Agenzia per la
Cybersicurezza Nazionale

LINEE GUIDA FUNZIONI CRITTOGRAFICHE

Firme Digitali

MAGGIO 2026

Versione	Data di pubblicazione	Note
1.0	**12/05/2026**	**Prima pubblicazione**

Sommario

pag.

Indice delle figure

pag.

Indice delle tabelle

Scopo del documento

Questo documento costituisce parte della serie "Linee Guida Funzioni Crittografiche" e fornisce le raccomandazioni di ACN in merito agli schemi crittografici per la **firma digitale**, da adottare in tutti i contesti in cui serve garantire integrità, autenticazione e non-ripudio dei dati inviati. Per ulteriori informazioni sulle Linee Guida e sulle utilità delle funzioni crittografiche in base ai contesti specifici, si faccia riferimento al documento introduttivo della serie [1].

Ogni documento tiene in considerazione le minacce presenti al giorno della sua pubblicazione. Data la diversa natura dei sistemi informativi di destinazione, non è possibile garantire che queste raccomandazioni possano essere utilizzate senza adattamenti specifici. In qualsiasi caso, la pertinenza dell'attuazione delle soluzioni proposte deve essere sottoposta, preventivamente, a valutazione e validazione da parte dei responsabili della sicurezza dei sistemi informativi di destinazione.

I contenuti delle Linee Guida sono indirizzati a sviluppatori, produttori di dispositivi e fornitori di servizi digitali, al fine di promuovere l'utilizzo di primitive crittografiche e relativi parametri di configurazione sicuri fin dalla fase di progettazione di prodotti, reti, applicazioni e servizi. Questi documenti sono altresì rivolti a security manager, referenti e responsabili della cybersicurezza di soggetti pubblici e privati affinché verifichino che i sistemi utilizzati dalla propria organizzazione siano conformi alle raccomandazioni fornite.

La legge 28 giugno 2024, n. 90 relativa a "Disposizioni in materia di rafforzamento della cybersicurezza nazionale e di reati informatici", all'articolo 9, stabilisce a tal fine che *«le strutture di cui all'articolo 8 della presente legge nonché quelle che svolgono analoghe funzioni per i soggetti di cui all'articolo 1, comma 2-bis, del decreto-legge 21 settembre 2019, n. 105, convertito, con modificazioni, dalla legge 18 novembre 2019, n. 133, e al decreto legislativo 18 maggio 2018, n. 65, verificano che i programmi e le applicazioni informatiche e di comunicazione elettronica in uso, che utilizzano soluzioni crittografiche, rispettino le linee guida sulla crittografia nonché quelle sulla conservazione delle password adottate dall'Agenzia per la cybersicurezza nazionale e dal Garante per la protezione dei dati personali e non comportino vulnerabilità note, atte a rendere disponibili e intellegibili a terzi i dati cifrati»*.

Il documento è stato curato dal Centro Nazionale di Crittografia istituito presso ACN.

Lista dei simboli matematici utilizzati

$\mod n$	Riduzione modulo n, restituisce il resto della divisione per n	$\phi(N)$	Funzione di Eulero, se $N = p \cdot q$ allora $\phi(N) = (p-1) \cdot (q-1)$
mcm	Minimo comune multiplo	MCD	Massimo comune divisore
$\lambda(N)$	Funzione di Carmichael, se $N = p \cdot q$ allora $\lambda(N) = mcm(p-1, q-1)$	$\mathbb{Z}_q$	Insieme degli interi modulo q
$\oplus$	Operazione XOR, ovvero la somma bit a bit tra stringhe binarie	$A^{k \times l}$	Matrici a coefficienti in A con k righe e l colonne
$\|\|$	Concatenazione di stringhe	$\mathbb{Z}_q[x]$	Polinomi con incognita x a coefficienti in $\mathbb{Z}_q$
$\mathbb{F}_p$	Campo finito con p elementi	R_q	Anello $\mathbb{Z}_q[x]/(x^{256}+1)$ contenente i polinomi in $\mathbb{Z}_q[x]$ di grado al più 255

1 Introduzione

In un mondo in cui la maggior parte dei dati e delle comunicazioni sono digitali, un requisito fondamentale è la possibilità per chiunque di applicare la propria firma in modo che tutti possano verificarne la validità e associare i dati firmati al rispettivo firmatario. Tale firma assume quindi il valore di una tradizionale firma autografa, rimuovendo però il vincolo di dover firmare dal vivo i documenti, semplificando la procedura. Questo processo può essere richiesto esplicitamente da un utente, ad esempio tramite l'utilizzo di uno dei servizi di firma remota o elettronica, ma più in generale viene eseguito automaticamente all'interno di numerosi protocolli digitali. Il concetto crittografico di firma digitale raccoglie quindi tutte queste tipologie di schemi e, assieme agli schemi di cifratura e di scambio di chiave, è uno degli ingredienti fondamentali per la crittografia moderna.
Il paradigma su cui si basa ogni schema di firma digitale standardizzato è quello della crittografia asimmetrica o a chiave pubblica: la chiave crittografica non è unica, ma si utilizza una coppia di chiavi. Una di esse è privata, cioè conosciuta solamente al firmatario, che la utilizza per generare la propria firma sui dati; a questa prima chiave se ne associa una pubblica, che viene fornita a chiunque voglia verificare la validità della firma apposta. La sicurezza di questa tipologia di schemi si basa su problemi matematici la cui risoluzione risulta molto complessa, tranne se si conosce il segreto rappresentato dalla chiave privata.
Per fornire un completo supporto sull'argomento, questo documento ha l'obiettivo di presentare i concetti generali alla base delle firme digitali, utili per comprendere il funzionamento e le differenze fra i vari schemi.
Successivamente, vengono approfonditi i dettagli tecnici, inclusi i parametri raccomandati per gli schemi riportati, così da supportare gli addetti ai lavori nell'identificare le scelte più appropriate in base agli specifici scenari di utilizzo.
Il documento si struttura come segue: nel capitolo 2 vengono introdotte le principali definizioni alla base degli schemi di firma digitale e dei certificati, un altro concetto strettamente correlato; nel capitolo 3 si descrivono nel dettaglio gli schemi di firma digitale classici, cioè basati su problemi non resistenti agli attacchi dei computer quantistici; il capitolo 4 si incentra quindi sulla crittoanalisi di tali schemi, concentrandosi sia sulla computazione classica sia su quella quantistica; gli schemi considerati sicuri anche in quest'ultimo contesto sono descritti nel capitolo 5; seguono un confronto tra i vari schemi dal punto di vista delle dimensioni dei dati trasmessi e delle prestazioni nel capitolo 6 e, infine, le raccomandazioni sugli schemi di firma digitale e i parametri sicuri nel capitolo 7.

2 Firme digitali e certificati

L'idea di un analogo digitale della firma autografa risale ai padri della crittografia a chiave pubblica, Diffie e Hellman, i quali nel 1976 definirono il concetto di firma digitale, descrivendone le proprietà senza tuttavia proporre uno schema concreto [2]. L'anno successivo venne ideato lo schema RSA [3], che riprese il concetto di firma digitale fornendo un'applicazione reale basata sulla matematica elementare e l'aritmetica modulare. A partire da queste pubblicazioni, la ricerca nel settore conobbe una rapida espansione: furono infatti proposti numerosi nuovi schemi di firma digitale e, negli anni successivi, alcuni di essi vennero riconosciuti dalla comunità scientifica e successivamente standardizzati.

Nel 1988, Goldwasser, Micali e Rivest [4] definirono formalmente per la prima volta i requisiti di sicurezza per le firme digitali, descrivendo i possibili scenari di attacco e l'impatto di tali attacchi sugli schemi in questione.

