POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Matematica

Tesi di Laurea Magistrale

Sistemi Post-Quantum: Crittografia Multivariata

Relatori
prof. Danilo Bazzanella
dott. Carlo Sanna
firma dei relatori

. . . . . .
. . . . . .

Candidato
Adriano Koleci

firma del candidato

. . . . . .

Anno Accademico 2021-2022

A Kry

Sommario

Questo elaborato si occupa di offrire una panoramica della crittografia multivariata. Il primo capitolo è dedicato alla definizione di una cifratura simmetrica e asimmetrica e a una breve panoramica sul crittosistema RSA, attualmente in uso nella maggior parte delle applicazioni. Del sistema RSA è stata offerta una panoramica di alcuni attacchi classici ed è stato analizzato l'algoritmo di Shor, un algoritmo che viene usato per la fattorizzazione degli interi in un tempo polinomiale su un computer quantistico e che di conseguenza rompe definitivamente RSA. Questa rottura ha reso necessario trovare un nuovo standard di crittografia e per questo motivo il NIST ha lanciato una competizione per trovare un nuovo standard di crittografia post-quantum.

Tra gli algoritmi post-quantum sono presenti i crittosistemi multivariati a chiave pubblica. Nel secondo capitolo è stata offerta una panoramica di questi crittosistemi concentrandosi sui sistemi bipolari e sulla loro loro struttura. In particolare dei sistemi bipolari ci si è concentrati sui sistemi Oil-Vinegar e Hidden Field Equation. Del primo sistema, un sistema di firma, è stato mostrato come generare una firma ed è stato offerto un esempio di firma con questo crittosistema. Del secondo crittosistema, che può essere usato sia per cifrare che per firmare, invece è stato mostrare come cifrare e come firmare un documento. Questi sistemi sono alla base di due finalisti della competizione indetta dal NIST per i sistemi crittografici post-quantum: Rainbow e GeMSS.

Nel terzo capitolo viene descritto il crittosistema Rainbow, finalista della competizione del NIST, che è basato su Oil-Vinegar. Del crittosistema è stato descritto il funzionamento e sui dettagli relativi ai parametri delle implementazioni proposte dagli autori al NIST.

Nel quarto capitolo viene descritto il crittosistema GeMSS, finalista alternativo della competizione del NIST per un eventuale quarto round, che è basato su HFE. Del crittosistema, in analogia a quanto visto con Rainbow, è stato descritto il funzionamento e i dettagli relativi ai parametri delle implementazioni proposte dagli autori al NIST.

Ringraziamenti

Eccomi qui, a scrivere i titoli di coda che in realtà saranno i titoli di testa.

Il primo ringraziamento va innanzitutto al Professor Danilo Bazzanella e al Dottor Carlo Sanna che hanno contribuito a rendere l'ultimo lavoro della mia laurea magistrale un lavoro motivante e appassionante, per cui valesse la pena dedicare il massimo impegno.

Arrivare fino a questo punto non è stato facile e chi mi è stato vicino lo sa. Dopo anni di lenta discesa è arrivato l'anno 2020 che ha reso chiaro quanto ci fosse qualcosa che non andava bene e che mi ha segnato definitivamente. È stata, per me, l'ora più buia. Riuscire a ricominciare da dove avevo lasciato è stata la sfida più difficile da portare avanti ed è stato un processo lungo, anche molto doloroso, ma che alla fine ha ripagato. Con il passare dei mesi ho capito perché ero dov'ero, perché studio matematica e cos'è che davvero conta per me: la conoscenza, che è tutto ciò che mi ha spinto ad andare avanti. Per questo devo ringraziare Adriana, che in un momento difficile di settimana in settimana mi ha guidato nella risalita. Mi ha aiutato a diventare una persona migliore.

Superare questi anni è stato dolorosissimo, ma grazie alle persone che mi sono sempre state vicino, questo è stato molto più facile. Un grande ringraziamento va al quarto piano del Collegio Einaudi sezione San Paolo (e non solo), ormai totalmente rinnovato rispetto a solo 4 anni fa, quando ho messo piede per la prima volta qui. L'esperienza mi ha chiarito che ciò che conta, alla fine, sono solo le persone che lo popolano e che lo rendono speciale, a modo suo. L'arrivo di una ventata di aria fresca ha dato spazio alla massima espressione di me stesso, senza avere alcun timore di sembrare strano. Per questo motivo ringrazio tutti da Roberto a Salvatore e non sto a scrivere tutti i nomi perché ne conto almeno 40 e i titoli di coda non devono mai durare quanto tutto il film.

Ringrazio anche gli amici di IMDIP che non sto ad elencare perché alla fine sono gli stessi pazzi che ringraziai in triennale e per cui provo sempre molto rispetto e che mi aiutano a superare anche le giornate più tristi. In particolare però ringrazio Cerru, che ormai conosco da una vita, per essere sempre stato al mio fianco.

La vita dentro l'università invece, come detto prima, è stata tutt'altro che facile. Ma senza tutti i colleghi e non con cui ho avuto a che fare in questi anni, con cui ho condiviso i lavori di gruppo e battute (oggettivamente le migliori) e che hanno sempre contribuito a presentare un lavoro non meno che eccellente, in ogni situazione. Grazie Gianluigi, Kairi e Marco per avermi accompagnato in questa pazza corsa. Non posso dimenticare Marianna, la miglior collega che io abbia mai avuto. Nonostante l'inizio non proprio stellare a causa della mia personalità, il rapporto di ora è qualcosa che non scambierei per nulla al mondo. Non solo una collega, un'amica, un'amica strettissima e anche una delle persone più buone

che io conosca. A lei devo tanto, se questa pazza corsa è arrivata alla fine è anche per merito suo.

L'università mi ha anche dato qualcosa per cui non smetterò mai di ringraziare ed è il percorso talenti. Sebbene io sia sempre stato critico nei confronti del percorso in sé, nuovamente ciò che l'ha reso particolare e importante sono state le persone che l'hanno reso speciale. Grazie a Pasca per avermi fatto apprezzare tutto quello che abbiamo fatto e grazie per essermi vicina ancora nonostante ti sia imbruttita.

Il percorso talenti però è anche quello che mi ha fatto conoscere Simone ed Alessandro, due persone splendide che mi hanno accompagnato dall'inizio dell'università e non hanno mai mollato un colpo, anche in situazioni di difficoltà. Sono sempre state al mio fianco e a loro devo tanto per tutto quello che abbiamo passato insieme.

Il ringraziamento va anche ai Brignone's, in particolare al gruppo BaSA. Un gruppo davvero incredibile che mi regala sempre tante emozioni, un gruppo con il quale ho passato delle vacanze fantastiche e che anche in tempi di pandemia hanno fatto sentire la loro vicinanza. Una cosa non da tutti.

Tutto quello che ho detto, però, è stato in parte possibile grazie ai miei genitori e alla mia famiglia. Grazie a loro ho fatto un'esperienza che non tutti possono fare. Grazie anche ai miei cugini, in particolare Francesco e Anna per il tempo passato insieme.

Ciò che però mi ha dato il Collegio Einaudi è stata la possibilità di conoscere Serena. Iniziata non nel migliore dei modi, da fine 2019 mi accompagna ogni singolo giorno. Abbiamo passato dei momenti difficili, è stata vicino a me in ogni match, è stata vicino a me quando era lontana più di un migliaio di chilometri, è stata vicino a me anche nei momenti più litigiosi. Ho avuto l'onore di condividere con lei tantissimi bei momenti che mi hanno arricchito e a cui sono molto grato. Ci sono stati tanti momenti difficili tra di noi ma alla fine siamo qui, forti perché siamo insieme. Ti amo con tutto il mio cuore.

Il ringraziamento più sentito però va a te Kry, che sei la sorella migliore che potessi avere. Sei la mia ispirazione, fin da quando sono piccolo, per quello che fai, per l'impegno che ci metti e per la passione che hai. Sei una ricchezza inestimabile per me, hai avuto la forza di stare vicino a me anche quando faticavo ad andare avanti. L'amore che ci hai messo per cucinarmi quando ho ripreso a mangiare è stato qualcosa che mi ha fatto sentire apprezzato da qualcuno, in un momento di grande crisi. Sei la parte migliore di me e sei la parte più importante di me. Ti voglio bene più di ogni altra cosa al mondo.

L'ultimo ringraziamento però va a chi ha scritto tutto ciò, cioè il sottoscritto. Senza tutta la forza che ci ho messo per superare le mie grandi difficoltà non sarei qui ora, a scrivere queste parole di lieto fine a una storia che è stata tutt'altro che facile. Questo è il momento in cui si chiude il cerchio di tutto quello che ho fatto e mi ritengo estremamente fortunato perché non tutti riescono a superare quello che ho passato. Ero a un passo dal mollare ma alla fine non ho mollato, capendo di aver bisogno di aiuto. Non tutti purtroppo riescono e il mio pensiero andrà sempre a loro.

Chiudo con una frase di un poeta libanese, Khalil Gibran: "Nulla impedirà al sole di sorgere ancora, nemmeno la notte più buia. Perché oltre la nera cortina della notte c'è un'alba che ci aspetta".

Indice

One must acknowledge with cryptography no amount of violence will ever solve a math problem.
[J.Applebaum, Cypherpunks: Freedom and the Future of the Internet]

Capitolo 1

Preliminari

Lo scopo della crittografia è legato a doppio filo con la necessità di instaurare una comunicazione sicura tra due o più parti. Il più famoso cifrario del passato è senza dubbio il cifrario di Cesare, citato da Svetonio nella biografia di Giulio Cesare, che nonostante la semplicità evidenzia la necessità di nascondere un messaggio. La crittografia pertanto è stata sempre alla base dell'instaurazione di una comunicazione sicura con altre parti e con l'evoluzione dei calcolatori i crittosistemi sono diventati sempre più sicuri e resistenti ad eventuali attacchi e nel tempo i protocolli sono anche stati adattati per verificare l'autenticità di un messaggio. La storia della crittografia quindi si può riassumere come un susseguirsi continuo di scoperte matematiche e tecnologiche che ha portato a diverse innovazioni nell'ambito della cifratura dei messaggi.

1.1 Crittosistemi a chiave pubblica

Fino al 1976 tutti i crittosistemi erano basati su un processo chiamato **cifratura simmetrica**. Sia $\mathcal{P}$ lo spazio dei messaggi in chiaro M, sia $\mathcal{C}$ lo spazio dei messaggi cifrati C e sia $\mathcal{K}$ lo spazio delle chiavi k. Si definiscono due funzioni

$$Enc : \mathcal{K} \times \mathcal{P} \to \mathcal{C}$$
$$Enc(k, M) = C$$

(una volta che k è fissato si denota da qui in avanti con $Enc_k(M)$) e

$$Dec : \mathcal{K} \times \mathcal{C} \to \mathcal{P}$$
$$Dec(k, C) = M$$

(in analogia a prima da qui in avanti denotata con $Dec_k(C)$). Per cifrare un messaggio M è sufficiente scegliere una chiave $k \in \mathcal{K}$ e applicare la funzione Enc per ottenere un messaggio cifrato C. Al contrario, per decifrare un messaggio cifrato C è necessario conoscere la chiave k e applicare la funzione Dec per ottenere il messaggio originale M. È chiaro come qui sia presente un grande limite per iniziare la comunicazione con un soggetto nuovo: per scambiare un messaggio è necessario scambiarsi in modo sicuro la chiave k,

con il rischio che la chiave sia smarrita o trafugata. Questo limite rendeva impraticabile l'applicazione commerciale in larga scala dei crittosistemi fino ad allora costruiti. Inoltre un sistema di questo tipo è problematico nel caso di scambio di documenti sicuri tra n utenti, in quanto il numero di chiavi necessarie per uno scambio sicuro è $\frac{n^2}{2}$ che cresce velocemente all'aumentare di n.

L'anno di svolta dei crittosistemi è stato il 1976, anno in cui Diffie e Hellman [DH76] hanno proposto un modo innovativo per lo scambio di chiavi che è tuttora alla base dei moderni crittosistemi che sono chiamati **Crittosistemi a chiave pubblica**. La proposta di Diffie ed Hellman (che viene definita **Scambio di chiavi di Diffie Hellman**) consisteva nell'introdurre un cifrario **asimmetrico**, in cui ogni utente aveva a disposizione due chiavi, definite **chiave pubblica**(**public key** o pk) e **chiave privata**(**secret key** o sk) dove la prima viene pubblicata e usata per cifrare, mentre la seconda viene tenuta segreta e usata per decifrare i messaggi che son stati precedentemente cifrati con la chiave pubblica. Pertanto, prese queste due nuove chiavi $pk \in \mathcal{PK}$ e $sk \in \mathcal{SK}$ (in genere si ricava la chiave pubblica a partire dalla chiave privata) si definiscono due nuove funzioni Enc e Dec dove

$$Enc_{pk}(M) = C$$
$$Dec_{sk}(C) = M.$$

Si supponga che Bob voglia mandare un messaggio ad Alice usando il cifrario a chiave pubblica. Alice dovrà pubblicare la chiave pubblica che Bob utilizzerà per cifrare il suo messaggio con la funzione Enc_{pk}, mentre dovrà tenere segreta la chiave privata sk, che utilizzerà per decifrare il messaggio cifrato precedentemente con la funzione Dec_{sk}. Le funzioni Enc e Dec per essere utilizzabili devono essere computazionalmente efficienti, così come deve essere efficiente ricavare la chiave pubblica da quella privata mentre il viceversa deve essere computazionalmente infattibile. Questo sistema rivoluzionario è tuttora alla base dei moderni crittosistemi in quanto permette di scambiare la chiave pubblica su un canale non sicuro e rende più improbabile la perdita della chiave segreta, basando interamente la robustezza del sistema sulla difficoltà di ricavare la chiave privata a partire dalla chiave pubblica.

