., *Public-key cryptosystems based on composite degree residuisity classes*, Proceedings of EUROCRYPT'99, Lecture Notes in Comput. Sci., vol. 1592, Springer (1999), 223–238.

[28] Galindo D., Martín S., Morillo P., Villar J.L. *An efficient semantically secure elliptic curvecryptosystem based on KMOV*, Proceedings of WCC'03 (2003), 213–221.

[29] Knuth D.E., *The art of Computer Programming*, Vol. II: Seminumerical Algorithms, 3rd Edition, MA: Addison-Wesley (1997).

[30] Yen S.-M., Laih C.-S., *Common-multiplicand multiplication and its applications to public key cryptography*, Electronics Letters, vol. 29, 17 (1993), 1583–1584.

[31] Montgomery P.L., *Modular multiplication without trial division*, Mathematics of Computation vol. 44, 170 (1985), 519–521.

[32] Stalling W., *Cryptography and Network Security; Principles and Practice*, Prentice-Hall (1999).

[33] Koren I., *Computer Arithmetic Algorithms*, 2nd Edition, A.K. Peters, Natick, MA (2002).

[34] Lenstra A.K., *Using cyclotomic polynomials to construct efficient discrete logarithm cryptosystems over finite fields*, Proceedings ACISP97, LNCS 1270, Springer-Verlag, (1997), 127–138.

[35] *Digital Signature Standard*, Federal Information Processing Standards Publication 186, NIST (1994).

[36] Shamir A., *Identity-based cryptosystem and signature schemes*, In proc. Crypto'84. Santa Barbara, CA: Springer-Verlag, (1984), 47–53.

[37] Jia Z., Jiqiang L., Zhen H., Changxiang S., *Identity based digital signature algorithm of XTR system*, 9th International Conference on Signal Processing, (2008), 2816–1819.

[38] Hu C., Xu N., *Application of XTR Public Key System in Certificate Authority System*, International Conference on Computer Science and Software Engineering, CSSE, vol. 5 (2008), 1233–1236.

[39] Chaum D., *Blind Signature for Untraceable Payments*, Eurocrypt'82, Plenum Press, (1983), 199–203.

[40] Chen X., Feng F., Wang Y., *New key improvements and Its Application to XTR System*, Proc. of 17th International Conference on Advanced Inf. Networking and App. AINA'03, (2003), 561–1.

[41] Bosma W., Hutton J., Verheul R., *Looking beyond XTR*, in Advances in Cryptology - Asiacrypt 2002, Lect. Notes in Comp. Sci. 2501, Springer, Berlin, (2002), 46–63.

[42] Rubin K., Silverberg A., *Torus-based cryptography*, in D.Boneh, editor, Advances in Cryptology - CRYPTO 2003, vol. 2729 of Lecture Notes in Computer Science, Springer-Verlag (2003), 349–365.

[43] Janusz G.J., *Algebraic Number Fields*, American Mathematical Society, 2nd edition (1997).

[44] Barbeau E.J., *Pell's equation*, Springer, (2000).