Tra gli standard più diffusi a livello internazionale si affermarono il Digital Signature Algorithm (DSA), introdotto nel 1991, e la sua versione basata su curve ellittiche (ECDSA), pubblicata nel 2000. Nonostante ciò, la ricerca di nuovi schemi non si è mai arrestata, soprattutto in risposta a contesti che richiedono particolari vincoli o l'introduzione di proprietà aggiuntive.

Tra gli sviluppi più recenti, uno dei più rilevanti è legato alla minaccia posta dall'avvento dei computer quantistici, che mette a rischio l'intera crittografia a chiave pubblica attualmente in uso. Di conseguenza, si rende necessaria l'adozione di nuovi protocolli in grado di resistere a questa tecnologia emergente. Le soluzioni più promettenti sono rappresentate dagli schemi cosiddetti post-quantum. In questo ambito, il National Institute of Standards and Technology (NIST) statunitense ha avviato nel 2016 una competizione internazionale per la selezione di nuovi standard. A seguito della conclusione di tale processo, che ha portato all'individuazione di tre nuovi schemi di firma digitale, è stata avviata una seconda fase di selezione, tuttora in corso, con l'obiettivo di identificare ulteriori alternative basate su differenti problemi matematici. Per un approfondimento su questo argomento, si rimanda al documento informativo dedicato [5].

2.1 Gli schemi di firma digitale

Una firma, in generale, assicura l'approvazione e la validazione di un documento, di un contratto o di un'opera d'arte, nonché l'identificazione del firmatario.

Ad esempio, nel caso di un contratto cartaceo, la firma autografa garantisce in particolare che il firmatario:

- approvi il contratto così come viene firmato, non concedendo nessuna variazione successiva;
- dichiari di essere effettivamente chi dice di essere, eventualmente aggiungendo il proprio documento;
- non possa negare di avere apposto la propria firma e quindi di aver approvato il contratto.

L'analogo digitale viene definito teoricamente come uno strumento in grado di assicurare le medesime funzioni. Nello specifico, queste corrispondono a richiedere che la firma di un utente su un determinato dato garantisca:

- l'**integrità**, cioè la possibilità di verificare che il dato firmato non sia stato alterato rispetto a quando è stato approvato e firmato;
- l'**autenticazione**, in quanto dev'essere possibile distinguere il firmatario da un possibile malintenzionato che tenti di apporre una firma a suo nome;
- il **non-ripudio**, cioè l'impossibilità da parte del firmatario di negare di aver firmato di suo pugno quel dato.

Dal punto di vista crittografico, gli schemi di firma digitale fanno parte dei crittosistemi a **chiave pubblica** e pertanto prevedono che l'utente firmatario possieda una chiave pubblica pk e una chiave privata sk. Per ulteriori dettagli sulle proprietà sopra elencate e sulla definizione di crittografia a chiave pubblica si rimanda al documento di introduzione alla crittografia e alle linee guida [1].
Uno schema di firma digitale è composto solitamente da tre algoritmi. Supponendo che Alice voglia firmare un messaggio m e che Bob voglia verificarne la validità, tali algoritmi sono:

1. **generazione delle chiavi**: Alice genera la propria coppia di chiavi, composta da una chiave pubblica pk e una chiave privata sk, oltre a eventuali parametri necessari per il corretto funzionamento dello schema scelto. Generalmente, le chiavi e i parametri ottenuti possono essere validi per più messaggi senza dover essere ricreati ogni volta. Tuttavia, bisogna prestare attenzione al tempo di validità delle chiavi, chiamato anche **crittoperiodo**, e rigenerarle quando queste raggiungono la scadenza. Nel caso in cui le chiavi generate siano valide per una singola firma, si parla di schema di firma effimero. Questa tipologia di firma fornisce una sicurezza elevata al costo di un importante aumento del costo computazionale dovuto alla necessità di generare una nuova coppia di chiavi ad ogni istanza e di inviare ogni volta la chiave pubblica insieme alla firma per rendere possibile la verifica della stessa;
2. **generazione della firma**: una volta generate le chiavi, Alice genera la firma vera e propria del messaggio. In questa fase, Alice applica una funzione di hash h (spesso definita dallo schema in uso) al messaggio in chiaro e genera poi la firma σ dal digest ottenuto usando la chiave segreta sk. La firma viene quindi condivisa a Bob insieme al messaggio stesso, eventualmente cifrato se risulta necessario mantenerne la confidenzialità;
3. **verifica della firma**: a partire dal messaggio m, dalla firma σ e dalla chiave pubblica pk, Bob controlla se la firma applicata da Alice al messaggio risulta valida. Il risultato di questo calcolo è semplicemente un valore booleano (vero/falso) che rivela la validità della firma.

2.2 I certificati digitali

Affinché una firma digitale possa fornire correttamente l'autenticazione, e quindi resistere ad attacchi del tipo man-in-the-middle, è fondamentale che il ricevente possa essere certo che la chiave pubblica utilizzata in fase di verifica appartenga effettivamente al firmatario.
Nella pratica, l'associazione tra utente e chiave pubblica viene realizzata tramite un documento detto **certificato digitale**, rilasciato preventivamente da un ente fidato denominato **Autorità di Certificazione** o **CA** (Certification Authority). In questo modo, se il firmatario invia il proprio certificato insieme al messaggio firmato, la fiducia nella CA garantisce al destinatario la validità della chiave pubblica da utilizzare per la verifica della firma.
Nello specifico, un certificato digitale deve contenere i seguenti elementi:

- il **nome del titolare** del certificato;
- un **numero seriale** che identifica univocamente il certificato rilasciato da una certa CA;
- la **data di scadenza** del certificato, oltre la quale il certificato non è più valido;
- la **chiave pubblica** del titolare del certificato, che può essere utilizzata per decifrare i messaggi o validare le firme provenienti dal possessore del certificato;
- la **firma digitale** della CA, al fine di verificare che il certificato non sia falso.

La validità di un certificato cessa con la sua scadenza o con la revoca da parte della CA. In quest'ultimo caso, il certificato viene aggiunto a una lista pubblica contenente tutti i certificati revocati, detta Certificate Revocation List (CRL).
Il formato dei certificati digitali è definito tramite standard internazionali. Il più utilizzato su internet, in particolare nel protocollo SSL/TLS, è quello elaborato dalla International Telecommunication Union (ITU-T) chiamato X.509 [6].

3 Schemi di firma classici

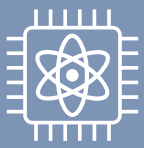
Come decritto nel documento introduttivo [1] e nel documento informativo su crittografia post-quantum e quantistica [5], la minaccia quantistica richiede nuovi paradigmi di sicurezza ed è quindi importante distinguere due categorie di schemi crittografici. Questo capitolo si incentra sugli schemi di firma digitale la cui sicurezza si basa su **problemi classici**, cioè matematicamente complessi per un computer classico ma risolvibili, almeno teoricamente, da un computer quantistico.

3.1 I problemi classici

I problemi matematici alla base degli standard classici di firma digitale sono principalmente due: la fattorizzazione di interi e il logaritmo discreto. Entrambi sono stati ampiamente studiati e risultano computazionalmente complessi anche utilizzando tecniche moderne.

3.1.1 Fattorizzazione di interi

Uno dei problemi più antichi e più famosi su cui si basa la sicurezza crittografica di schemi come RSA è il problema della **fattorizzazione dei numeri interi**. L'assunzione fondamentale è che, sebbene il prodotto fra due numeri primi $pq = N$ sia particolarmente semplice da calcolare, partendo da N è computazionalmente complesso ricavare i suoi fattori p e q, in particolare se sono numeri molto grandi e di lunghezza simile.