1.2 Crittosistema RSA

Il primo crittosistema a chiave pubblica è il crittosistema **RSA**, dalle iniziali di Rivest, Shamir e Adleman [RSA83], la cui sicurezza si basa sull'impossibilità di fattorizzare in modo efficiente un prodotto di interi primi. Per costruire un sistema RSA pertanto si inizia scegliendo due numeri primi *grandi* p e q e si calcola il numero $N = pq$. Si calcola inoltre $\varphi(N) = (p-1)(q-1)$, si sceglie un valore $e \in \mathbb{Z}_{\varphi(N)^*}$ e si calcola d tale che $ed \equiv 1 \mod \varphi(N)$. La chiave pubblica del crittosistema RSA è costituita dalla coppia (N, e) mentre la chiave privata è costituita dalla coppia $(\varphi(N), d)$.

Cifrare e decifrare un messaggio

Per cifrare un messaggio $M \in \mathbb{Z}_N$ è sufficiente usare la funzione

$$f_A(x) = x^e \mod N$$

Per decifrare un messaggio invece basta calcolare

$$f_A^{-1}(x) = x^d \mod N$$

È chiaro che questo sistema funziona grazie al Teorema di Eulero generalizzato e che il messaggio si può decifrare solo se l'utente conosce la chiave privata.

1.2.1 Sicurezza del sistema RSA

La sicurezza del sistema RSA si basa sul fatto che fattorizzare N non sia computazionalmente efficiente, pertanto basta utilizzare numeri sufficientemente *grandi* per ottenere un sistema inattaccabile mediante forza bruta. Non è tuttavia vero il viceversa, cioè non è noto se la rottura del crittosistema RSA sia equivalente alla fattorizzazione degli interi.

Il crittosistema non è tuttavia esente da alcune tipologie di attacchi e alcuni di questi sono l'attacco **broadcast**[Hås88], l'attacco **CCA** [Dav82], l'attacco **low private exponent** [Wie90] e l'attacco **low public exponent** [Cop97].

Attacco Broadcast

Questa tipologia di attacco considera una situazione in cui uno stesso messaggio viene inviato a più utenti. Si supponga che sia stato scelto e e che il messaggio venga inviato a k, con $k \geq e$ utenti. Si avranno pertanto k scelte di (N_i, e) e k diversi cifrati della forma

$$C_i = M^e \mod N_i.$$

Se l'attaccante riesce ad entrare in possesso di almeno e messaggi, allora grazie al teorema cinese del resto esiste un'unica soluzione all'equazione

$$C = M^e \mod \prod_{i=1}^{k} N_k.$$

Attacco Chosen Ciphertext

Si supponga che *Bob* voglia mandare un messaggio ad *Alice* con la sua chiave pubblica (N, e): *Bob* manderà un messaggio cifrato

$$C = M^e \mod N$$

Un ipotetico attaccante che intercetta questo messaggio può modificarlo scegliendo un valore R e creando

$$C_1 = CR^e = R^e M^e \mod N.$$

Immaginando che questo messaggio venga rispedito indietro decifrato si avrà che il nuovo messaggio in chiaro sarà

$$M_1 = MR \mod N.$$

All'attaccante non servirà far altro moltiplicare per R^{-1} per ottenere il messaggio in chiaro. Questo attacco non rompe propriamente RSA ma sfrutta una caratteristica della cifratura per attaccarlo. Nonostante tutto questa qualità viene utilizzata per dei protocolli di firma cieca.

Attacco low private exponent

Per la robustezza del sistema RSA è necessario anche prestare attenzione alla scelta dell'esponente privato e. Infatti, dato $N = pq$, se $q < p < 2p$ e vale che $d < \frac{1}{3}N^{\frac{1}{4}}$ allora è possibile calcolare efficientemente d a partire dalla chiave pubblica [Wie90].

Attacco low public exponent

L'ultimo attacco classico famoso al sistema RSA è certamente quello chiamato **low public exponent**, grazie a Coppersmith. Per rendere più rapida la codifica e la verifica di una generica firma di solito si sceglie per un valore di e non molto grande. Ricordando che la rottura del cifrario *RSA* corrisponde alla ricerca di una radice dell'equazione

$$x^e - C = 0 \mod N,$$

Coppersmith ha proposto un algoritmo efficiente per calcolare le radici minori di $N^{\frac{1}{e}}$ modulo N di un polinomio monico, come quello in questione. Da qui è diventato rapidamente di largo utilizzo $e = 65537$ come esponente pubblico nelle applicazioni più comuni di RSA che garantiva sia sicurezza che velocità (seppur più lento della scelta precedente di $e = 3$).

In meno di vent'anni dunque si sono susseguite diverse scoperte che hanno indebolito il sistema RSA senza tuttavia romperlo del tutto. Il cambio di passo è avvenuto qualche anno prima dell'attacco di Coppersmith, nel 1994, quando arrivò la prima proposta di algoritmo in grado di fattorizzare i numeri interi in un tempo polinomiale: fu quella di Shor [Sho94] ed è un algoritmo che per funzionare necessita di un **computer quantistico**.

1.3 Algoritmo di Shor

Arrivati a questo punto è chiaro che per rompere definitivamente RSA sia necessario riuscire a fattorizzare un intero in maniera efficiente e ad oggi il miglior algoritmo *classico* usato per fattorizzare numeri maggiori di 10^{100} è **GNFS** [BLP93] che comunque ha un tempo di esecuzione esponenziale, rendendo la fattorizzazione di chiavi a 2048 o 4096 bit computazionalmente infattibile. Inoltre è opportuno considerare che la scelta dei primi che vengono usati in RSA non può essere casuale ma anzi esistono precise indicazioni per la scelta degli stessi [KSD13].

In linea di principio si devono scegliere due numeri primi sufficientemente grandi (non è necessario che siano numeri primi forti, in quanto non offrono un beneficio sostanziale in termini di sicurezza [RS01]) in modo che sia difficile fattorizzare in tempi brevi: per RSA-2048 solitamente vengono usati due numeri primi di oltre 300 cifre.

Il cuore dell'algoritmo di Shor però, che è a tutti gli effetti la parte quantistica dello stesso, non tratta direttamente la fattorizzazione bensì fa riferimento a un risultato molto importante ottenuto nel 1976 da Miller [Mil76] che definiva una classe di problemi equivalente a quello della fattorizzazione: l'algoritmo di Shor rende efficiente il problema di trovare il periodo di una specifica funzione e come detto prima si può arrivare, sempre in modo efficiente, alla fattorizzazione di un intero.

Definizione 1 (Trovare il periodo di una funzione)**.** *Sia N un intero e sia $f(x)$ una funzione così definita*

$$f(x) = a^x \mod N.$$

Trovare il minimo intero $r > 0$ tale che

$$f(x + r) = f(x) \mod N$$

Avendo dunque a disposizione un algoritmo per calcolare in modo efficiente il periodo di una funzione si può ridurre il problema di fattorizzazione a quello di ricerca del periodo di una funzione. L'idea dietro alla soluzione di Miller è che dato un generico numero N bisogna scegliere un generico $x \mod N$, trovarne l'ordine r e calcolare $\text{MCD}(x^{\frac{r}{2}} - 1, n)$ (da qui in poi $\text{MCD}(a, b)$ sarà indicato come (a, b) salvo diversa specifica) ed è fattibile in un tempo polinomiale grazie all'algoritmo di Euclide. Visto che

$$(x^{\frac{r}{2}} - 1)(x^{\frac{r}{2}} + 1) = (x^r - 1) \equiv 0 \mod N$$

si ha che $(x^{\frac{r}{2}} - 1, n)$ è un divisore banale solo se r è dispari oppure se $(x^{\frac{r}{2}} \equiv -1 \mod N)$. Applicando questa procedura a una $x \mod N$ casuale si ottiene un fattore di N con una probabilità pari a $1 - 1/2^{k-1}$ dove k è il numero di primi che compone N.

La pubblicazione di Shor tuttavia non si ferma solamente alla fattorizzazione ma va oltre, descrivendo anche un algoritmo per rompere il problema del logaritmo discreto (fino al 1993 il miglior algoritmo per rompere il logaritmo discreto era quello di Gordon [Gor93]), alla base di sistemi crittografici come **El Gamal** e **DSA** e grazie all'esistenza dei pairing di curve ellittiche, anche i sistemi basati su queste ultime (uno su tutti **ECDSA**) sono a rischio.

Bisogna evidenziare tuttavia che allo stato attuale la potenza dei computer quantistici è molto limitata, specialmente a causa delle condizioni estreme a cui devono operare e della loro poca stabilità. Ciononostante il **NIST**[NIS17b] ha indetto una competizione per la ricerca di un nuovo algoritmo in grado di resistere a un computer quantistico e tra le proposte spiccano proprio i **crittosistemi multivariati**.

Capitolo 2

Crittosistemi a chiave pubblica multivariati

2.1 Il problema quadratico multivariato

L'algoritmo di Shor rende necessario basare i sistemi di crittografia a chiave pubblica su problemi che siano difficili da risolvere anche per un computer quantistico. Nel corso degli anni sono state diverse le alternative nella costruzione di un sistema più robusto e attualmente ci sono varie opzioni: una di queste consiste nella costruzione di un crittosistema a chiave pubblica che si basa su polinomi multivariati su un campo finito. La scelta del grado dei polinomi dev'essere sufficiente da rendere il sistema sufficientemente sicuro: dei polinomi lineari renderebbero la soluzione ai polinomi troppo semplice, grazie all'algoritmo di eliminazione gaussiana. Passando ai polinomi di secondo grado invece otteniamo quello che viene chiamato un **Problema Quadratico Multivariato**.

Definizione 2 (Il problema quadratico multivariato)**.** *Sia $\mathbb{F}_q$ un campo finito di q elementi e siano dati m polinomi quadratici $p_1, ..., p_m \in \mathbb{F}_q[X_1, .., X_n]$ in n variabili. Il problema consiste nel trovare la soluzione al sistema di equazioni*

$$p_i(X_1, ..., X_n) = 0 \quad per\ i = 1, ..., m. \tag{2.1}$$

Questo problema è noto per essere un problema NP-Completo (prima su $\mathbb{F}_2$ [GJ79] e poi su ogni campo [PG97]) e sembra che i computer quantistici non siano avvantaggiati nella sua risoluzione. Chiaramente il fatto che il problema sia di tipo NP-Completo non garantisce che non esistano polinomi per cui la soluzione al problema non sia efficiente anzi, si farà proprio uso di questo fatto per decrittare i messaggi in modo efficiente.

Molti dei sistemi di crittografia classica si basano sulla capacità di creare funzioni che sono poco efficienti da invertire, come il logaritmo discreto o il logaritmo discreto su curva ellittica. In un certo senso, anche con l'approccio multivariato si ha una soluzione simile ma si cerca di **mascherare** la funzione quadratica con altre funzioni che costituiscono la chiave segreta dell'utente.

Ci sono tre sottocategorie di sistemi multivariati: sistemi **bipolari**, sistemi **misti** e sistemi basati su **Isomorfismi di Polinomi**. La maggior parte dei crittosistemi multivariati si basa sulla prima tipologia.

2.2 Sistemi bipolari

Definizione 3 (Sistema bipolare)**.** *Sia K un campo finito. Si definisce* ***sistema bipolare*** *multivariato a chiave pubblica un sistema il cui cifrario è definito come una mappa $\overline{F}$ da K^n a K^m*

$$\overline{F}(x_1, ..., x_n) = (\overline{f}_1, ..., \overline{f}_m)$$

dove ogni $\overline{f}_i$ è un polinomio in $K[x_1, ..., x_n]$.

Per costruire un sistema di questo genere si parte innanzitutto da una mappa $F : K^n \to K^m$ tale che:

$$F(x_1, ..., x_n) = (f_1, ..., f_m) \text{ dove } f_i \in K[x_1, ..., x_n].$$

Ogni sistema

$$F(x_1, ..., x_n) = (y_1^*, ..., y_m^*)$$

dev'essere di facile risoluzione, o, equivalentemente, che sia computazionalmente efficiente trovare la preimmagine di $(y_1^*, ..., y_m^*)$, che sarà denotata come

$$F^{-1}(y_1^*, ..., y_m^*)$$

Una volta trovata la mappa F, bisogna scegliere casualmente due mappe $L_1 : K^m \to K^m$ e $L_2 : K^n \to K^n$ che siano funzioni affini invertibili. Il cifrario $\overline{F}$ si può costruire come composizione delle tre mappe fino a qui trovate:

$$\overline{F} := L_1 \circ F \circ L_2.$$

In un sistema di questo tipo, la chiave pubblica consiste nelle m-componenti del polinomio $\overline{F}$ e nel campo K, mentre la chiave segreta consiste nelle mappe L_1 e L_2. Si noti che non è necessario che la mappa F sia parte della chiave segreta, e questo dipenderà dal tipo di implementazione del sistema bipolare.
Criptare e decriptare un messaggio sfruttando questo metodo multivariato è facile. Si inizia scrivendo il messaggio X in forma $X' = (x'_1, ..., x'_n)$ e si calcola $\overline{F}'(X')$. Per decifrare un cifrato $Y' = (y'_1, ..., y'_m)$ bisogna risolvere il sistema di equazioni definito da

$$F(x_1, ..., x_n) = Y' \tag{2.2}$$

La scelta di mappe facilmente invertibili torna molto utile, in quanto per decifrare il messaggio basta applicare l'inverso della mappa vista precedentemente:

$$\overline{F}^{-1} := L_2^{-1} \circ F^{-1} \circ L_1^{-1}.$$

Per firmare un messaggio è sufficiente trovare una soluzione a 2.2 che si può denotare con $X' = (x_1, ..., x_n)$. Visto che la chiave pubblica è $\overline{F}$ si può calcolare

$$\overline{F}(x'_1, ...x'_n) = Y'$$

per verificare l'autenticità della firma.

Si può notare qui che la vera difficoltà in questo caso è data dalle due mappe, L_1 e L_2 che servono a mascherare la mappa F che, nella realtà, può essere facilmente invertita e non è necessario nemmeno che sia nascosta. La costruzione di $\overline{F}^{-1}$ è facile, a patto di essere a conoscenza delle mappe L_1 e L_2.