Minaccia quantistica

I problemi della fattorizzazione e del logaritmo discreto, su cui si basano molti dei moderni schemi di crittografia a chiave pubblica, risultano **VULNERABILI** agli attacchi perpetrati da un computer quantistico tramite l'**algoritmo di Shor** [7], che consente di risolverli in un tempo polinomiale rispetto alla dimensione dei dati. Nonostante i computer quantistici attuali non siano ancora sufficientemente potenti, è necessario prepararsi all'eventualità che questi possano costituire una minaccia reale entro qualche decennio. In attesa degli sviluppi di questa nuova tecnologia, si raccomanda di sostituire al più presto gli schemi classici con alternative resistenti agli attacchi quantistici. Per maggiori dettagli sulla minaccia quantistica, si rimanda al documento informativo dedicato [5].

Per dare un'idea delle dimensioni effettive, attualmente si utilizza RSA con una chiave pubblica lunghe 2048 bit, corrispondenti a un intero N di al più 617 cifre decimali, ma a partire da gennaio 2026 si raccomanda di incrementare ulteriormente tale dimensione a un minimo di 3072 bit.

Sebbene non esista una dimostrazione formale della complessità di questo problema, ogni strategia e algoritmo classico realizzato finora non risulta efficiente o consente di risolverne solo particolari istanze. Al momento, il migliore algoritmo per eseguire la fattorizzazione è il crivello dei campi numerici generale (GNFS, dall'inglese General Number Field Sieve) [8], il quale si basa su tecniche avanzate di teoria dei numeri che permettono di ridurre la complessità di risoluzione a livello subesponenziale (ancora molto lontano dalla complessità polinomiale richiesta per la risoluzione efficiente del problema). Grazie al GNFS negli ultimi anni sono stati fattorizzati moduli RSA sempre più grandi, fino al record raggiunto nel 2020 con la fattorizzazione di un numero con 250 cifre decimali, corrispondenti a 829 bit. Ulteriori dettagli sugli attacchi sono forniti nella sezione 4.1.

3.1.2 Logaritmo discreto

Un altro dei principali problemi matematici utilizzati per la costruzione di cifrari a chiave pubblica è il **Problema del Logaritmo Discreto** (DLP), introdotto per la prima volta da Whitfield Diffie and Martin Hellman nel 1976 nella proposta del famoso schema di scambio chiave che porta il loro nome [2]. Questo problema si esprime all'interno di una particolare struttura algebrica chiamata **gruppo**: un insieme di elementi munito di un'operazione con delle proprietà specifiche. Un semplice esempio di gruppo è l'insieme dei numeri interi $\mathbb{Z}$ con l'operazione somma. All'interno dei gruppi utilizzati in crittografia si possono identificare degli elementi speciali detti **generatori** dai quali è possibile ottenere l'intero insieme applicando ripetutamente l'operazione. In formule, dato un generatore α, ogni elemento β si potrà ricavare come $\beta = \alpha^x$ per un qualche intero x, dove l'operazione di elevamento a potenza è da interpretare come l'applicazione dell'operazione del gruppo ripetuta x volte. Risolvere il logaritmo discreto consiste nel trovare l'intero x dati gli elementi α e β.
Esclusi alcuni gruppi all'interno dei quali il DLP è facilmente risolvibile e altre istanze particolari per le quali esistono metodi risolutivi subesponenziali, gli unici algoritmi che risolvono il DLP nel caso generale hanno complessità esponenziale nella dimensione dei parametri. In particolare, la formulazione del DLP con le curve ellittiche, chiamata **ECDLP**, risulta molto più complessa da risolvere rispetto alle istanze con altri gruppi, a parità di dimensione dei dati. Per ulteriori dettagli su attacchi e sicurezza si veda la sezione 4.2.

3.2 Schema di firma RSA

Uno dei primi e più famosi schemi di crittografia a chiave pubblica è sicuramente **RSA**, sigla dovuta ai cognomi dei tre ideatori Rivest, Shamir e Adleman, che lo hanno introdotto nel 1978 [3]. Il loro articolo descrive sia uno schema di cifratura a chiave pubblica (o Public-Key Encryption, PKE), sia uno schema di firma digitale, entrambi basati sulla **fattorizzazione dei numeri interi**, un problema proveniente dalla teoria dei numeri. Nonostante si tratti di un problema storico, studiato fin dai tempi degli antichi greci, la complessità richiesta per risolverlo risulta elevata anche con i mezzi più moderni. Per questo motivo, RSA risulta tuttora molto utilizzato, sebbene siano state apportate alcune modifiche all'implementazione originale e la dimensione delle chiavi sia stata progressivamente aumentata per resistere agli attacchi che sono stati ideati nel corso degli anni. Per maggiori dettagli sugli attacchi a RSA si veda la sezione 4.1.
Gli standard di firma RSA sono definiti dalla Internet Engineering Task Force (IETF) che propone due varianti, entrambe nella categoria SSA (Signature Scheme with Appendix) [9]: RSASSA-PKCS1-v1.5 (Public-Key Cryptography Standards), ora considerata obsoleta, e **RSASSA-PSS** (Probabilistic Signature Scheme), descritta in seguito. Altri documenti che definiscono lo schema come standard sono stati pubblicati dal NIST [10] o da ISO [11].

3.2.1 Generazione delle chiavi RSASSA-PSS

In questa fase, per prima cosa viene generato l'**esponente pubblico** e, cioè un intero dispari lungo almeno 17 bit e al più 256 bit. Una scelta molto comune per l'esponente pubblico è $e = 2^{16} + 1 = 65537$ che rende molto efficienti gli elevamenti a potenza. In seguito, dato un intero pari $n \geq 2048$ rappresentante la lunghezza del **modulo pubblico**, vengono generati casualmente due numeri primi p e q in modo che le seguenti proprietà siano rispettate [10]:

- $p - 1$ e $q - 1$ sono coprimi con e;
- p e q sono più grandi di $2^{(n-1)/2}$ e lunghi al più $n/2$ bit;

- la differenza tra p e q dev'essere almeno $2^{n/2-100}$;
- $p \pm 1$ e $q \pm 1$ hanno un fattore primo di almeno 140, 170 e 200 bit per n uguale a 2048, 3072 e 4096, rispettivamente.

In questo modo si ha la certezza che l'esponente pubblico sia invertibile, che il modulo $N = pq$ sia lungo n bit e che sia sicuro. Inoltre, si calcola l'**esponente privato** $d = e^{-1} \mod \lambda(N)$, dove $\lambda(N) = \text{mcm}(p-1, q-1)$ è la cosiddetta funzione di Carmichael. È importante controllare che d sia lungo più di $n/2$ bit e generare una nuova coppia di primi p, q qualora questa condizione non fosse rispettata.
La **chiave pubblica** è composta dalla coppia $pk = (N, e)$, mentre per la **chiave privata** sk ci sono diverse possibilità di rappresentazione:

- (N, d), la possibilità più semplice e diretta;
- (p, q, d), la terna con i fattori primi di N, che consente di utilizzare più efficientemente l'esponente segreto;
- $(N, p, q, d, d_p, d_q, q_{inv})$, dove $d_p = e^{-1} \mod (p-1)$, $d_q = e^{-1} \mod (q-1)$ e $q_{inv} = q^{-1} \mod p$. Questa versione è ancora più efficiente dal punto di vista computazionale, a scapito della memoria necessaria, e sfrutta il cosiddetto Teorema Cinese dei Resti o CRT (Chinese Remainder Theorem) per ottenere delle informazioni parziali sulle chiavi. Il CRT afferma che l'elevamento a potenza $M^d \mod N$ può essere calcolato ricavando l'unica soluzione $x < N$ del sistema dato dalle congruenze

$$\begin{aligned} x &\equiv M^{d_p} \mod p, \\ x &\equiv M^{d_q} \mod q, \end{aligned}$$

  il quale richiede calcoli con numeri molto più piccoli rispetto a quelli effettuati modulo N.

Inoltre è necessario concordare una funzione di hash o XOF h crittograficamente sicura, per le quali si rimanda al documento dedicato [12].

3.2.2 Generazione della firma RSASSA-PSS

Preliminarmente il messaggio m viene elaborato da una funzione di codifica utilizzando un **salt**. In particolare, il salt è una stringa di bit casuale e diversa per ogni firma che rende RSASSA-PSS uno schema probabilistico, ossia tale che ogni firma differisce dalle altre, anche se applicata dallo stesso soggetto allo stesso messaggio. La funzione restituisce una stringa di lunghezza limitata della forma

$$M = salt \oplus mask \,\|\, H,$$

dove $H = h(h(m) \,\|\, salt)$ e $mask$ è una stringa in funzione di H generata appositamente.
La generazione della **firma** σ avviene effettuando le seguenti operazioni, a seconda della rappresentazione della chiave privata utilizzata:

- nel primi due casi, $\sigma = M^d \mod N$;
- nel caso della rappresentazione CRT, si calcolano prima gli interi $\sigma_p = M^{d_p} \mod p$ e $\sigma_q = M^{d_q} \mod q$, poi si ricava $k = (\sigma_p - \sigma_q) q_{inv} \mod p$ e infine si ottiene il valore della firma come $\sigma = \sigma_q + kq$.

Le firme così ottenute coincidono, ma il secondo caso risulta più efficiente in quanto si svolgono operazioni con valori molto più piccoli rispetto al primo, che è rallentato dai calcoli modulo N che è molto grande.

3.2.3 Verifica della firma RSASSA-PSS

Nella fase di verifica della firma, si ha a disposizione il messaggio m, la firma σ e la chiave pubblica (N, e). Si ricostruisce quindi $M' = \sigma^e \mod N$, a cui si applica l'inversa della funzione di codifica che restituisce

$$M' = salt' \oplus mask' \,\|\, H',$$

per cui si può ricavare $mask'$ dal valore H' e quindi ricostruire $salt'$. A questo punto, si calcola nuovamente $H = h(h(m) \,\|\, salt')$ e si compara il valore ottenuto con H'. Se i due valori coincidono la firma è valida, altrimenti la firma viene considerata invalida e viene riportato un messaggio di errore.

3.2.4 Sicurezza di RSASSA-PSS

Lo schema di firma RSA risulta sicuro in quanto un attaccante che non conosce la chiave privata non è in grado di produrre una firma RSA valida per un messaggio scelto o casuale (diverso da uno già firmato).
In particolare, se non si utilizza una funzione di codifica, sarebbe possibile generare la firma del prodotto tra due messaggi firmati semplicemente valutando il prodotto tra le due firme conosciute, e quindi lo schema risulterebbe malleabile. Questa vulnerabilità si risolve semplicemente introducendo la funzione di codifica.

RSA-n	Sicurezza
2048	112 bit
3072	128 bit
4096	192 bit
15360	256 bit

Tabella 1 - Dimensioni minime per RSA per livello di sicurezza

Nel caso generale, l'attaccante deve riuscire a valutare l'elevamento alla potenza segreta d. Svolgere questa operazione senza conoscere d è matematicamente equivalente a trovare i fattori primi p e q di N, e quindi la sicurezza di RSA si riconduce alla difficoltà della fattorizzazione di interi.
La Tabella 1 specifica i valori minimi per la lunghezza della chiave pubblica di RSA e i corrispondenti livelli di sicurezza, ognuno dei quali è determinato dalla complessità di realizzazione dell'attacco più efficiente, come specificato nel documento introduttivo [1]. Si osservi come la prima riga, corrispondente a RSA-2048 e a soli 112 bit di sicurezza, è segnalata in rosso in quanto tale soluzione risulta obsoleta a partire da gennaio 2026.

3.3 Schema di firma con curve ellittiche

Lo schema di firma digitale DSA è stato proposto dal NIST nel 1991 come standard denominato Digital Signature Standard (DSS), ed è stato il primo schema di firma digitale riconosciuto da un governo. Si tratta di una variante SSA dello schema di ElGamal [13] che si basa sul **logaritmo discreto** e sull'utilizzo di una funzione di hash (nello standard iniziale era esplicitamente SHA-1). Dopo aver ricevuto diversi aggiornamenti, DSA è diventato obsoleto a partire dalla versione del 2023 dello standard [10] per questioni di vulnerabilità ed efficienza.
In particolare, DSA non può competere con la più recente variante che utilizza le **curve ellittiche**, una costruzione matematica derivante dalla geometria sulla quale si definisce un'operazione che permette di ottenere un nuovo tipo di gruppo. La formulazione risultante, chiamata ECDSA, ha riscosso fin dalla sua introduzione nel 1992 [14] un grande successo per via dell'altissima efficienza computazionale e di memoria. A parità di sicurezza, questo vantaggio delle curve ellittiche rispetto ai classici gruppi utilizzati in DSA è dovuto a un grande risparmio nella lunghezza in bit dei valori su cui si eseguono i calcoli, e quindi anche di chiavi e firma.
Lo standard ECDSA è stato pubblicato dal NIST [10] e da ISO [15], ma sono previste anche delle varianti come EC-KCDSA (Elliptic Curve Korean Certificate-based DSA) e ECGDSA (Elliptic Curve German DSA) [16], le quali presentano solo leggere differenze rispetto allo schema standard.

3.3.1 Le curve ellittiche

Nonostante il nome richiami la figura geometrica dell'ellissi, non c'è nessun legame con essa. Una curva ellittica E è un insieme di punti con coordinate date dall'equazione

$$E(x, y) : y^2 = x^3 + ax + b,$$

con $4a^3 + 27b^2 \neq 0$. Questa formulazione è la più utilizzata ed è detta forma di Weierstrass, ma esistono formulazioni alternative, meno comuni, come la forma di Montgomery

$$E(x, y) : By^2 = x^3 + Ax^2 + x,$$

dove $A \neq \pm 2$ e $B \neq 0$.
A prescindere dalla forma adottata, è possibile definire un'operazione di somma tra i punti di una curva ellittica considerando che ogni retta interseca la curva in esattamente 3 punti non necessariamente distinti e scegliendo come elemento neutro il punto all'infinito (O), cioè un immaginario terzo punto di intersezione comune a tutte le rette verticali. In Figura 1 si rappresentano i grafici di due diverse curve ellittiche con, a sinistra, l'interpretazione grafica dell'operazione di somma e, a destra, del prodotto di un punto per un intero (l'analogo dell'elevamento a potenza).