A livello teorico, la funzione F può essere qualsiasi polinomio di qualsiasi grado. Una costruzione di questo tipo però, esclude direttamente i polinomi di primo grado, in quanto si avrebbe a che fare con un sistema lineare che è invertibile in modo efficiente. Nei sistemi bipolari appena visti la chiave pubblica è solitamente costituita da polinomi di secondo grado in più variabili, visto che il problema quadratico multivariato è, generalmente, un problema NP-Completo. Sebbene sia possibile anche usare polinomi di grado superiore (del resto nella costruzione appena vista non ci sono limitazioni in tal senso) la scelta di polinomi quadratici deriva da considerazioni su sicurezza ed efficienza.

Innanzitutto, dato un campo K^n, il numero di termini possibili di un polinomio di grado d è dato dalla formula

$$\binom{n+d}{d} = \frac{(n+d)!}{n!d!}$$

e si può osservare come cresca molto rapidamente al crescere di d. Polinomi di grado molto alto pertanto portano inevitabilmente a una chiave pubblica molto grande, facendo decadere rapidamente l'efficienza del sistema. Inoltre ogni polinomio di grado superiore al secondo può essere scomposto in un sistema di polinomi di secondo grado aggiungendo ulteriori variabili: un esempio può essere il polinomio di terzo grado $xyz = 1$ che può essere scomposto in

$$\begin{cases} xy - t = 0 \\ tz = 1 \end{cases}$$

Pertanto, i benefici in termini di sicurezza potrebbero non giustificare il peggioramento di efficienza dovuto alla scelta di polinomi di grado superiore al secondo. I sistemi che verranno presentati, pertanto, faranno uso di polinomi di secondo grado, e saranno prettamente sistemi bipolari.

2.3 Schemi per firme di tipo Oil-Vinegar

Il primo gruppo di crittosistemi multivariati bipolari è quello degli schemi di tipo **Oil-Vinegar** [DGS06]. In questo gruppo ci sono tre famiglie di schemi che vengono utilizzati: il sistema **balanced Oil and Vinegar**, il sistema **unbalanced Oil-Vinegar** oppure **UOV** e **Rainbow**, un crittosistema multilivello che fa uso di UOV ad ogni livello. Mentre i primi due sistemi sono a rischio per alcuni difetti intrinsechi, il sistema Rainbow si è dimostrato molto sicuro, al punto che il suo algoritmo di firma digitale è arrivato al terzo

round della finale della competizione del NIST [NIS20] per la scelta di uno standard del futuro.

Alla base di un sistema di tipo Oil-Vinegar c'è il **polinomio Oil-Vinegar**, un polinomio quadratico in cui, quando vengono fissate le variabili **Vinegar**, si ottiene un polinomio di primo grado nelle variabili **Oil**.

Scegliendo un insieme di polinomi Oil-Vinegar si possono calcolare le variabili Oil per calcolare la firma. Da qui in avanti le variabili $(x_1, ..., x_o)$ rappresenteranno le variabili Oil e le variabili $(\tilde{x}_1, ..., \tilde{x}_v)$ rappresenteranno le variabili Vinegar.

Definizione 4 (Polinomio Oil-Vinegar)**.** *Sia K un campo finito di q elementi. Si definisce un polinomio Oil-Vinegar un polinomio $f \in K[x_1, ..., x_o, \tilde{x}_1, ..., \tilde{x}_v]$ nella forma*

$$f = \sum_{i=1}^{o}\sum_{j=1}^{v} a_{ij}x_i\tilde{x}_j + \sum_{i=1}^{v}\sum_{j=1}^{v} b_{ij}\tilde{x}_i\tilde{x}_j + \sum_{i=1}^{o} c_i x_i + \sum_{j=1}^{v} d_j\tilde{x}_j + e, \quad a_{ij}, b_{ij}, c_i, d_j, e \in K \tag{2.3}$$

Definizione 5 (Mappa Oil-Vinegar)**.** *Sia $F : K^n \to K^o$ una mappa polinomiale della forma*

$$F(x_1, ..., x_o, \tilde{x}_1, ..., \tilde{x}_v) = (f_1, ..., f_o)$$

dove $f_1, ..., f_o \in K[x_1, ..., x_o, \tilde{x}_1, ..., \tilde{x}_v]$ sono i polinomi Oil-Vinegar. La mappa F è definita come ***Mappa Oil-Vinegar****.*

Una proprietà relativa a questa mappa è che partendo da una selezione casuale di coefficienti di F e dato un vettore $(y_1^*, ..., y_o^*) \in K^o$, si può trovare la preimmagine di F con una selezione casuale $(\tilde{x}_1', ..., \tilde{x}_v')$ e risolvendo il sistema lineare nelle variabili Oil

$$F(x_1, ..., x_o, \tilde{x}_1, ..., \tilde{x}_v) = (y_1', ..., y_o').$$

È possibile che un sistema di questo tipo non ammetta soluzioni e la probabilità che questo avvenga tuttavia si può quantificare ed è circa $1-q^{-1}$. Se questo sistema non ha soluzione, si può iterare il processo fino a che non si trova una soluzione. Chiaramente, con una q sufficientemente grande e una scelta di $o + v$ sufficientemente piccola, la probabilità di riuscire al primo tentativo è certamente maggiore.

Fissate quindi $(y_1', ..., y_o') \in K^o$ e $(\tilde{x}_1', ..., \tilde{x}_v') \in K^v$ tale che esiste $(x_1', ..., x_o')$ che soddisfi

$$F(x_1', ..., x_o', \tilde{x}_1', ..., \tilde{x}_v') = (y_1', ..., y_o')$$

si può calcolare l'inversa di F valutata in $(y_1', ..., y_o')$, rispetto a $(\tilde{x}_1', ..., \tilde{x}_v')$:

$$F^{-1}(y_1', ..., y_o') = (x_1', ..., x_o').$$

E' interessante notare come la notazione suddetta non mostra la dipendenza della funzione dalla scelta $(\tilde{x}_1', ..., \tilde{x}_v') \in K^v$: l'unico interesse per queste variabili è che esista l'inversa della funzione su definita, per una scelta di $(\tilde{x}_1', ..., \tilde{x}_v') \in K^v$.

In quanto schema bipolare, dopo aver costruito la mappa centrale F, è necessario scegliere due funzioni affini atte a nascondere la mappa Oil-Vinegar che vanno composte con F. In questo caso, si sceglie $L_1 : K^o \to K^o$ dove L_1 è la funzione identità, mentre

$L_2 : K^{o+v} \to K^{o+v}$ è una generica funzione affine. Il risultato sarà una mappa quadratica $\overline{F} : K^{o+v} \to K^o$ definita da

$$\overline{F} = F \circ L_2 = (\overline{f}_1, ..., \overline{f}_o).$$

La scelta casuale dei coefficienti della mappa Oil-Vinegar rende superfluo scegliere una funzione L_1 con cui comporre la funzione a sinistra per mascherare la mappa.

2.3.1 Firmare con Oil-Vinegar

Chiave pubblica

La chiave pubblica di uno schema Oil-Vinegar è una coppia costituita dal campo K (con relativa struttura) e la mappa $\overline{F}$ o, equivalentemente, le sue componenti $\overline{f}_1, ..., \overline{f}_o \in K[z_1, ..., z_n]$ dove $n = o + v$.

Chiave privata

La chiave privata di uno schema Oil-Vinegar, invece, è una coppia costituita dalla trasformazione affine $L : K^n \to K^n$ e dalla mappa F o, equivalentemente, le sue componenti $f_1, ..., f_o \in K[x_1, ..., x_o, \tilde{x}_1, ..., \tilde{x}_v]$.

Un'osservazione importante è che in questo crittosistema parte della sicurezza dipende dal fatto che la funzione F non è nota al pubblico.

Generazione della firma

Sia $(y'_1, ..., y'_o) \in K^o$ il documento da firmare. Innanzitutto il firmatario calcola

$$(x'_1, ..., x'_o) = F^{-1}(y'_1, ..., y'_n)$$

per una scelta casuale di $(\tilde{x}'_1, ..., \tilde{x}'_v) \in K^v$ ricordando che F^{-1} qui non è propriamente l'inversa bensì la preimmagine. Avendo fissato le variabili Vinegar, pertanto, si ha un sistema lineare nelle variabili Oil che è di facile risoluzione

$$F(x_1, ..., x_o, \tilde{x}'_1, ..., \tilde{x}'_v) = (y'_1, ..., y'_o).$$

Risolto questo problema, la firma relativa a $(y'_1, ..., y'_o)$ sarà

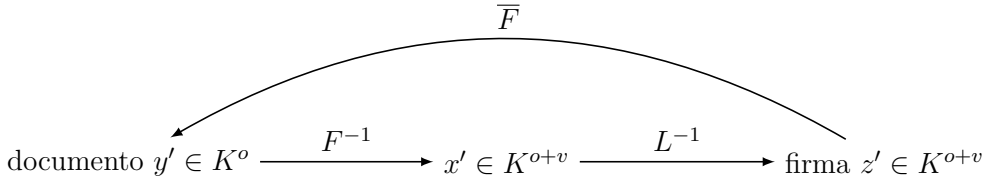
$$(z'_1, ..., z'_n) = L^{-1}(x'_1, ..., x'_o, \tilde{x}'_1, ..., \tilde{x}'_v)$$

Verifica della firma

Per verificare l'autenticità della firma $(z'_1, ..., z'_n)$, è sufficiente che il destinatario verifichi che valga la seguente uguaglianza

$$\overline{F}(z'_1, ..., z'_n) = (y'_1, ..., y'_o).$$

Il calcolo e la verifica di una firma con Oil-Vinegar può essere riassunto in questo grafico:



A questo punto si può mostrare un esempio basilare che permette di chiarire il funzionamento di un sistema di tipo Oil-Vinegar. Per questo esempio si sceglierà il campo $K = \mathbb{F}_7$. La scelta del numero di variabili Oil-Vinegar determina la tipologia del sistema: se $v = o$ si ha un sistema bilanciato. Per l'esempio si scelgono $v = o = 2$ e quindi $n = 4$, in modo da semplificarne l'illustrazione (per applicare questo crittosistema si scelgono valori più grandi). Sia quindi $(x_1, x_2, \tilde{x}_1, \tilde{x}_2)$ il vettore di variabili Oil-Vinegar. Si possono costruire, pertanto, i polinomi Oil-Vinegar: un modo rapido per rappresentarli è scriverli in forma matriciale come $f_i = x' Q_i x + Px + C$. Si pongano per semplicità $P = 0$ e $C = 0$ mentre si sceglie Q a piacere: le variabili Oil non devono apparire in grado superiore al primo, quindi qui è necessario che $Q_{i,11} = Q_{i,12} = Q_{i,21} = Q_{i,22} = 0$.

$$Q_1 = \begin{pmatrix} 0 & 0 & 1 & 4 \\ 0 & 0 & 1 & 0 \\ 6 & 1 & 2 & 6 \\ 3 & 3 & 0 & 1 \end{pmatrix}, \quad Q_2 = \begin{pmatrix} 0 & 0 & 3 & 6 \\ 0 & 0 & 2 & 1 \\ 4 & 2 & 0 & 1 \\ 2 & 6 & 5 & 1 \end{pmatrix}.$$

Il risultato sarà:

$$\begin{aligned} f_1 &= 2\tilde{x}_1^2 + \tilde{x}_2^2 + 2x_2\tilde{x}_1 + 3x_2\tilde{x}_2 + 6\tilde{x}_1\tilde{x}_2, \\ f_2 &= \tilde{x}_2^2 + x_1\tilde{x}_2 + 4x_2\tilde{x}_1 + 6\tilde{x}_1\tilde{x}_2. \end{aligned}$$

Si sceglie ora una trasformazione L invertibile e, per semplicità, lineare (ricordando che può essere una funzione affine). Si avrà quindi

$$x = Lz,$$

dove x è il vettore dato prima, $z := (z_1, z_2, z_3, z_4)'$ e L è

$$\begin{pmatrix} 4 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 \\ 2 & 5 & 1 & 0 \\ 3 & 0 & 1 & 1 \end{pmatrix}.$$

Da qui si possono calcolare i coefficienti dei polinomi pubblici in questo modo:

$$\overline{f_i} = z' Q_i z = x'(L' Q_i L)x = x' \overline{Q}_i x.$$

Pertanto $\overline{Q}_i$ diventano:

$$\overline{Q}_1 = \begin{pmatrix} 3 & 0 & 5 & 0 \\ 6 & 4 & 3 & 2 \\ 2 & 3 & 2 & 0 \\ 6 & 6 & 6 & 4 \end{pmatrix} \quad \overline{Q}_2 = \begin{pmatrix} 2 & 6 & 3 & 3 \\ 0 & 6 & 6 & 1 \\ 5 & 0 & 2 & 0 \\ 2 & 0 & 1 & 2 \end{pmatrix}$$

ed è chiaro che conoscendo L sia facile risalire a Q.