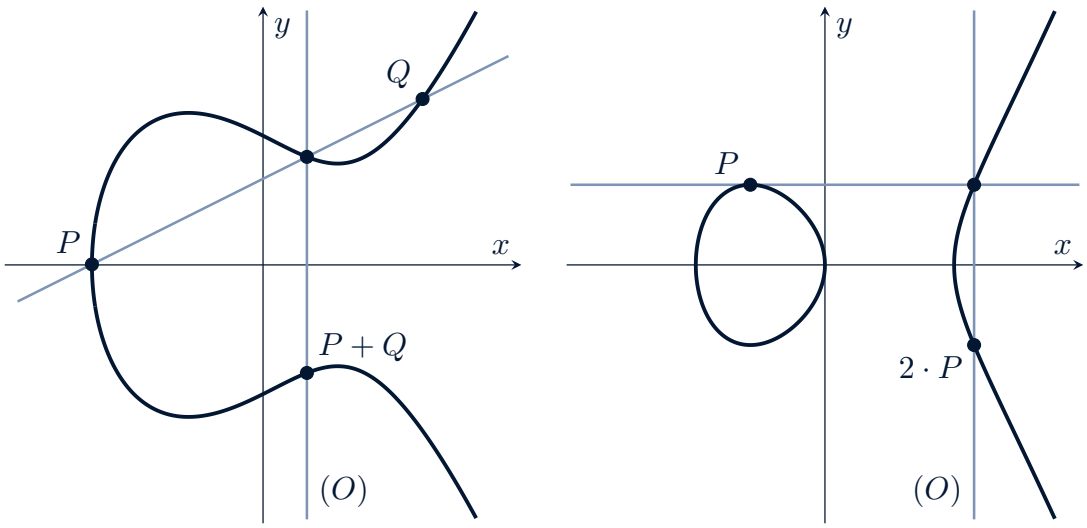


Figura 1 - Esempi di operazioni su curve ellittiche: a sinistra, somma tra punti, a destra, moltiplicazione per un intero

La struttura di gruppo definita dalla somma e la possibilità di prendere coordinate su **campi finiti** [17], cioè insiemi finiti su cui sono definite due operazioni (generalmente somma e prodotto) che rispettano diverse proprietà, rendono le curve ellittiche particolarmente utili dal punto di vista crittografico. In generale, considerare campi con un numero primo p di elementi $\mathbb{F}_p$ corrisponde a selezionare un sottoinsieme finito dei punti della curva che rispetta le proprietà di gruppo con l'operazione di somma definita nella figura. A questo punto, è possibile scegliere un **punto base** P come generatore e considerare il sottogruppo da esso generato, cioè $\{k \cdot P \mid k = 0, 1, 2, \ldots\}$ che avrà un numero finito q di punti detto ordine di P (in particolare $q \cdot P = (O)$). Questa struttura può quindi essere sfruttata per formulare un'istanza di **ECDLP**.
La scelta della curva da utilizzare nello schema è il parametro più delicato, per cui si raccomanda di utilizzare una delle curve note per le robuste proprietà di sicurezza. In certi contesti la curva può essere generata casualmente partendo da un seed tramite procedure standardizzate, le quali prevedono anche un metodo di identificazione del processo adottato. Tuttavia, si raccomanda sempre di preferire una delle curve elencate in Tabella 2, in modo da evitare eventuali problematiche di sicurezza. Di ognuna di queste curve è definita: l'equazione della curva E, il campo finito $\mathbb{F}_p$ sulla quale questa curva viene definita, il punto base P e il suo ordine q. Per semplicità, la tabella contiene solamente il livello di sicurezza e la forma della curva, per gli altri parametri si rimanda ai riferimenti specifici.
Negli schemi che utilizzano curve ellittiche si raccomanda sempre di controllare che i punti considerati, come ad esempio il punto base P, appartengano alla curva, cioè che verifichino l'equazione della curva. Infatti, esistono diverse strategie di attacco che sfruttano dei falsi punti di una curva per ottenere informazioni sulle chiavi private.

3.3.2 Generazione delle chiavi ECDSA

Una volta stabiliti globalmente i parametri pubblici come il campo $\mathbb{F}_p$, l'equazione della curva E, il punto base P e il suo ordine q, la **chiave privata** è un intero $k < q$ ottenuto tramite un generatore di numeri casuali sicuro.
La **chiave pubblica** associata a k è il punto della curva ottenuto calcolando $Q = k \cdot P$.
Inoltre è necessario concordare una funzione di hash o XOF h crittograficamente sicura, per le quali si rimanda al documento dedicato [12].

3.3.3 Generazione della firma ECDSA

Per generare la firma di un messaggio m è necessario conoscere i parametri di base e utilizzare la chiave privata.

Curva	Sicurezza	Riferimenti	Forma
secp256r1	128 bit	NIST [18]	Weierstrass
secp384r1	192 bit		
secp521r1	256 bit		
brainpoolP256r1	128 bit	IETF [19]	
brainpoolP384r1	192 bit		
brainpoolP512r1	256 bit		
Curve25519	128 bit	IETF [20]	Montgomery
Curve448	224 bit		

Tabella 2 - Curve ellittiche raccomandate con corrispondenti livelli di sicurezza, riferimenti e forma

Le curve NIST sono uno standard crittografico ampiamente adottato, ma la legittimità della scelta dei parametri viene spesso contestata in quanto i documenti ufficiali non forniscono dettagli sul processo adottato per la loro generazione. Per questo motivo, alcuni strumenti di cybersecurity e valutazione automatica ne sconsigliano l'uso, preferendo curve di nuova generazione (come Brainpool o Montgomery) che offrono una maggiore trasparenza nella selezione dei parametri. Nonostante tale scetticismo, non esistono evidenze scientifiche di vulnerabilità nelle curve NIST e non sono noti attacchi pratici contro protocolli che le utilizzano, per cui tali soluzioni restano raccomandate.

Come nel DSA classico, lo schema di firma ECDSA utilizza un **nonce** $n < q$ che può essere:

- **casuale**, a patto che sia ottenuto tramite un generatore di numeri casuali sicuro. In questo caso, le firme generate sono sempre diverse tra loro, anche se il messaggio è lo stesso, e questo consente di evitare attacchi di recupero della chiave;
- **deterministico**, cioè calcolato a partire dal messaggio da firmare e dalla chiave privata, utilizzando l'algoritmo HMAC [21]. In questo caso, firmare più volte lo stesso messaggio fornisce sempre lo stesso risultato. Questa versione è raccomandata nei casi in cui non si ha a disposizione un generatore di numeri casuali efficiente e sicuro, ma rende necessario prendere precauzioni contro attaccanti in grado di monitorare il canale di comunicazione (attacchi man-in-the-middle).

La generazione della firma prevede i seguenti passi:

1. si calcola il punto $R = n \cdot P = (x_R, y_R)$;
2. la prima parte della firma è l'intero $r = x_R \mod q$;
3. la seconda parte della firma è calcolata come
$$s = n^{-1}(h(m) + rk) \mod q;$$
4. se $r \neq 0$ e $s \neq 0$, allora la **firma** è data dalla coppia (r, s). Altrimenti: se n è casuale, si genera un nuovo valore e il processo ricomincia; se n è deterministico, l'algoritmo termina con un errore.

Per garantire la sicurezza della firma è fondamentale che n resti segreto, così come la chiave privata k.
Da osservare come, nel caso deterministico, è possibile che la firma generata non sia valida. Quindi, n deve essere generato in modo da ridurre la probabilità che ciò avvenga e, in ogni caso, è necessario tenere in considerazione la possibilità che il passo 4 restituisca un errore.

3.3.4 Verifica della firma ECDSA

La fase di verifica della firma (r, s) del messaggio m avviene utilizzando la chiave pubblica Q. Una volta controllato che questo punto sia effettivamente un punto della curva E,

vengono svolti i seguenti passaggi:

1. si calcola l'inverso di s modulo l'ordine q, cioè $s_{inv} = s^{-1} \mod q$;
2. si calcolano gli interi $u = h(m)s_{inv} \mod q$ e $v = rs_{inv} \mod q$;
3. si calcola il punto $R' = u \cdot P + v \cdot Q = (x_{R'}, y_{R'})$. Se $R' = (O)$, allora la firma viene rifiutata;
4. si verifica se $r \equiv x_{R'} \mod q$. In caso affermativo la firma è valida, altrimenti viene rifiutata.

3.3.5 Sicurezza di ECDSA

Come già specificato, tutte le varianti di DSA basano la sicurezza contro attacchi di falsificazione sulla difficoltà del **DLP** e dell'invertire una funzione di hash. Per maggiori dettagli si rimanda al documento introduttivo [1]. In particolare, le istanze di **ECDLP** in ECDSA sono due:

- ricavare la chiave privata sapendo il generatore e la chiave pubblica, dato che $Q = k \cdot P$;
- ricavare il nonce n a partire da firma e chiave pubblica, siccome $n \cdot P = (x_R, y_R)$.