Si supponga di aver già calcolato l'Hash di un documento e aver diviso il digest in un messaggio $M = (m_1, m_2) = (1, 1)$ con una firma $S = (s_1, ..., s_4)$ che verifichi $\overline{F}(M) = S$. Si inizia scegliendo 2 variabili Vinegar casualmente e una possibile scelta è

$$(\tilde{x}_1, \tilde{x}_2) = (2, 3).$$

Sostituendo queste variabili nei polinomi Oil-Vinegar si ottengono dei polinomi che, come detto prima, sono lineari:

$$f_1(x_1, x_2,2,3) = 6x_2 + 4,$$
$$f_2(x_1, x_2,2,3) = 3x_1 + x_2 + 3.$$

Fissando $f_i(x_1, x_2, 2, 1) = m_i$ si ottiene il sistema lineare

$$\begin{cases} 4x_1 + 2x_2 + 4 & = 1 \\ 3x_1 + x_2 + 3 & = 1 \end{cases}$$

che ha soluzione

$$(x_1, x_2) = (3, 3).$$

Pertanto, l'accoppiata Oil-Vinegar sarà

$$(x_1, x_2, \tilde{x}_1, \tilde{x}_2) = (3,3,2,3).$$

La firma si potrà calcolare nel modo seguente:

$$S = L^{-1}(3,3,1,1) = (3,5,6,2).$$

Per verificare la firma basterà pertanto calcolare la funzione $\overline{F}$ definita prima:

$$\overline{F}(3,5,6,2) = (1,1).$$

Questo piccolo esempio mostra come costruire una firma basandosi sullo schema Oil-Vinegar. L'assunzione fatta all'inizio è che $o = v$ cioè che il sistema fosse bilanciato. I sistema di tipo Oil-Vinegar bilanciato sono però soggetto a un attacco mostrato da Kipnis e Shamir nel 1998 [KS98]. Si può ricorrere pertanto a un sistema chiamato **unbalanced Oil-Vinegar** o **UOV** scegliendo diverse dimensioni di v e o. E' stato mostrato che $v \gtrsim o^2$ non è una scelta sicura, in quanto ci sono algoritmi efficienti per trovare le soluzioni di sistemi quadratici fortemente sottodeterminati. Pertanto, una scelta ottimale per la robustezza di questo sistema, è scegliere $v \simeq 3o$ [KPG99]. Un sistema di firma che fa grande uso dello schema UOV è Rainbow, che verrà trattato a parte.

2.4 Hidden Field Equations

Un altro crittosistema che rientra nella categoria di sistemi bipolari è quello delle **Hidden Field Equations** o **HFE** [Pat96]. Sia $\mathbb{F}_{q^n}$ un'estensione del campo $\mathbb{F}_q$. Se $g(x) \in \mathbb{F}[x]$ è un polinomio irriducibile di grado n, allora si può dire che $\mathbb{F}_{q^n} \cong \mathbb{F}_q[x]/g(x)$. Si può pertanto definire un isomorfismo $\phi : \mathbb{F}_{q^n} \to \mathbb{F}_q^n$ nel modo seguente:

$$\phi\left(\sum_{i=0}^{n-1} a_i X^i\right) := (a_0, ..., a_{n-1}). \tag{2.4}$$

Per costruire un sistema HFE si inizia definendo una mappa $\tilde{F}$ nel modo seguente

$$\tilde{F}(x) = \sum_{i=0}^{r_2-1} \sum_{j=0}^{i} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{r_1-1} b_i X^{q^i} + c \tag{2.5}$$

dove i coefficienti a_{ij}, b_i e c sono scelti in modo casuale, mentre r_1 e r_2 devono essere scelti in modo che il grado di $\tilde{F}$ sia d. Da qui si può pertanto definire la mappa centrale F

$$F := \phi \circ \tilde{F} \circ \phi^{-1}.$$

Il seguente teorema garantisce che saranno presenti al più termini $(x_1, ..., x_n)$ quadratici:

Teorema 1. *Sia $\mathbb{F}_q$ un campo, $g(X) \in \mathbb{F}_q[X]$ un polinomio irriducibile di grado n, $\mathbb{F}_{q^n} \cong F_q[X]/g(X)$ l'estensione di campo associata e sia $p(X) = \sum_{i=0}^{n-1} f_i(x_1, ..., x_n)X^i$ un generico polinomio $p(X) \in \mathbb{F}_q[X]$, con dove $f_i : \mathbb{F}_q^n \to \mathbb{F}_q$. Allora, un polinomio $F \in \mathbb{F}_q[X]$ così definito*

$$F(X) := \sum_{i=0}^{r_2-1} \sum_{j=0}^{i} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{r_1-1} b_i X^{q^i} + c$$

se valutato in $p(X)$ avrà coefficienti che saranno al più quadratici in $f_i(x_1, ...x_n)$.

Dimostrazione. Per un generico campo finito di cardinalità q, vale sempre che

$$a^q \equiv a.$$

Per dimostrare la tesi, è sufficiente dimostrare che X^{q^i} valutata in una generica funzione $(f_0(x_1, ..., x_n) + f_1(x_1, ..., x_n)x + ... + f_{n-1}(x_1, ..., x_n)x^{n-1})$ sia un polinomio, di grado al più $q^i(n-1)$, in cui coefficienti devono essere al più combinazioni lineari di f_i. Per provare questo si calcola $(f_0(x_1, ..., x_n)+f_1(x_1, ..., x_n)x+...+f_{n-1}(x_1, ..., x_n)x^{n-1})^{q^i}$. L'espansione di un generico polinomio multivariato si può scrivere come

$$(x_1 + ... + x_m)^n = \sum_{k_1+...+k_m=n} \binom{n}{k_1, ..., k_m} \prod_{i=1}^{m} x_i^{k_i}$$

Si può osservare tuttavia che nel caso in questione $n = q^i$ e scrivendo

$$(x_1 + ... + x_m)^n = (((x_1 + ... + x_m)^q)^{\cdot\cdot})^q$$

si ha che, applicando in seguito quanto visto all'inizio:

$$(x_1 + ... + x_m)^q = (x_1^q + ... + x_m^q) = (x_1 + ... + x_m).$$

Pertanto si potrà scrivere

$$\left(\sum_{k=0}^{n-1} f_k(x_1, ..., x_n)x^k\right)^{q^i} = \sum_{k=0}^{n-1} f_k(x_1, ..., x_n)x^{q^i k}$$

□

In quanto un sistema bipolare, si avrà una funzione del tipo

$$\overline{F} = L_1 \circ F \circ L_2$$

dove la chiave privata sarà costituita da F, L_1 e L_2, con L_1 e L_2 funzioni affini invertibili.

2.4.1 Cifrare con HFE

Chiave pubblica

La chiave pubblica di un sistema HFE sarà costituita dal campo $\mathbb{F}$ (con relativa struttura) e dalla mappa $\overline{F}$, o, equivalentemente, dai polinomi di secondo grado $\overline{f}_1(x_1, ..., x_n), ..., \overline{f}_n(x_1, ..., x_n) \in K[x_1, ..., x_n]$.

Chiave privata

La chiave privata sarà costituita dalle funzioni $\tilde{F}$ e dalle trasformazioni affini L_1 e L_2. Per come è stata costruita $\tilde{F}$, si ritiene che sia praticamente impossibile indovinarla a partire da $\overline{F}$.

Cifratura

Per cifrare un messaggio M si procede innanzitutto scegliendo una funzione hash e calcolando $r := H(M)$. Si scrive $r = (r_1, ..., r_n)$ e si calcola

$$e = \overline{F}(r_1, ..., r_n).$$

Si procede a pubblicare quindi e che sarà il messaggio cifrato.

Decifratura

Dato un messaggio cifrato $(y'_1, ..., y'_n)$ si può decifrare il messaggio del modo seguente

- calcolare $(\overline{y}_1, ..., \overline{y}_n) = L_1^{-1}(y'_1, ..., y'_n)$;

- calcolare $\overline{Y} = \phi^{-1}(\overline{y}_1, ..., \overline{y}_n)$ e trovare le soluzioni all'equazione

$$\tilde{F}(\tilde{x}_i) = \overline{Y}. \tag{2.6}$$

  Per questo passaggio esiste un algoritmo chiamato algoritmo di **Berlekamp** [Ber67] che permette un'efficiente risoluzione del sistema di equazioni. L'efficienza dell'algoritmo dipende dal grado polinomiale scelto per $\tilde{F}$ e questo sarà visibile nelle tre declinazioni che vengono offerte dallo schema di firma GeMSS. E' opportuno osservare che l'equazione suddetta può avere molteplici soluzioni.

- Calcolare

$$M_i = (m_{i,1}, ..., m_{i,n}) = L_2 \circ \phi(\tilde{x}_i);$$

- calcolare infine l'hash dei vari messaggi e si otterranno $r_i = H(M_i)$. Il messaggio originario sarà quello tale che $r_i = r$.

Sulla falsariga dello schema UOV, ecco un esempio di cifratura con lo schema HFE.
Si parte innanzitutto con la scelta di un campo che in questo caso può essere $\mathbb{Z}_5$ e con la scelta di un polinomio irriducibile, che in questo caso è

$$g(x) = x^3 + x + 1.$$

Essendo un polinomio di terzo grado è sufficiente verificare che non abbia radici in $\mathbb{F}_5$ per assicurarne l'irriducibilità, pertanto si ha un'estensione di grado $n = 3$. Si scelgono inoltre due matrici L_1 e L_2 che serviranno a mascherare la mappa centrale:

$$L_1 = \begin{pmatrix} 1 & 0 & 3 \\ 4 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad L_2 = \begin{pmatrix} 3 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 2 & 1 \end{pmatrix}$$

E' necessario inoltre scegliere un polinomio $\tilde{F}(x)$ che sarà parte integrante della mappa centrale. Si possono scegliere $r_2 = 2$ e $r_1 = 1$ che in questo caso portano a un polinomio con $d = 10$. Pertanto, una possibile $\tilde{F}(x)$ può essere la seguente:

$$\tilde{F}(x) = x^{10} + 3x^6 + 2x + 1.$$

Ci sono quindi tutti gli ingredienti necessari per costruire uno schema di tipo HFE. Si inizia calcolando la funzione $\phi^{-1} \circ L_2$ relativa al messaggio in chiaro che viene identificato come (x_1, x_2, x_3)

$$\phi^{-1} \circ L_2 = (3x_1 + x_3) + (x_1)x + (2x_2 + x_3)x^2.$$

Nell'estensione di campo scelta, è chiaro come

$$\tilde{F}(x) = x^{10} + 3x^6 + 2x + 1 \equiv x^2 + x + 2$$

Pertanto si avrà che

$$\begin{aligned} \tilde{F} \circ \phi^{-1} \circ L_2 = {} & 4x_1^2 + x_3^2 + 4x_1x_3 + x_1x_2 + 3x_1 + x_3 + 2 + \\ & (x_1^2 + x_2^2 + 4x_3^2 + x_1x_2 + x_2x_3 + x_1)x + \\ & (x_1^2 + x_2^2 + x_3^2 + 2x_1x_2 + x_1x_3 + 2x_2 + x_3)x^2. \end{aligned}$$

Applicando infine $L_1 \circ \phi$ si ottengono i seguenti polinomi pubblici che costituiscono la chiave $\overline{F}$:

$$\overline{f}_1(x_1, x_2, x_3) = 2x_1^2 + 3x_2^2 + 4x_3^2 + 2x_1x_2 + 2x_1x_3 + 3x_1 + x_2 + 4x_3 + 2$$
$$\overline{f}_2(x_1, x_2, x_3) = 3x_1^2 + 2x_2^2 + 2x_3^2 + x_1x_2 + x_1x_3 + 2x_2x_3 + 4x_1 + 4x_3 + 3$$
$$\overline{f}_3(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2 + 2x_1x_2 + x_1x_3 + 2x_2 + x_3.$$

Si voglia ora cifrare il messaggio la cui funzione hash scelta ha immagine (1,1,1). La persona in possesso della chiave pubblica non dovrà far altro che valutare la funzione $\overline{F} = (\overline{f}_1, \overline{f}_2, \overline{f}_3)$ nel messaggio, ottenendo:

$$\overline{F}(1,1,1) = (3,2,4).$$

Si supponga ora di aver ricevuto il suddetto messaggio cifrato. Per decifrarlo si inizia calcolando $(\overline{y}_1, ..., \overline{y}_n) = L_1^{-1}(y'_1, ..., y'_n)$ che porta a

$$(\overline{y}_1, ..., \overline{y}_n) = (1,4,4)$$

che, applicando ϕ^{-1}, diventa $1 + 4x + 4x^2$. A questo punto si cercano le soluzioni dell'equazione

$$X^{10} + 3X^6 + 2X + 1 = 1 + 4x + 4x^2.$$

Si ottengono due soluzioni:

$$Y = \{4x + 2x^2, 4 + x + 3x^2.\}$$

Applicando infine la funzione $L_2^{-1} \circ \phi$ alle due soluzioni si ottengono due messaggi

$$M = \{(4,1,0), (1,1,1)\}$$

da cui si evince come solo uno sia il messaggio giusto. Aggiungendo una ridondanza al messaggio si può fare in modo di riconoscere immediatamente il messaggio giusto.

2.4.2 Firmare con HFE

Sebbene sia possibile cifrare con HFE come visto sopra, gli algoritmi basati su HFE sono solitamente sfruttati per degli schemi di autenticazione. Si supponga di aver già definito, come nel processo di cifratura, la chiave privata $(\tilde{F}, L_1, L_2)$.

Generazione della firma

Per generare la firma si calcola innanzitutto bisogna calcolare la funzione hash del messaggio da firmare. Si suppone che la funzione hash restituisca un digest $h = (h_1, ..., h_b) \in \mathbb{F}^b$ dove $b \leq n$, si scelga un padding $r = (r_{b+1}, ..., r_n) \in \mathbb{F}^{n-b}$ e si definisca quindi $y' = H||R$. Il processo di firma consiste nell'eseguire il processo eseguito prima a ritroso.

- Si calcola $\overline{y} = L_1^{-1}(y'_1, ..., y'_n)$.

- Si cerca $\overline{x}$ tale $F(\overline{x}) = \overline{y}$. Qui è opportuno prestare particolare attenzione, in quanto per com'è stato definito il polinomio centrale (F non è, in generale, suriettiva) può non ammettere soluzione. Nel caso non esista soluzione, si cambia la stringa concatenata R fino a che non si trova una soluzione. La probabilità di trovare $\overline{x}$ in questo caso dipende dai campi scelti [Pat96].
- Si calcola $z' = L_2^{-2}(\overline{x}_1, ..., \overline{x}_n)$.
- La firma associata è

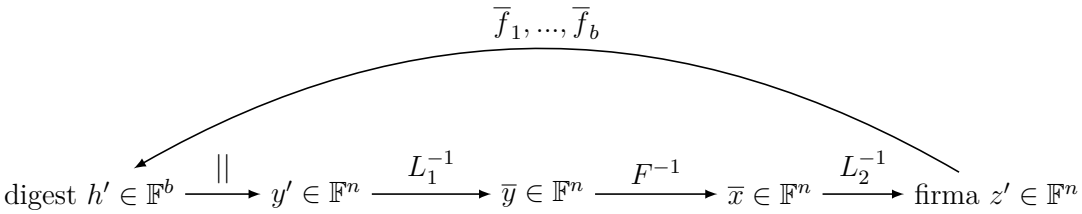
$$\Big((\overline{f}_1(x_1, ..., x_n), ..., \overline{f}_b(x_1, ..., x_n)), x\Big)$$

  dove i polinomi $\overline{f}_i$ per $i = 1, ..., b$ sono i polinomi HFE definiti nel processo di cifratura.

Verifica della firma

Per verificare l'autenticità della firma è sufficiente valutare i polinomi pubblici relativi alla firma in x e calcolare $(\overline{f}_1, ..., \overline{f}_b)(x) = h'$. Se $h' = h$ allora il documento è autentico.

Si può riassumere di generazione e verifica della firma del digest di un documento con HFE nel seguente grafico:



2.4.3 Varianti di HFE

Lo schema di cifratura di HFE come illustrato fino ad ora non è sicuro ed esistono diversi algoritmi di attacco: un esempio su tutti è quello di Faugére e Joux [FJ03] che rompe efficientemente lo schema sfruttando le basi di Gröbner. Esistono tuttavia alcune variazioni di HFE che sono resistenti agli attacchi.

HFE$^-$ e HFE$^+$

Le prime variazioni su HFE che si possono realizzare sono quelle dette **plus** [Pat96] e **minus**[Sha94] e si definiscono **HFE$^-$** e **HFE$^+$** che consistono nell'aggiungere o togliere polinomi dalla chiave pubblica F. Nel primo caso si parte dai polinomi pubblici di HFE e si eliminano u polinomi. Si definisce pertanto una nuova chiave pubblica $\overline{F}^- : \mathbb{F}_q^n \to \mathbb{F}_q^l$ dove $l := n - u$, mantenendo segreti gli ultimi u polinomi. Se nel messaggio è presente un certo grado di ridondanza, allora sarà possibile ritrovare i messaggi originali. Per quanto riguarda la variante HFE$^+$, si supponga di partire da una funzione $\overline{F} : \mathbb{F}_q^n \to \mathbb{F}_q^l$, scegliendo i polinomi $p_1, ..., p_s \in \mathbb{F}_q^n[x_1, ..., x_n]$ si definisce la funzione $\overline{F}^+ : \mathbb{F}_q^n \to \mathbb{F}_q^{l+s}$ aggiungendo

in coda i polinomi scelti e mascherando il tutto con una funzione L_3 affine invertibile. Si avrà pertanto:

$$\overline{F}^{+} = L_3 \circ (\overline{f}_1, ..., \overline{f}_l, p_1, ..., p_s).$$

E' opportuno notare che è possibile utilizzare insieme plus e minus per creare uno schema di tipo **plus-minus**.

HFEv e HFEv^{-}

Molto più interessante è invece la combinazione di HFE con lo schema Oil-Vinegar nello schema **HFEv**. Si riprende lo schema come visto prima, a cui si aggiungono delle variabili di tipo Vinegar. Si cambia quindi la funzione di partenza, che ora è $\tilde{F} : K \times \mathbb{F}_q^v \to K$

$$\tilde{F}(x, z_1, ..., z_v) = \sum_{i=0}^{r_2-1} \sum_{j=0}^{i} a_{ij} x^{q^i+q^j} + \sum_{i=0}^{r_1-1} b_i(z_1, ..., z_v) x^{q^i} + c(z_1, ..., z_v).$$

Le variabili appena introdotte sono scelte casualmente, mentre il processo di cifratura e decifratura resta invariato. La scelta delle nuove funzioni dipendenti dalle variabili vinegar deve essere tale per cui la chiave pubblica sia un polinomio di secondo grado. Pertanto è necessario che la funzione $b_i(z_1, ..., z_v)$ sia al più un polinomio di primo grado mentre la funzione $c_i(z_1, ..., z_v)$ sia al più un polinomio di secondo grado. Questa modifica al protocollo HFE tuttavia non porta alcun vantaggio in termini di sicurezza [DS05a] mentre abbinando la modificazione con le variabili Vinegar al processo minus prima introdotto si ottiene un sistema sicuro. Questo sarà alla base del crittosistema GeMSS.

Capitolo 3

Rainbow

Nel tentativo di creare un crittosistema ancora più robusto ed efficiente (necessario per implementazioni su dispositivi di basso costo), Ding e Schmidt hanno proposto nel 2005 un nuovo crittosistema che si può definire una generalizzazione di UOV ed è stato chiamato **Rainbow** [DS05b] e si può considerare come una versione a più livelli di UOV. La robustezza di questo sistema è tale che ad oggi è arrivato al terzo e ultimo round come schema di firma digitale nella competizione del NIST [NIS20] per la scelta di uno standard di firma nell'ambito della crittografia post-quantistica.

3.1 Costruzione dello schema Rainbow

Per costruire un sistema di tipo Rainbow si inizia innanzitutto definendo un insieme V nel seguente modo

$$V := \{1, ..., n\}.$$

Siano $v_1, ..., v_u$ u interi tali che $0 < v_1 < v_2 < ... < v_u = n$ e siano $V_l := \{1, .., v_l\}$ per $l = 1, ..., u$, in modo che

$$V_1 \subset V_2 \subset ... \subset V_u = V.$$

Dalla definizione il numero di elementi in V_i sarà v_i. Si definiscono le variabili x_j dove $j \in V_l$ come variabili **Vinegar**. Si definiscano ora i valori o_i tali che

$$o_i = v_{i+1} - v_i \text{ per } i = 1, ..., u - 1$$

e gli insiemi O_i tali che

$$O_i = S_{i+1} - S_i \text{ per } i = 1, ..., u - 1.$$

Da qui si possono definire le variabili **Oil** x_i dove $i \in O_l$.

Si può quindi costruire lo spazio dei polinomi P_l di tipo **Oil-Vinegar** che avranno struttura del tutto simile a quella già incontrata precedentemente:

$$P_l := \sum_{i \in O_l, j \in S_l} \alpha_{i,j} x_i x_j + \sum_{i,j \in V_l} \beta_{i,j} x_i x_j + \sum_{i \in V_{l+1}} \gamma_i x_i + \eta.$$

Si definisce un polinomio in P_l come **polinomio Oil-Vinegar di livello** l e si osserva che $P_i \subset P_j$ per $i < j$. Si può ora costruire la mappa centrale F relativa allo schema di firma Rainbow. Sia $\mathbb{F}_q^n$ un campo, si definisce la mappa $F : \mathbb{F}_q^n \to \mathbb{F}_q^{n-v_1}$ tale che

$$\begin{aligned} F(x_1, .., x_n) &= (\tilde{F}_1(x_1, ..., x_n), ..., \tilde{F}_{u_1}(x_1, ..., x_n)) \\ &= (\check{F}_1(x_1, ..., x_n), ..., \check{F}_{n-v_1}(x_1, ..., x_n)) \end{aligned}$$

dove ogni $\tilde{F}_i$ consiste in o_i polinomi scelti $\check{F}_1, ..., \check{F}_{o_i}$. L'unica cosa rimanente da scegliere sono i coefficienti dei polinomi, che in questo caso vengono scelti in modo casuale. Lo schema chiaramente ha $u - 1$ livelli di costruzioni di tipo Oil-Vinegar. Il primo strato è costituito da o_1 polinomi $\check{F}_1, ..., \check{F}_{o_i}$ dove $x_i,\ i \in O_1$ sono le variabili Oil e $x_j,\ j \in V_1$ sono le variabili Vinegar. All'i-esimo livello ci saranno o_i polinomi, $\check{F}_{v_i+1}, ..., \check{F}_{v_{i+1}}$ e via discorrendo. A questo punto si avranno tante variabili, *a rainbow of variables*, Oil e Vinegar ad ogni livello:

- Al primo livello si avrà:

  $$\begin{aligned} &\text{Variabili Oil: } \{x_{v_1+1}, ..., x_{v_2}\} \\ &\text{Variabili Vinegar: } \{x_1, ..., x_{v_1}\} \end{aligned}$$

- Al secondo livello si avrà:

  $$\begin{aligned} &\text{Variabili Oil: } \{x_{v_2+1}, ..., x_{v_3}\} \\ &\text{Variabili Vinegar: } \{x_1, ..., x_{v_1}, x_{v_1+1}, ..., x_{v_2}\} \end{aligned}$$

- E infine al livello i-esimo si avrà:

  $$\begin{aligned} &\text{Variabili Oil: } \{x_{v_i+1}, ..., x_{v_{i+1}}\} \\ &\text{Variabili Vinegar: } \{x_1, ..., x_{v_1}, x_{v_1+1}, ..., x_{v_2}, ..., x_{v_{i-1}+1}..., x_{v_i}\} \end{aligned}$$

si definisce infine la funzione F un polinomio **Rainbow a** $u - 1$ **livelli**. È evidente come il sistema Rainbow a 1 livello sia corrispondente a uno schema Oil-Vinegar.
Per la costruzione dello schema, in quanto sistema bipolare, avendo definito la mappa centrale F, è sufficiente definire due mappe lineari o affini $L_1 : \mathbb{F}_q^{n-v_1} \to \mathbb{F}_q^{n-v_1}$ e $L_2 : \mathbb{F}_q^n \to \mathbb{F}_q^n$. Infine si definisce con $m = n - v_1$ il numero di equazioni.

3.2 Generazione della chiave

3.2.1 Chiave privata

Come in ogni sistema bipolare, le mappe affini L_1 e L_2 fanno parte della chiave privata. Alle suddette si aggiunge la mappa centrale $F : \mathbb{F}_q^n \to \mathbb{F}_q^m$ che consiste in m polinomi multivariati del tipo

$$\check{F}_k(x_1, ..., x_n) = \sum_{\substack{i,j \in V_l, \\ i \leq j}} \alpha_{k,ij} x_i x_j + \sum_{\substack{i \in V_l, \\ j \in O_l}} \beta_{k,ij} x_i x_j + \sum_{i \in V_{i+1}} \gamma_{k,i} x_i + \eta_k.$$

dove $k = v_1+1, ..., n$. E' importante notare che per ogni k esiste una sola l, per costruzione, tale che $k \in O_l$, dunque ogni polinomio può appartene solo ad uno specifico livello. I coefficienti $\alpha_{k,ij}$, $\beta_{k,ij}$, $\gamma_{k,i}$ e η sono invece scelti dal campo $\mathbb{F}_q$. La dimensione della chiave privata in bit è data da

$$\underbrace{\frac{m(m+1)}{\text{mappa } L_1}}_{\text{mappa } L_1} + \underbrace{\frac{n(n+1)}{\text{mappa } L_2}}_{\text{mappa } L_2} + \overbrace{\sum_{i=1}^{u}\left(\frac{v_i(v_i+1)}{2} + v_i o_i + v_i + o_i + 1\right)}^{\text{mappa } F}.$$

moltiplicata per $\log_2 q$.

3.2.2 Chiave pubblica

La chiave pubblica, in modo del tutto analogo a quanto definito per un generico sistema bipolare, sarà la mappa $\overline{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ dove

$$\overline{F} := L_1 \circ F \circ L_2$$

che non saranno altro che m polinomi quadratici nel campo F_q^n.
Pertanto la dimensione della chiave pubblica in bit sarà

$$\frac{m(n+1)(n+2)}{2}$$

moltiplicato per $\log_2 q$.

3.3 Generazione della firma

Per generare la firma di un generico documento d bisogna innanzitutto scegliere una funzione hash $\mathcal{H} : \{0,1\}^p \to \mathbb{F}^m$ dove p è la lunghezza di d. Se ne calcola la funzione hash $h := \mathcal{H}(d) \in \mathbb{F}^m$. Ricordando che Rainbow è un sistema bipolare, quindi

$$\overline{F} := L_1 \circ F \circ L_2,$$

si calcola

$$y = L_1^{-1}(h_1, ..., h_m).$$

Una volta calcolato y, si procede in maniera del tutto analoga a quanto fatto per lo schema Oil-Vinegar. Si scelgono innanzitutto valori casuali per le variabili $x_1, ..., x_{v_1}$ e si sostituiscono le stesse all'interno del polinomio $\check{F}_i$. Per ogni livello $l : 1, .., u$ si risolvono, con eliminazione gaussiana, il polinomio $\check{F}_i$, $i \in O_l$ che è sempre lineare nelle variabili Oil. Si sostituiscono quindi le variabili nei vari polinomi $\check{F}_i$ e iterando per ogni livello (ricordando che passando al livello successivo, le precedenti variabili Oil diventano le nuove variabili Vinegar, avendo quindi a che fare con la risoluzione di u sistemi lineari.
La firma, applicando l'ultima trasformazione affine L_2^{-1} sarà:

$$(z_1, ..., z_n) := L_2^{-1}(x_1, ..., x_n)$$

Il processo di generazione della firma, una volta scelti L_1, F ed L_2 si può riassumere nel seguente algoritmo:

Algorithm 1 Generazione firma Rainbow a u-1 livelli

Input La chiave segreta L_1, F, L_2, il documento d
Output Firma z

1: $h \leftarrow H(d)$ ▷ H è la funzione di hash scelta
2: $y \leftarrow L_1^{-1}(h)$
3: Scegliere un vettore random $(x_1, ..., x_{v_1})$
4: Calcolare $(\check{F}_{v+1}, ..., \check{F}_n)$ nelle variabili appena estratte ▷ I polinomi sono ora lineari
5: **for** $l = 1, ..., u$ **do**
6: Con eliminazione gaussiana ricavare x_i con $i \in O_l$ dai polinomi $\check{F}_i$ con $i \in O_l$
7: Sostituire i valori x_i trovati in $(\check{F}_{v_{l+1}+1}, ..., \check{F}_n)$
8: **end for**
9: $z \leftarrow L_2^{-1}(x)$

3.4 Verifica della firma

La verifica della firma avviene in due passaggi. Partendo dal documento d si calcola l'impronta hash con la funzione $\mathcal{H}$ scelta prima. Con la chiave pubblica $\overline{F}$ si calcola

$$\overline{F}(z_1, ..., z_n) = (h'_1, ..., h'_m).$$

Se vale la condizione

$$(h'_1, ..., h'_m) = (h_1, ..., h_m)$$

si considera la firma valida.