In aggiunta, la dimensione dei dati scambiati (chiavi e firme) risulta minore rispetto a RSA o DLP con altri gruppi, a parità di livello di sicurezza. Per queste motivazioni, le curve ellittiche hanno riscosso grande successo e sono al momento alla base della maggior parte dei sistemi che richiedono schemi di firma digitale.

3.4 Schema di firma con le curve di Edwards

Non tutti gli schemi di firma che utilizzano curve ellittiche si basano sul classico DSA. In particolare, è possibile considerare una variante della firma di Schnorr [22] sfruttando una tipologia di curve ellittiche, dette curve di Edwards. Lo schema risultante, chiamato **EdDSA**, è uno degli standard più recenti [10]. A differenza di ECDSA, non viene utilizzato un nonce casuale in quanto la firma generata con EdDSA si ottiene deterministicamente a partire dal messaggio m e dal digest della chiave privata, calcolato utilizzando una funzione di hash prestabilita.

3.4.1 Le curve di Edwards ritorte

Una curva di Edwards ritorta (in inglese, twisted Edwards curve) è una curva ellittica E nella forma

$$E(x, y) : ax^2 + y^2 = 1 + dx^2y^2,$$

con $a, d \neq 0$ e $a \neq d$. Si tratta di una generalizzazione delle curve di Edwards che presentano la stessa forma ma con $a = 1$. Inoltre, ogni curva di Edwards ritorta è equivalente a una curva nella forma di Montgomery, per cui le curve standard sono le stesse introdotte precedentemente. Come nel caso generale, è possibile definire un'operazione di somma tra i punti di una curva di Edwards che determina una struttura di gruppo, ma la formulazione risulta più efficiente rispetto a quella ottenibile su una generica curva ellittica. Anche in questo caso la curva viene definita su un campo finito, in modo che le istanze di ECDLP ottenute siano computazionalmente difficili da risolvere.

3.4.2 Generazione delle chiavi EdDSA

Per prima cosa si stabiliscono globalmente i parametri pubblici, cioè:

- un campo finito $\mathbb{F}_p$ con $p < 2^{b-1}$ primo, per cui i punti della curva occupano al più b bit;
- una curva di Montgomery E che su $\mathbb{F}_p$ risulta avere $2^c q$ punti con c intero e q primo;
- un punto base P di ordine q;
- una funzione di hash o XOF h con digest lunghi $2b$ bit.

Nello specifico, i parametri degli schemi EdDSA standardizzati sono riportati in Tabella 3.

La **chiave privata** è un intero k di b bit ottenuto tramite un generatore di numeri casuali sicuro. Da essa si calcola $h(k) = l \parallel s$, dove l, s sono stringhe di b bit, e la **chiave pubblica** è il punto della curva $Q = s \cdot P$.

Schema	Sicurezza	Curva	Lunghezza chiavi (b)	Hash/XOF (h)	Riferimenti
Ed25519	128 bit	Curve25519	256	SHA-512	NIST [18], IETF [23]
Ed448	224 bit	Curve448	456	SHAKE256	

Tabella 3 - Parametri standard per EdDSA

3.4.3 Generazione della firma EdDSA

La **firma** di un messaggio m è data dalla coppia (R, S), dove:

- $R = r \cdot P$ è un punto sulla curva E con $r = h(l \,\|\, m)$;
- $S = r + h(R \,\|\, Q \,\|\, m)s \mod q$ è un intero di b bit.

3.4.4 Verifica della firma EdDSA

Per verificare la validità della firma (R, S) sul messaggio m, dati i parametri concordati e la chiave pubblica Q, è sufficiente controllare la validità della seguente equazione tra punti sulla curva

$$(2^c S) \cdot P = 2^c \cdot R + (2^c h(R \,\|\, Q \,\|\, m)) \cdot Q.$$

3.4.5 Sicurezza di EdDSA

Come per ECDSA, la sicurezza si basa sulla difficoltà del **ECDLP** e dell'invertire una funzione di hash. In particolare, questi due problemi rendono computazionalmente difficile ricavare la chiave privata k a partire da:

- generatore e chiave pubblica, dato che $Q = s \cdot P$ con $h(k) = l \,\|\, s$;
- firma e chiave pubblica, siccome $R = r \cdot P$ con $r = h(l \,\|\, m)$.

Le considerazioni effettuate per ECDSA sulla dimensione dei dati scambiati a parità di livello di sicurezza sono valide anche per EdDSA.

4 Crittoanalisi degli schemi classici

Rompere uno schema di firma digitale corrisponde ad avere successo nella sua **contraffazione**, cioè nel produrre una coppia di messaggio e firma (m, σ) valida, che sia m scelto o casuale, senza conoscere la chiave privata. Nonostante gli schemi descritti siano standard validi e non esistano attacchi efficienti rispetto ai livelli di sicurezza raccomandati, è importante prestare attenzione a particolari casi vulnerabili.

4.1 Attacchi a RSASSA

Come osservato precedentemente, la sicurezza dello schema RSA si può ricondurre alla difficoltà della fattorizzazione di un intero molto grande. Sebbene non sia mai stato ideato un algoritmo di complessità polinomiale implementabile su computer classici che risolva tale problema in generale, esistono attacchi che funzionano particolarmente bene quando i fattori primi p, q o gli esponenti e, d non vengono scelti in modo adeguato.
Per questa ragione, il NIST ha stabilito la seguente lista di criteri da rispettare affinché l'implementazione di RSA possa essere ritenuta sicura [10]:

1. la lunghezza n di N deve essere pari e almeno 2048 (in aggiornamento ad almeno 3072);
2. $2^{(n-1)/2} < p < 2^{n/2}$ e $2^{(n-1)/2} < q < 2^{n/2}$;
3. $|p - q| > 2^{n/2-100}$;
4. $p \pm 1$ e $q \pm 1$ hanno un fattore primo di almeno 140, 170 e 200 bit per n pari a 2048, 3072 e 4096, rispettivamente;
5. $2^{16} < e < 2^{256}$;
6. $2^{n/2} < d < \text{mcm}(p-1, q-1) = \lambda(N)$.

Gli attacchi che hanno portato alla scelta di questi criteri sono descritti nelle sezioni seguenti.

4.1.1 Attacchi al problema della fattorizzazione

Come osservato precedentemente, alla base della sicurezza di RSA vi è l'incapacità di un attaccante di recuperare la chiave privata d ed è stato dimostrato matematicamente che recuperare questo valore è equivalente a trovare la **fattorizzazione** di N, cioè a ricavare i fattori primi p e q oppure $\lambda(N)$ o ancora il valore della funzione di Eulero $\phi(N) = (p-1)(q-1)$. Per questa ragione, la comunità scientifica si è concentrata sulla ricerca di algoritmi che potessero risolvere il problema di fattorizzare un numero in tempi accettabili utilizzando uno o più calcolatori.
L'idea alla base dei principali metodi di fattorizzazione è quella di trovare due interi x, y tali che $x^2 \equiv y^2 \mod N$ ma $x \not\equiv \pm y \mod N$, in quanto, se tali condizioni fossero rispettate, $\text{MCD}(x - y, N)$ sarebbe un divisore non banale di N. Il metodo attualmente più efficace per trovare una coppia (x, y) adeguata è il General Number Field Sieve (GNFS) [8] che, ad oggi, ha permesso di fattorizzare al più RSA-250 (829 bit).
Per mantenere un buon margine di sicurezza, si raccomanda l'utilizzo di RSA con chiavi di minimo 3072 bit, mentre RSA-2048 deve essere dismesso al più presto.