3.5 Scelta dei parametri

La scelta dei parametri per lo schema Rainbow dipende dal livello di sicurezza desiderato. Prima di tutto è necessario scegliere quanti livelli u sono necessari per costruire lo schema di firma. E' chiaro fin qui che $u = 1$ corrisponde al sistema Oil-Vinegar già visto in precedenza. La scelta di $u = 2$ offre uno schema più robusto, con chiavi più piccole senza sacrificare la sicurezza. Non sembra che ci siano benefici significativi dalla scelta di $u > 2$ che, sebbene dia un leggero miglioramento delle prestazioni, la dimensione delle chiavi aumenta per garantire lo stesso livello di sicurezza. Pertanto, la scelta di $u = 2$ è quella ideale, ed è anche quella sfruttata per la competizione del NIST [NIS21], che richiede vengano soddisfatti almeno 3 livelli di sicurezza. Avendo pertanto definito i livelli per tutte le categorie di sicurezza, per ogni schema Rainbow si riassumono i parametri che bisogna scegliere:

- il campo $\mathbb{F}_q$;
- il numero di livelli u, che è stato fissato a 2;

- il numero di variabili Vinegar v_i per ogni livello fino al livello $u-1$;
- il numero di variabili Oil o_i per ogni livello fino al livello u.

Per soddisfare i 3 livelli di sicurezza vengono proposti 3 insiemi di parametri:

Livello di sicurezza	q	v_1	o_1	o_2
I	16	36	32	32
III	256	68	32	48
V	256	96	36	64

Il numero di variabili Oil corrisponde al numero di sistemi lineari da risolvere: nel primo caso sono 64, nel secondo caso 80 mentre nel terzo caso sono 100. Infine è interessante osservare che i primi due insiemi di parametri soddisfano due categorie di sicurezza del NIST: Rainbow I soddisfa le categorie di sicurezza I e II, mentre Rainbow III soddisfa le categorie di sicurezza III e IV.

In base all'implementazione varia anche la funzione hash per calcolare l'impronta del documento: per Rainbow I si usa SHA256 mentre per Rainbow III e V si usano rispettivamente SHA384 e SHA512.

Capitolo 4

G*e*MSS

Nel capitolo precedente l'analisi si è concentrata su un algoritmo che si è classificato come finalista nell'ultimo round del NIST. Assieme ai candidati che potrebbero vincere la competizione si aggiungono anche dei candidati che vengono definiti come *candidati alternativi* che potrebbero essere oggetto di maggiore studio in un ulteriore round con lo scopo di scegliere un ulteriore standard.

G*e*MSS [Cas+17] (che sta per *Great Multivariate Short Signature*) è, come indicato dal nome, un crittosistema multivariato che la scopo di produrre firme di piccola dimensione. A differenza di Rainbow che era un crittosistema basato su Oil-Vinegar, G*e*MSS è un crittosistema basato su HFE e più precisamente è basato su HFEv-, in modo da essere resistente agli attacchi. Per com'è costruito G*e*MSS è un'evoluzione (più sicura e più veloce) di un precedente crittosistema chiamato **QUARTZ** [PCG01] che ancora non ha subito alcun attacco (il miglior attacco al crittosistema HFE è infatti proprio quello delle basi di Gröbner [FJ03]) pertanto si ritiene che sia questo il punto di forza di G*e*MSS.

4.1 Costruzione dello schema G*e*MSS

Innanzitutto è necessario scegliere un campo da cui partire per poter poi definire i polinomi: avendo a che fare con calcolatori è conveniente usare estensioni di $\mathbb{F}_2$, pertanto si dovrà scegliere il grado n dell'estensione con cui si vorrà lavorare.

Essendo uno schema basato su HFE è chiaro che bisogna definire il grado D del polinomio centrale dello schema, cioè il grado del polinomio segreto $\tilde{F}(X) \in \mathbb{F}[X]$ che, nello schema HFE è stato identificato con la scelta di i e j nel polinomio 2.5. Tuttavia G*e*MSS è uno schema che fa parzialmente uso di variabili Vinegar e dello schema *minus*, da cui bisognerà quindi scegliere il numero v di variabili Vinegar e il numero Δ che rappresenta il quanti polinomi verranno tolti dalla chiave pubblica finale. Il risultato sarà quindi una chiave pubblica di dimensione $m = n - \Delta$.

La formulazione del polinomio $\tilde{F}(X)$ sarà pertanto analoga a quella presentata per lo

schema HFEv:

$$\tilde{F}(x, z_1, ..., z_v) = \sum_{\substack{0 \le j < i < n \\ 2^i + 2^j \le D}} a_{ij} x^{q^i + q^j} + \sum_{\substack{0 \le i < n \\ 2^i \le D}} b_i(z_1, ..., z_v) x^{q^i} + c(z_1, ..., z_v).$$

Chiaramente si ha che i coefficienti $a_{ij} \in \mathbb{F}_{2^n}$ mentre i coefficienti b_i e c sono funzioni dipendenti dalle nuove variabili Vinegar. In generale si scelgono funzioni $b_i : \mathbb{F}_2^v \to \mathbb{F}_{2^n}$ **lineari** nelle variabili Vinegar, mentre $c : \mathbb{F}_2^v \to \mathbb{F}_{2^n}$ in generale sarà una funzione quadratica. Dal Teorema 1 è evidente che questa scelta porterà nuovamente ad avere coefficienti che saranno al più quadratici sia nelle variabili $(x_1, ..., x_n)$ che nelle variabili Vinegar $(z_1, ..., z_v)$. Nuovamente, in quanto GeMSS è un sistema bipolare, per completare il sistema si dovranno anche scegliere due trasformazioni L_1 e L_2 che in questo caso è sufficiente che siano trasformazioni lineari, dunque si dirà che $L_1 : \mathbb{F}_2^n \to \mathbb{F}_2^n$ e $L_2 : \mathbb{F}_2^{n+v} \to \mathbb{F}_2^{n+v}$.

4.2 Generazione della chiave

4.2.1 Chiave privata

Come in ogni sistema bipolare bisogna scegliere le mappe lineari L_1 e L_2 come chiave privata. Alle suddette si aggiunge alla chiave privata anche il polinomio centrale caratteristico $\tilde{F}$

$$\tilde{F}(x, z_1, ..., z_v) = \sum_{\substack{0 \le j < i < n \\ 2^i + 2^j \le D}} a_{ij} x^{q^i + q^j} + \sum_{\substack{0 \le i < n \\ 2^i \le D}} b_i(z_1, ..., z_v) x^{q^i} + c(z_1, ..., z_v).$$

Bisogna inoltre osservare che con GeMSS la scelta del polinomio irriducibile da cui si definisce l'estensione di campo di $\mathbb{F}_2$ è implicita nella scelta del grado n ma sarà comunque definita nelle specifiche di GeMSS . Ciò che è di interesse è che ci sarà nuovamente un isomorfismo $\phi : \mathbb{F}_{q^n} \to \mathbb{F}_q^n$ come definito in 2.4 che permetterà di definire n polinomi $f_1, ..., f_n$ nel modo seguente

$$\phi\left(\tilde{F}(X)\right) = (f_1, ..., f_n) \tag{4.1}$$

dove

$$f_i \in F_2[x_1, ...x_n, z_1, ...z_v] \tag{4.2}$$

con questa notazione pertanto le f_i sono le *componenti* di F sul campo $\mathbb{F}_2$.

4.2.2 Chiave pubblica

Per ottenere la chiave pubblica bisogna innanzitutto applicare le mappe L_1 e L_2 scelte in precedenza. Ricordando che in ogni sistema bipolare si ha generalmente una forma

$$\overline{F} := L_1 \circ F \circ L_2.$$

Applicando dunque le trasformazioni L_1 e L_2 si avranno n polinomi:

$$(f_1((x_1,..,x_n,z_1,...,z_v)L_2),...,f_n((x_1,...,x_n,z_1,...,z_v)L_2)L_1 = (p_1,...,p_n). \tag{4.3}$$

Questi polinomi poi devono essere ridotti modulo $< x_1^2-x_1,...,x_n^2-x_n, z_1^2-z_1,...,z_v^2-z_v >$. Prendendo inoltre i primi $m = n - \Delta$ polinomi si otterrà la chiave pubblica costituita da m polinomi multivariati *square-free* (questa particolare costruzione permette di avere una forma vantaggiosa per ottimizzare lo spazio occupato dalla chiave pubblica):

$$\overline{F} = (\overline{f}_1,...,\overline{f}_m)$$

dove

$$\overline{f}_i \in \mathbb{F}_2[x_1,...,x_n,z_1,...,z_v]$$

4.3 Generazione della firma

Si supponga ora di avere un messaggio M che si vuole firmare. In tutte le implementazioni di GeMSS si procede applicando la funzione hash **SHA-3** al messaggio da inviare, ottendendo un digest h. Si prendono quindi in considerazione i primi m bit di h e viene definita questa quantità d. Si procede poi scegliendo in modo casuale $r = (r_1,...,r_{n-m})$ che andrà aggiunto in coda a d, restituendo $d' = (d,r)$. Si calcola poi

$$D' := (\phi^{-1} \circ L_1^{-1})(d') \in \mathbb{F}_{2^n}. \tag{4.4}$$

Si vorrà quindi risolvere l'equazione

$$F(X,z_1,...,z_v) - D' = 0.$$

Un'importante osservazione da fare è questa, così com'è definita, in generale, un'equazione multivariata fino a che non vengono fissate le variabili Vinegar. Scegliendo casualmente le variabili Vinegar $\tilde{z} = (\tilde{z}_1,...,\tilde{z}_n)$ si ritorna a un'equazione del tutto analoga 2.6, che avrà la forma

$$F(X,\tilde{z}) - D' = 0. \tag{4.5}$$

La ricerca efficiente delle soluzioni è data, come detto con HFE, all'algoritmo di Berlekamp. Questa equazione può avere, in generale, più soluzioni e la soluzione proposta, che può essere identificata come $\tilde{X}$ è quella di scegliere casualmente una soluzione tra quelle proposte. Infine si calcola

$$s := (L_2^{-1} \circ \phi)(\tilde{X}, v)$$

che sarà la firma associata al messaggio M. Per com'è strutturato però il sistema però, la scelta di m lo rende facilmente attaccabile (ed era anche un attacco a cui era soggetto il protocollo QUARTZ). Per ovviare a questo problema, si itera il processo di calcolo della firma digitale più volte, scegliendo però come input non più i primi m bit di h (che ad

ogni iterazione è il digest del digest ottenuto nella fase precedente) ma lo XOR tra i primi m bit di h e i primi m bit dell'iterazione precedente che sarà identificata da S_{i-1}, mentre i bit rimanenti delle iterazioni precedenti vengono identificati con X_i. Questo processo, in tutti gli schemi di implementazione proposta, l'algoritmo viene fatto iterare 4 volte. In generale, in base al numero di iterazioni N di questo processo, la firma sarà

$$(S_N, X_N, ..., X_1). \tag{4.6}$$

Una piccola nota è dovuta all'inizializzazione del processo iterativo, in quanto si parte da una condizione $S_0 = 0$.

Algorithm 2 Generazione firma di GeMSS

Input La chiave segreta $(L_1,\ \tilde{F},\ L_2)$, ϕ, il documento M
Output Firma z

$h \leftarrow H(M)$ ▷ H è la funzione di hash **SHA-3**
$S_0 = 0$
for $i = 1, ..., nb_{ite}$ **do**
    $M_i \leftarrow$ primi m bit di H
    $d_i \leftarrow M_i \bigoplus S_{i-1}$
    repeat
        Scegliere casualmente $r_i \in \mathbb{F}_2^{n-m}$
        Scegliere casualmente $v \in \mathbb{F}_2^v$
        $D_i \leftarrow (\phi^{-1} \circ L_1^{-1})((d_i, r_i))$
        $F_{D'}(X) \leftarrow \tilde{F}(X, v) - D'$
        Roots $\leftarrow$ FindRoots$(F_{D'}(X))$ ▷ Si sfrutta l'algoritmo di Berlekamp
    until Roots$\neq \varnothing$
    Scegliere casualmente $Z \in$ Roots
    $P \leftarrow (\phi(Z), v)$
    $(S_i, X_i) \leftarrow P$ ▷ S_i rappresenta i primi m bit di P, quindi $S_i \in \mathbb{F}_2^m$ mentre X_i gli ultimi m bit di P, quindi $X_i \in F_2^{n+v-m}$
    $H \leftarrow hash(H)$
end for
$z \leftarrow (S_{nb_{ite}}, X_{nb_{ite}}, ..., X_1)$

4.4 Verifica della firma

Per verificare la firma è sufficiente procedere in direzione opposta a quanto fatto precedentemente. Si inizia calcolando la funzione hash del messaggio, si itera N volte il calcolo della funzione hash e si prendono ad ogni iterazione i primi m bit del digest. Infine si valuta il polinomio pubblico nell'accoppiata (S_{i+1}, X_{i+1}) ad ogni iterazione i e se ne calcola lo XOR con D_{i+1}. La firma è valida se l'output del processo è 0, che è la condizione iniziale imposta all'inizio della generazione della firma.

4.5 Scelta dei parametri

Analogamente a quanto visto con Rainbow, anche i parametri di GeMSS devono essere scelti per soddisfare almeno per soddisfare i 3 livelli di sicurezza richiesti dal NIST. Innanzitutto nel processo di definizione di GeMSS è stato visto come siano necessarie diverse iterazioni nella fase di calcolo della firma: questo parametro, che è stato battezzato nb_{ite} è stato fissato uguale a 4 per tutte le varianti di GeMSS . Le varianti proposte per GeMSS sono 3 e sono caratterizzate da tempi di esecuzione diverse: GeMSS tradizionale, BlueGeMSS e RedGeMSS che sono progressivamente più efficienti in termini di velocità di calcolo delle chiavi ma che hanno firme progressivamente più grandi. I parametri da scegliere per GeMSS sono

- D il grado del polinomio centrale;
- K che è la dimensione dell'output della funzione di hash scelta;
- λ che è il livello di sicurezza di GeMSS (che serve per identificare se si vuole soddisfare il primo, il terzo o il quinto livello di sicurezza del crittosistema);
- nb_{ite} che è stato detto essere fissato a 4;
- n il grado dell'estensione di $\mathbb{F}_2$;
- v il numero di variabili Vinegar;
- Δ il numero di minus, cioè il numero di equazioni da rimuovere dalla chiave pubblica;
- m che è il numero di polinomi che fanno parte della chiave pubblica e che è definito come $m := n - \Delta$

E' opportuno anche osservare che la funzione di hash scelta è SHA-3 di cui toccherà modificare solo il numero di bit di output.

4.5.1 Parametri di GeMSS

GeMSS è il sistema che è stato sottoposto al primo round del NIST per quanto riguarda la competizione dei sistemi di crittografia post-quantum. Vengono proposti qui tre livelli di parametri per rispettare tre livelli di sicurezza del NIST, in analogia a quanto visto con il crittosistema Rainbow:

Livello di sicurezza	D	K	λ	nb_{ite}	n	v	Δ
I	513	256	128	4	174	12	12
III	513	384	192	4	265	20	22
V	513	512	256	4	354	33	30

In base al livello di sicurezza vengono scelti tre polinomi con cui definire l'estensione di campo $\mathbb{F}_{2^n}$:

$$\text{I} : \frac{\mathbb{F}_2[x]}{X^{174} + X^{13} + 1}$$
$$\text{III} : \frac{\mathbb{F}_2[x]}{X^{265} + X^{42} + 1}$$
$$\text{V} : \frac{\mathbb{F}_2[x]}{X^{354} + X^{99} + 1}$$

4.5.2 Parametri di BlueG*e*MSS

La scelta intermedia tra velocità di esecuzione e dimensione della firma è BlueG*e*MSS di cui cambia la dimensione del polinomio D e il numero di v variabili Vinegar.

Livello di sicurezza	D	K	λ	nb_{ite}	n	v	Δ
I	129	256	128	4	175	14	13
III	129	384	192	4	265	23	22
V	129	512	256	4	358	32	34

Sempre in base al livello di sicurezza vengono scelti tre polinomi con cui definire l'estensione di campo $\mathbb{F}_{2^n}$:

$$\text{I} : \frac{\mathbb{F}_2[x]}{X^{175} + X^{16} + 1}$$
$$\text{III} : \frac{\mathbb{F}_2[x]}{X^{265} + X^{42} + 1}$$
$$\text{V} : \frac{\mathbb{F}_2[x]}{X^{358} + X^{57} + 1}$$

4.5.3 Parametri di RedG*e*MSS

La scelta più aggressiva in termini di velocità di calcolo della firma a scapito della sua dimensione e della velocità di verifica della stessa:

Livello di sicurezza	D	K	λ	nb_{ite}	n	v	Δ
I	17	256	128	4	177	15	15
III	17	384	192	4	266	25	23
V	17	512	256	4	358	35	34

Infine ecco la scelta dei polinomi con cui definire l'estensione di campo di RedG*e*MSS :

$$\text{I} : \frac{\mathbb{F}_2[x]}{X^{177} + X^{8} + 1}$$
$$\text{III} : \frac{\mathbb{F}_2[x]}{X^{266} + X^{47} + 1}$$
$$\text{V} : \frac{\mathbb{F}_2[x]}{X^{358} + X^{57} + 1}$$

Capitolo 5

Conclusioni

Questa tesi si è concentrata sulla descrizione di alcuni crittosistemi a chiave pubblica multivariati, che sono candidati a succedere i crittosistemi classici come standard di sicurezza nelle comunicazioni. La necessità di crittosistemi di tipo post-quantum è stata esplicitata con l'arrivo dell'algoritmo di Shor che, sebbene sia un algoritmo quantistico, rompe definitivamente sistemi come RSA, DSA e ECDSA che sono estremamente diffusi per la generazione di firme digitali (ECDSA ad esempio è largamente utilizzata nella blockchain di Bitcoin). Una precisazione degna di nota è che tuttora non esistono ancora computer quantistici in grado di rompere chiavi di RSA e sarà così fino a quando i suddetti non saranno abbastanza potenti, ma è chiaro che con il passare del tempo la tecnologia progredisce, nonostante i limiti fisici (temperature estremamente basse di operazione e grande quantità di errori da gestire con tecnologie di correzione di errore quantistiche). È notevole come poco più di 20 anni fa si sia riuscito a fattorizzare per la prima volta il numero 15 su hardware quantistico [IBM22].

Il passaggio più naturale quindi è quello di cercare uno standard in grado di rimpiazzare i sistemi odierni che sono caratterizzati da chiavi piccole e da velocità nella firma ed è proprio quello che ha fatto il NIST con la competizione lanciata nel 2017 [NIS17b]. I crittosistemi multivariati propongono di costruire dei polinomi di secondo grado in modo da creare quello che è stato chiamato un **problema quadratico multivariato**, che è noto per essere un problema di tipo NP-Completo e che non sempre è attaccabile efficientemente da un computer quantistico. Ci si è concentrati in questo lavoro su sistemi di tipo bipolari, il cui cuore è una mappa chiamata **mappa centrale** che è costituita da una serie di polinomi (che saranno necessariamente di grado superiore al primo), mascherando la mappa con due trasformazioni che prese singolarmente sono facilmente invertibili, in modo da creare una nuova mappa che è difficile da invertire. La sicurezza quindi è data dal fatto che solo chi conosce tutti gli elementi della chiave segreta (mappa centrale e trasformazioni affini) è in grado di invertire la funzione e cifrare o firmare un documento dato.

È stata offerta una disamina dei crittosistemi Oil-Vinegar e Hidden Field Equations che sono alla base di due candidati scelti dal NIST per il terzo round di standardizzazione di un crittosistema per la firma digitale: **Rainbow**, che è un finalista effettivo basato su Oil-Vinegar e **G*e*MSS**, basato su HFE, che è un finalista alternativo che può essere

scelto in un ipotetico quarto round. Di questi due crittosistemi quindi è stato descritto il funzionamento e sono stati inquadrati nel contesto dei sistemi bipolari introdotti inizialmente.

La prospettiva futura per questi crittosistemi è certamente una crittoanalisi approfondita per certificarne la sicurezza, in quanto alcuni concorrenti per la firma digitale arrivati al terzo round hanno subito alcuni attacchi (CRYSTALS-Dilithium [GMP21] e FALCON [KA21]) che nel secondo caso richiedono più attenzione in fase di standardizzazione.

Appendice A

L'algoritmo di Berlekamp

Nei crittosistemi prima elencati come HFE e GeMSS ad un certo punto è necessario un algoritmo che permetta una rapida fattorizzazione del polinomio in polinomi irriducibili: uno di questi è l'algoritmo di **Berlekamp** [Ber67]. Avere un sistema che permette l'efficiente fattorizzazione di un polinomio su un generico campo finito $\mathbb{F}_q$ è molto utile non solo per l'applicazione in questi specifici crittosistemi ma anche per essere in grado di determinare in modo efficiente se un polinomio su un determinato campo sia irriducibile o meno. E' chiaro che affinché la fattorizzazione ottenuta sia unica, il polinomio di partenza deve appartenere ad un anello a fattorizzazione unica, cosa vera nel caso in cui l'anello di polinomi è definito a partire da un campo finito, come nel caso in oggetto.

Si supponga di avere un generico polinomio $f(x) \in \mathbb{F}_q[x]$ square-free

$$f(x) = \sum_{k=0}^{m} f_k x^k, \quad f_k \in \mathbb{F}_q \tag{A.1}$$

e di volerlo fattorizzare. Se il polinomio non è square-free è sufficiente calcolare il massimo comune divisore tra il polinomio e la sua derivata formale e poi procedere con la sua fattorizzazione.

Il primo passo per fattorizzare il polinomio è costruire una matrice $Q \in \mathbb{F}_q^{m \times m}$ dove la riga i rappresenta $x^{q(i-1)} \mod f(x)$. Quindi si avrà, in ogni riga

$$x^{q(i-1)} \equiv \sum_{k=0}^{m-1} Q_{i,k+1} x^k \mod f(x). \tag{A.2}$$

Dato quindi un qualunque polinomio $g(x) \in \mathbb{F}_q[x]$ dove $\deg g < m$, $g(x) = \sum_{i=0}^{m-1} g_i x^i$ e si voglia ora calcolare il resto di $(g(x))^q \mod f(x)$. Per calcolare questo resto si può innanzitutto osservare che $(g(x))^q = g(x^q)$. Ma quindi

$$g(x^q) = \sum_{i=0}^{m-1} g_i z^{qi} = \sum_{i=0}^{m-1} \left(\sum_{k=0}^{m-1} g_i Q_{i+1,k+1} x^k \right) = \tag{A.3}$$

$$= \sum_{i=0}^{m-1} \left(\sum_{k=0}^{m-1} g_i Q_{i+1,k+1} \right) x^k. \tag{A.4}$$

Ma questo non è altro che il prodotto tra il vettore riga dei coefficienti di g, che è $[g_0, ..., g_{m-1}]$ e la matrice Q. La matrice Q, così com'è stata definita, rappresenta l'endomorfismo di Frobenius rispetto alla base $1, x, ..., x^{n-1} \mod f$ dell'anello quoziente $\mathbb{F}_q[x]/(f(x))$. È chiaro che la condizione imposta inizialmente sul grado del polinomio g non è davvero restrittiva nel momento in cui si lavora nella classe di resto di $f(x)$. Si calcola, in maniera del tutto analoga, il residuo $(g(x))^q - g(x)) \mod f(x)$ e si può esprimere come prodotto tra il vettore riga dei coefficienti di g e $Q - I$, dove $I \in \mathbb{F}_q^{m \times m}$ è la matrice identità.

Il passaggio successivo consiste nel cercare una base per il nucleo di $Q - I$. Ogni elemento della base soddisferà

$$(g(x))^q - g(x) \equiv 0 \mod f(x) \tag{A.5}$$

e, viceversa, ogni $g(x)$ che soddisfi la condizione suddetta può essere scritta come un vettore nel nucleo di $(Q-I)$. Gli elementi che soddisfano A.5 appartengono all'anello quoziente $\mathbb{F}_q[x]/(f(x))$ e gli stessi formano una sottoalgebra che prende il nome **sottoalgebra di Berlekamp**.

Una volta trovata questa base si sceglie uno qualunque tra i polinomi $g(x)$ trovati e si applica l'algoritmo di Euclide per calcolare $(f(x), g(x) - s)$ (si ricorda che $MCD(a, b)$ è indicato con (a, b)) per tutte le $s \in F_q$. Il risultato di questo sarà la fattorizzazione seguente:

$$f(x) = \prod_{s \in \mathbb{F}_q} (f(x), g(x) - s). \tag{A.6}$$

Una particolarità di questa costruzione è che se $g(x)$ ha grado nullo, allora la fattorizzazione di $f(x)$ è banalmente $f(x)$. Più interessante è il caso in cui $g(x)$ ha un grado positivo, in quanto la fattorizzazione sarà non banale. Per mostrare questo innanzitutto si osservi che per costruzione $(g(x))^q - g(x) \equiv 0 \mod f(x)$ e pertanto $f(x)$ divide $(g(x))^q - g(x)$ ma vale anche

$$(g(x))^q - g(x) = \prod_{s \in \mathbb{F}_q} (g(x) - s). \tag{A.7}$$

Da qui quindi si può affermare che $f(x)$ divide

$$\prod_{s \in \mathbb{F}_q} (f(x), g(x) - s)). \tag{A.8}$$

Viceversa si può dire sicuramente che $(f(x), g(x) - s)$ per $s \in \mathbb{F}_q$ divide $f(x)$. Si prenda $t \neq s$ tale che $t \in \mathbb{F}_q$, allora si potrà dire che $g(x) - s$ e $g(x) - t$ saranno coprimi e saranno coprimi anche $(f(x), g(x) - s)$ e $(f(x), g(x) - t)$. Pertanto si potrà dire che $\prod_{s \in \mathbb{F}_q} (f(x), g(x) - s)$ divide $f(x)$. Assumendo che i due polinomi siano monici, allora devono essere uguali visto che entrambi si dividono a vicenda.

La fattorizzazione del polinomio $f(x)$ in generale si potrà scrivere come

$$f(x) = \prod_{i=1}^{n} p^{(i)}(x) \tag{A.9}$$

dove ogni $p^{(i)}(x) \in \mathbb{F}_q[x]$ rappresenta un polinomio irriducibile. Il teorema cinese dei resti garantisce che, in generale, dati $s_1, ..., s_n \in \mathbb{F}_q$, esiste ed è unica la soluzione alle equazioni $g(x) = s_i \mod p^{(i)}(x)$. Visto che gli scalari sono n ci saranno esattamente q^n scelte possibili e tante saranno le soluzioni all'equazione $(g(x))^q - g(x) = 0 \mod f(x)$. In generale si può dire che il numero di fattori distinti in cui si può fattorizzare un polinomio $f(x)$ è uguale alla dimensione del nucleo dell'applicazione $Q-I$. Alla luce di questo quindi è chiaro che il polinomio $f(x)$ è potenza di un polinomio irriducibile se e solo se il nucleo di $Q - I$ ha dimensione 1. In questo caso le soluzioni al polinomio A.5 sono esattamente gli scalari in $\mathbb{F}_q$ e il nucleo $Q - I$ ha elementi del tipo [s,0, ...,0].

Se invece il nucleo ha dimensione n, la base di questo spazio sarà composta esattamente da n polinomi monici $g^{(1)}(x), ..., g^{(n)}(x)$. Si applica poi l'algoritmo di Euclide tra $f(x)$ e $g^{(1)}(x) - s$ per cercare i fattori comuni distinti, applicando poi lo stesso algoritmo tra $f(x)$ e $g^{(2)}(x) - s$ e così via e l'algoritmo si ferma una volta fattorizzato completamente il polinomio.