I controlli 1 e 2 del NIST rispondono alla richiesta di utilizzare valori di p e q abbastanza grandi da fare in modo che il modulo RSA corrispondente N sia difficile da fattorizzare, ma abbia comunque la lunghezza desiderata. In realtà, esistono vari algoritmi che risolvono il problema della fattorizzazione in tempi relativamente brevi e con minore complessità, ma essi sono applicabili solo a particolari scelte dei parametri. Questi attacchi, sebbene non risolvano il problema della fattorizzazione nella sua formulazione più generale, sono di estrema importanza in quanto vincolano nella scelta dei parametri. Alcuni contesti in cui esistono algoritmi di risoluzione efficienti sono elencati in seguito.

Primi troppo vicini: alcuni algoritmi come il metodo di Fermat o il metodo di Lehman [24] consentono di ricavare p e q quando la distanza tra di essi è piccola. Per questo motivo è importante effettuare il controllo 3.

Fattori primi piccoli nella fattorizzazione di $p \pm 1, q \pm 1$: l'algoritmo $p-1$ di Pollard [25] permette di ricavare la fattorizzazione di un modulo RSA nel caso in cui i fattori di almeno uno tra $p-1$ e $q-1$ siano tutti piccoli. Analogamente l'attacco $p+1$ di Williams [26] ricava i fattori di N, quando almeno uno tra $p+1$ e $q+1$ si fattorizza completamente utilizzando fattori di dimensioni contenute. Il requisito 4 è stato quindi introdotto in modo da rendere inefficaci questi attacchi. Va tuttavia notato che se i valori di p e q sono generati casualmente, data la loro dimensione, la probabilità che i fattori di $p \pm 1$ e $q \pm 1$ siano tutti piccoli può essere considerata trascurabile.

Conoscenza di alcuni bit di p e q: tra i numerosi metodi per recuperare la chiave privata di RSA che non si basano sulla generazione dei parametri iniziali, è utile evidenziare quelli che sfruttano la conoscenza di alcuni bit di p e q. In questo contesto, un attaccante in possesso di alcuni blocchi di bit consecutivi di uno dei due fattori primi, invece di effettuare un attacco a forza bruta per indovinare i bit rimanenti, può utilizzare algoritmi specifici [27, 28] per recuperare completamente il fattore parzialmente conosciuto. Tuttavia, va notato che la complessità di questi algoritmi dipende notevolmente dal numero di blocchi e dal numero di bit che l'attaccante è riuscito a recuperare.

Esponente pubblico piccolo: nel caso in cui si utilizzino degli esponenti pubblici piccoli ($e = 3$ o $e = 17$ erano scelte comuni) e il messaggio risulti particolarmente corto, esistono alcuni attacchi basati sul metodo di Coppersmith [29] per la ricerca di soluzioni di una congruenza polinomiale modulo N. Il controllo 5 permette di evitare tale vulnerabilità. Come già osservato, nella maggioranza dei casi si utilizza il valore $e = 2^{16} + 1 = 65537$ che rispetta tale condizione e rende più efficienti gli elevamenti a potenza. In aggiunta, la funzione di RSASSA-PSS che codifica il messaggio prima della cifratura fornisce totale copertura contro questa tipologia di attacchi.

Esponente privato piccolo: il controllo 6 serve a rendere inefficace l'attacco di Wiener [30], che permette di ricavare d a partire dalla conoscenza della chiave pubblica nel caso in cui $d < \sqrt[4]{n}/3$. Questo attacco sfrutta la rappresentazione di frazioni continue, uno strumento proveniente dalla teoria dei numeri.

4.1.2 Attacchi all'implementazione dello schema

La vulnerabilità più comune nelle implementazioni di RSA riguarda il **generatore di numeri pseudocasuali** adottato. Questa è una scelta fondamentale per la fase di generazione delle chiavi, in quanto se i due primi p e q vengono generati in modo non sicuro, lo schema risulta completamente vulnerabile [31].
Una seconda componente fondamentale per la sicurezza di RSA è la funzione di **codifica** del messaggio da applicare prima di generare la firma. RSASSA-PSS risulta sicuro ma alcune istanze della versione PKCS1-v1.5, che utilizza una semplice funzione di padding, risultano vulnerabili all'attacco di Bleichenbacher [32, 33].
L'altra importante minaccia a RSA, specialmente alle implementazioni hardware, è costituita dagli attacchi **side channel**. Se un attaccante fosse in grado di ottenere informazioni sulle tempistiche richieste dai singoli passi della generazione della firma, allora potrebbe essere in grado di ricostruire l'intera chiave privata. Uno dei primi attacchi di questo tipo risale al 1996 ed è dovuto a Kocher [34], ma negli anni successivi sono state trovate altre strategie molto più efficienti, in particolare se si utilizza il CRT per ottimizzare la firma [35]. Altri attacchi side channel più recenti sfruttano le informazioni ricavate dalla BPU

(Branch Prediction Unit) [36] o sull'introduzione di errori (fault injection), specialmente nella versione con il CRT [37].

4.2 Attacchi a DSA

In generale, l'implementazione di uno schema di tipo DSA risulta molto più semplice rispetto a RSA, che invece richiede di prestare attenzione a diversi dettagli per evitare di introdurre vulnerabilità. Se si considera anche il guadagno in termini di dimensioni di chiavi e firma ottenuto con ECDSA, è chiaro come questo schema abbia avuto grande successo e sia ora il più utilizzato. Risulta comunque importante seguire alcuni accorgimenti al fine di mantenere la sicurezza dello schema di firma.

4.2.1 Attacchi al problema del logaritmo discreto

Come già introdotto precedentemente, DSA e le sue varianti basano la propria sicurezza sulla complessità di risoluzione del **logaritmo discreto**. Per questo motivo la scelta del **gruppo** e del rispettivo **generatore** è fondamentale.
Lo schema DSA classico, o DSS, utilizza campi finiti, generalmente costituiti dagli interi tra 0 e $p-1$, dove p è un numero primo tale che $p-1$ abbia un fattore primo q molto grande. In particolare, si sceglie come generatore un elemento che abbia ordine q. Se la scelta di questi parametri non viene effettuata correttamente, è facile ricadere in istanze non sicure, ad esempio definendo un sottogruppo di ordine inferiore a quello previsto, dove è più facile risolvere il DLP. Per questo motivo esistono dei campi standard con annesso generatore in base al livello di sicurezza, espresso tramite la lunghezza in bit dei primi p e q.
Per ECDSA e EdDSA il discorso è analogo: serve una curva ellittica definita su un campo finito con un numero primo p di elementi e un generatore di ordine primo q. Come prima, è fondamentale conoscere a fondo le proprietà di tale costruzione al fine di evitare di ricadere in casi vulnerabili: ogni curva ellittica su un campo finito è composta da un numero finito di punti n, dipendente dai parametri scelti, che generalmente non è primo. Per questo motivo, quando si sceglie il generatore è importante controllare prima di tutto che abbia ordine primo q, e inoltre che il suo **cofattore**, cioè $c = n/q$, sia più piccolo possibile in modo da massimizzare il livello di sicurezza. Tuttavia, non sempre è semplice determinare tali quantità e quindi, per ottenere istanze di ECDLP sicure, si considerano curve standard ampiamente studiate, come quelle consigliate in Tabella 2.
Se si scelgono correttamente i parametri dello schema crittografico, gli algoritmi per risolvere un'istanza generica di DLP sono poco efficienti. Tra i più importanti si hanno:

- metodi che consentono di migliorare l'algoritmo di forza bruta su gruppi qualsiasi, come baby-step giant-step o rho di Pollard [38];
- algoritmi più efficienti se l'ordine del generatore ha fattori piccoli, come Pohlig-Hellman [39];
- soluzioni per gruppi specifici, come l'algoritmo index calculus dedicato ai campi finiti [40], in particolare nelle varianti di Coppersmith per $\mathbb{F}_{2^n}$ [41] e number field sieve per $\mathbb{F}_p$ [42].