Questo algoritmo è deterministico, quindi fornirà sempre una fattorizzazione completa del polinomio. Per mostrare questo si consideri $S \in \mathbb{F}_q^{n\times n}$ definita da $g^{(j)} \equiv s_{i,j} \mod p^{(i)}(x)$. S in questo caso dev'essere non singolare, in quanto se

$$\sum_j A_i S_{ij} \text{ per } i = 1, ..., n \Rightarrow \sum_j A_i g^{(j)}(x) \equiv 0 \mod p^{(i)}(x) \text{ per } i = 1, ..., n \tag{A.10}$$

$$\Rightarrow \sum_j A_i g^{(j)}(x) = 0 \text{ per } i = 1, ..., n \tag{A.11}$$

ma A.11 contraddice il fatto che gli elementi della base trovata sono linearmente indipendenti. Quando si calcola il massimo comune divisore tra $f(x)$ e $g^{(j)}(x) - s$ si ottiene una fattorizzazione di $f(x)$ in tanti fattori quanti sono gli elementi distinti della riga j di S. Le potenze irriducibili $p^{(i)}(x)$ e $p^{(k)}(x)$ quindi sono separate se e solo se $S_{i,j} \neq S_{k,j}$. Visto che S è non singolare, per ogni i e k esiste una j tale che $S_{i,j} \neq S_{k,j}$. Da qui si potrà quindi dire che fattori irriducibili di $f(x)$ saranno separati da qualche $g^{(i)}(x)$.

Appendice B

Comparazione tra i due crittosistemi

In tutti i sistemi crittografici sono due le caratteristiche importanti: la sicurezza e la velocità di cifratura e/o di firma. Per quanto riguarda Rainbow e GeMSS sono presenti notevoli differenze in termini di velocità e di dimensione delle firme generate. Per valutare le prestazioni dei due crittosistemi il NIST [NIS17a] ha adottato una piattaforma di rifermento con processore Intel e con sistema operativo a scelta tra Windows e Linux. Sebbene i processori utilizzati per la valutazione delle prestazioni non siano uguali, la comparazione può essere fatta a parità di architettura (in questo caso Intel Skylake) per le implementazioni di riferimento che non fanno uso di speciali set di istruzioni mentre per quanto riguarda i sistemi operativi sono entrambi basati su Linux. L'unità di misura utilizzata per la comparazione delle velocità in questo caso è quella dei cicli di clock (indicati con C) per lo svolgimento dei compiti assegnati, affinché la misurazione sia indipendente dalla frequenza del processore. Questo strumento di misura è integrato nel compilatore GCC usato per compilare il codice relativo ad entrambi i crittosistemi. In questo caso i sistemi vengono usati per firmare un documento standard di 32 bytes.

Velocità

Di seguito sono esplicitati i tempi di generazione della chiave di Rainbow, per ogni set di parametri sottomesso al NIST.

Livello di sicurezza	Gen. chiavi	Gen. firma	Verifica firma
I	32 MC	0.319 MC	41 kC
III	197 MC	1.47 MC	203 kC
V	436 MC	2.48 MC	362 kC

Di seguito, invece, i tempi per GeMSS.

Set di Parametri	Livello di sicurezza	Gen. chiavi	Gen. firma	Verifica firma
GeMSS128	I	1.88 GC	6690 MC	29.1 MC
BlueGeMSS128	I	1.51 GC	774 MC	30 MC
RedGeMSS128	I	1.21 GC	17.6 MC	26.8 MC
GeMSS192	III	7.92 GC	15100 MC	89 MC
BlueGeMSS192	III	6.72 GC	1280 MC	89 MC
RedGeMSS192	III	5.89 GC	28 MC	72.3 MC
GeMSS256	V	20.5 GC	25300 MC	172 MC
BlueGeMSS256	V	19.4 GC	1640 MC	184 MC
RedGeMSS256	V	17.7 GC	37.3 MC	146 MC

La prima caratteristica che salta all'occhio è la velocità più alta a seconda della scelta del grado del polinomio centrale di GeMSS. La scelta del sistema Red presenta sia una maggior velocità di generazione delle chiavi ma soprattutto presenta un taglio importante del tempo di generazione della firma, senza presentare significative differenze in termini di verifica.

La seconda caratteristica è che nonostante GeMSS abbia un'implementazione più orientata alla velocità velocità, comunque rimane inferiore rispetto a Rainbow che si comporta meglio di GeMSS in tutti e tre i livelli di sicurezza proposti. Per entrambi i crittosistemi sono presenti inoltre implementazioni che fanno uso di speciali set di istruzioni per accelerare i calcoli (AVX2, disponibili solo su architetture x86) e in ogni caso vantaggio di Rainbow sulla generazione delle chiavi è netto, vantaggio che viene amplificato su processori embedded relativi a implementazioni a basso costo. Nel documento di Rainbow è infatti offerta anche un'implementazione su un'architettura a basso costo.

Dimensioni di chiavi e firme

Un'ulteriore comparazione, anch'essa molto utile soprattutto in sistemi a basso costo, è quella relativa alle dimensioni delle chiavi e firme che vengono generate. Qui di seguito viene presentata la tabella di Rainbow relativamente alla dimensione della chiave pubblica e della chiave privata in bytes, mentre la dimensione della firma è presentata in bit.

Livello di sicurezza	Chiave pubblica	Chiave Privata	Firma
I	157.8 kB	101.2 kB	528 bit
III	861.4 kB	611.3 kB	1312 bit
V	1885.4 kB	1375.7 kB	1696 bit

Mentre di seguito la dimensione delle chiavi di tutte le varianti di GeMSS.

Set di Parametri	Livello di sicurezza	Chiave pubblica	Chiave privata	Firma
GeMSS128	I	417.4 kB	14.5 kB	258 bit
BlueGeMSS128	I	430.9 kB	14.7 kB	270 bit
RedGeMSS128	I	444.7 kB	13.8 kB	282 bit
GeMSS192	III	1304.2 kB	40.3 kB	411 bit
BlueGeMSS192	III	1331.7 kB	41.7 kB	423 bit
RedGeMSS192	III	1359.6 kB	40.8 kB	435 bit
GeMSS256	V	3046.8 kB	83.7 kB	576 bit
BlueGeMSS256	V	3094.2 kB	78.1 kB	588 bit
RedGeMSS256	V	3141.9 kB	78.4 kB	600 bit

Anche qui è chiaro come ci siano differenze tra i due crittosistemi. Mentre la chiave pubblica è in generale più piccola nel crittosistema Rainbow, la situazione si ribalta per quanto riguarda la chiave privata e la dimensione della firma. Tuttavia è opportuno specificare come le differenze in questo caso non siano nette come nel caso delle prestazioni e nel caso di Rainbow sono previste anche varianti che permettono di ridurre le dimensioni delle chiavi, rendendo il sistema competitivo anche dal punto di vista delle dimensioni delle chiavi.

Bibliografia

[Ber67] Elwyn R. Berlekamp. «Factoring polynomials over finite fields». In: *The Bell System Technical Journal* 46.8 (1967), pp. 1853–1859. DOI: 10.1002/j.1538-7305.1967.tb03174.x.

[DH76] Whitfield Diffie e Martin E. Hellman. «New Directions in Cryptography». In: *IEEE Transactions on Information Theory* 22.6 (nov. 1976), pp. 644–654.

[Mil76] Gary L. Miller. «Riemann's hypothesis and tests for primality». In: *Journal of Computer and System Sciences* 13.3 (1976), pp. 300–317. ISSN: 0022-0000. DOI: https://doi.org/10.1016/S0022-0000(76)80043-8. URL: https://www.sciencedirect.com/science/article/pii/S0022000076800438.

[GJ79] Michael R. Garey e David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979, p. 251. ISBN: 0716710455.

[Dav82] George I. Davida. *Chosen signature cryptanalysis of the RSA (MIT) public key cryptosystem*. Technical Report TR-CS-82-2. Department of Electrical, Computer Science, College of Engineering e Applied Science, University of Wisconsin, Miluwakee, ott. 1982.

[RSA83] Ronald L. Rivest, Adi Shamir e Leonard M. Adleman. *CRYPTOGRAPHIC COMMUNICATION SYSTEM AND METHOD patent no. US4405829A*. Set. 1983.

[Hås88] Johan Håstad. «Solving simultaneous modular equations of low degree». In: *SIAM Journal on Computing* 17 (1988). http://www.nada.kth.se/~johanh/papers.html, pp. 336–341. ISSN: 0097–5397.

[Wie90] Michael J. Wiener. «Cryptanalysis of short RSA secret exponents». In: *IEEE Transactions on Information Theory* 36.3 (1990), pp. 553–558. DOI: 10.1109/18.54902.

[BLP93] Joe P. Buhler, Hendrik W. Lenstra e Carl Pomerance. «Factoring integers with the number field sieve». In: *The development of the number field sieve*. A cura di Arjen K. Lenstra e Hendrik W. Lenstra. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 50–94. ISBN: 978-3-540-47892-8.

[Gor93] Daniel Gordon. «Discrete Logarithms in GF(P) Using the Number Field Sieve». In: *Siam Journal on Discrete Mathematics - SIAMDM* 6 (feb. 1993), pp. 124–138. DOI: 10.1137/0406010.

[Sha94] Adi Shamir. «Efficient Signature Schemes Based on Birational Permutations». In: *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '93. Santa Barbara, California, USA: Springer-Verlag, 1994, pp. 1–12. ISBN: 0387577661.

[Sho94] Peter W. Shor. «Algorithms for quantum computation: discrete logarithms and factoring». In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[Pat96] Jacques Patarin. «Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms». In: *Advances in Cryptology — EUROCRYPT '96*. A cura di Ueli Maurer. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 33–48. ISBN: 978-3-540-68339-1.

[Cop97] Don Coppersmith. «Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities». In: *Journal of Cryptology* 10.4 (set. 1997), pp. 233–260. ISSN: 1432-1378. DOI: 10.1007/s001459900030. URL: https://doi.org/10.1007/s001459900030.

[PG97] Jacques Patarin e Louis Goubin. «Trapdoor one-way permutations and multivariate polynomials». In: *Information and Communications Security*. A cura di Yongfei Han, Tatsuaki Okamoto e Sihan Qing. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 356–368. ISBN: 978-3-540-69628-5.

[KS98] Aviad Kipnis e Adi Shamir. «Cryptanalysis of the oil and vinegar signature scheme». In: *Advances in Cryptology — CRYPTO '98*. A cura di Hugo Krawczyk. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 257–266. ISBN: 978-3-540-68462-6.

[KPG99] Aviad Kipnis, Jacques Patarin e Louis Goubin. «Unbalanced Oil and Vinegar Signature Schemes». In: *Advances in Cryptology — EUROCRYPT '99*. A cura di Jacques Stern. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 206–222. ISBN: 978-3-540-48910-8.

[PCG01] Jacques Patarin, Nicolas Courtois e Louis Goubin. «QUARTZ, 128-bit long digital signatures». In: vol. 2020. Apr. 2001, pp. 282–297. ISBN: 978-3-540-41898-6. DOI: 10.1007/3-540-45353-9_21.

[RS01] Ronald L. Rivest e Robert D. Silverman. «Are 'Strong' Primes Needed for RSA». In: (2001). https://ia.cr/2001/007.

[FJ03] Jean-Charles Faugère e Antoine Joux. «Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases». In: *Advances in Cryptology - CRYPTO 2003*. A cura di Dan Boneh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60. ISBN: 978-3-540-45146-4.

[DS05a] Jintai Ding e Dieter Schmidt. «Cryptanalysis of HFEv and Internal Perturbation of HFE». In: vol. 3386. Gen. 2005, pp. 288–301. ISBN: 978-3-540-24454-7. DOI: 10.1007/978-3-540-30580-4_20.

[DS05b] Jintai Ding e Dieter Schmidt. «Rainbow, a New Multivariable Polynomial Signature Scheme». In: *Applied Cryptography and Network Security*. A cura di John Ioannidis, Angelos Keromytis e Moti Yung. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 164–175. ISBN: 978-3-540-31542-1.

[DGS06] Jintai Ding, Jason E. Gower e Dieter S. Schmidt. «Oil-Vinegar Signature Schemes». In: *Multivariate Public Key Cryptosystems*. Boston, MA: Springer US, 2006, pp. 63–97. ISBN: 978-0-387-36946-4. DOI: 10.1007/978-0-387-36946-4_3. URL: https://doi.org/10.1007/978-0-387-36946-4_3.

[KSD13] Cameron F. Kerry, Acting Secretary e Charles Romine Director. *FIPS PUB 186-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Digital Signature Standard (DSS)*. 2013.

[Cas+17] Antoine Casanova, Jean-Charles Faugère, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret e Jocelyn Ryckeghem. «GeMSS: A Great Multivariate Short Signature». In: 2017.

[NIS17a] NIST. *Post Quantum Cryptography: Evaluation Process*. 2017. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-process.

[NIS17b] NIST. *Post-Quantum Cryptography Standardization*. 2017. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization.

[NIS20] NIST. *Post-Quantum Cryptography Standardization*. 2020. URL: https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement.

[GMP21] Paul Grubbs, Varun Maram e Kenneth G. Paterson. *Anonymous, Robust Post-Quantum Public Key Encryption*. Cryptology ePrint Archive, Report 2021/708. https://ia.cr/2021/708. 2021.

[KA21] Emre Karabulut e Aydin Aysu. *Falcon Down: Breaking Falcon Post-Quantum Signature Scheme through Side-Channel Attacks*. Cryptology ePrint Archive, Report 2021/772. https://ia.cr/2021/772. 2021.

[NIS21] NIST. *Post Quantum Cryptography: Security (Evaluation Criteria)*. 2021. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria).

[IBM22] IBM. *It's been 20 years since "15" was factored on quantum hardware*. 2022. URL: https://research.ibm.com/blog/factor-15-shors-algorithm.