Questi ultimi sono i migliori attacchi esistenti che, nonostante abbiano complessità solamente **subesponenziale**, hanno portato all'esclusione di DSS dagli standard.

4.2.2 Attacchi all'implementazione dello schema

Per la sicurezza di tutte le varianti di DSA è fondamentale che il **nonce** utilizzato per firmare resti segreto e venga generato correttamente ad ogni istanza di firma. Nel caso in cui si verifichi anche solo una tra le seguenti condizioni:

- il generatore di numeri casuali utilizzato non è crittograficamente sicuro;
- si utilizza lo stesso nonce per anche solo due firme diverse;
- alcuni bit del nonce risultano pubblici;

allora un attaccante può ricostruire la chiave segreta e la sicurezza dello schema è altamente compromessa.

4.3 Attacchi quantistici

Come già osservato, sia il problema della fattorizzazione che quello del logaritmo discreto sono minacciati dal possibile arrivo dei computer quantistici causa dell'**algoritmo di Shor** [7] che permette di risolverli in tempo polinomiale.
Nonostante i computer quantistici esistenti non siano in grado di rompere la crittografia a chiave pubblica attualmente utilizzata, schemi come RSASSA-PSS, ECDSA e EdDSA devono essere sostituiti al più presto con le soluzioni resistenti ad attacchi quantistici, come gli schemi **post-quantum** descritti nel capitolo successivo.

5 Schemi di firma post-quantum

Al fine di contrastare la minaccia quantistica, la comunità scientifica sta ricercando dei nuovi schemi che siano resistenti sia agli attacchi che utilizzano la computazione classica, sia a quelli che utilizzano i computer quantistici, detti crittosistemi post-quantum. Il principale promotore in questo ampio campo di ricerca è stato il National Institute of Standard and Technology (NIST) Statunitense che nel 2016 ha avviato un processo di selezione [43], attivo ancora oggi, che ha lo scopo di stabilire nuovi schemi standard per lo scambio di chiave e per la firma digitale. Ad oggi, per quanto riguarda le firme digitali, i sistemi crittografici risultati vincitori sono tre [44]. A questi si aggiungono due schemi basati su funzioni di hash già standardizzati prima della competizione [45], anch'essi quantum-safe.

5.1 Schemi di firma stateful basati su funzioni di hash

Le funzioni di hash sono una primitiva crittografica fondamentale per assicurare l'integrità dei dati. Dal punto di vista quantistico, l'unica minaccia alle funzioni di hash è data dall'algoritmo di Grover [46], che garantisce solo un aumento quadratico della velocità degli attacchi di ricostruzione della preimmagine. Quindi, per mantenere lo stesso livello di sicurezza attuale, è sufficiente raddoppiare la lunghezza dei digest, come specificato nel documento dedicato alle funzioni di hash [12]. Grazie a questa sicurezza consolidata, tali funzioni sono state utilizzate come basi costruttive di schemi di firma post-quantum che possono sostituire **direttamente** gli schemi classici attualmente in uso. Tra questi, i primi introdotti rientrano nella tipologia **stateful** [47]: schemi caratterizzati dall'impiego di molteplici chiavi effimere, cioè utilizzabili una singola volta, in cui è necessario tenere conto delle chiavi utilizzate per evitare che una stessa chiave venga usata più volte. Nello specifico, gli schemi stateful attualmente standardizzati sono due: **LMS** (Leighton-Micali Signature) [48] e **XMSS** (eXtended Merkle Signature Scheme) [49].
Questi due schemi sono molto simili: si utilizza uno schema di firma con chiave effimera, chiamato **OTS** (dall'inglese One-Time Signature); quindi, per poter firmare più messaggi, si generano molteplici chiavi effimere, le quali vengono combinate in un'unica coppia di chiavi tramite un metodo che consente di tenere traccia di ogni chiave effimera utilizzata.

5.1.1 Schema OTS di Winternitz

Come OTS, entrambi gli schemi stateful trattati utilizzano varianti dello schema di Winternitz, con parametri:

- una funzione di hash o XOF h, con digest di n bit;
- un intero w che rappresenta la quantità di bit del digest che vengono processati a ogni iterazione*.

*Nelle specifiche di XMSS, w rappresenta il numero di iterazioni. In questo documento si adotta la notazione di LMS anche per XMSS, quindi w è la lunghezza in bit dei blocchi mentre il numero di iterazioni è 2^w.

A questo punto, per prima cosa si genera casualmente la **chiave privata** $K = k_1 \| \ldots \| k_p$ in cui ogni k_j è una stringa di n bit. A ogni k_j viene applicata h per $2^w - 1$ volte in modo da ottenere un egual numero di blocchi che compongono la **chiave pubblica** $PK = pk_1 \| \ldots \| pk_p$ dove

$$pk_j = h(h(\ldots h(k_j))) = h^{2^w-1}(k_j).$$

Per firmare un messaggio m, si calcola il suo digest e lo si divide in b blocchi da w bit ottenendo

$$h(m) = m_1 \| m_2 \| \ldots \| m_b,$$

ai quali si affianca una somma di controllo (checksum) lunga c blocchi data da

$$\sum_{i=1}^{b}(2^w - 1 - m_i) = m_{b+1} \| \ldots \| m_{b+c}.$$

La **firma** si genera quindi come $\sigma(m) = \sigma_1 \| \ldots \| \sigma_p$ dove i blocchi sono i digest

$$\sigma_j = h^{m_j}(k_j).$$

Da osservare come la somma di controllo risulti fondamentale per la resistenza alla falsificazione in quanto, senza di essa, un attaccante può facilmente generare firme valide per tutti i digest con blocchi $m'_j \geq m_j$.
La verifica avviene semplicemente ottenendo i blocchi m_j del digest a partire dal messaggio ricevuto e valutando se

$$h^{2^w-1-m_j}(\sigma_j) = pk_j,$$

per ogni $1 \leq j \leq p$, sono equazioni valide.

5.1.2 Alberi di Merkle

Il metodo più utilizzato per raccogliere molteplici chiavi di uno schema OTS in un'unica coppia di chiavi a lungo termine sono gli alberi di Merkle [50]. Tramite questa struttura è possibile utilizzare la stessa coppia di chiavi per firmare grandi quantità di messaggi, seppur limitate.
La **chiave privata**, composta da tutte le chiavi private delle OTS, non richiede nessuna particolare elaborazione ma deve essere salvata in modo da poter mantenere traccia delle chiavi già utilizzate.
Per la chiave pubblica a lungo termine, invece di eseguire una semplice concatenazione di molte chiavi OTS, che risulterebbe molto dispendioso dal punto di vista dei dati da condividere, si costruisce una struttura ad albero in cui ogni nodo contiene il digest ottenuto dalla concatenazione dei due nodi precedenti. I nodi di partenza, detti foglie, sono ottenuti applicando la funzione di hash h alle varie chiavi pubbliche.
L'ultimo nodo, detto radice, è un digest dipendente da tutti i precedenti nodi che rappresenta la **chiave pubblica**. Il parametro che definisce un albero è la sua altezza, cioè il numero t di passaggi per arrivare da una foglia alla radice. Ne consegue che le foglie saranno 2^t, che rappresenta anche la quantità di firme calcolabili prima di dover generare una nuova coppia di chiavi.
In Figura 2 si rappresenta un esempio di albero di Merkle con altezza $t = 3$ e quindi $2^3 = 8$ foglie. La chiave pubblica risultante è la radice $h_{1:8}$.
Il guadagno nella dimensione della chiave viene compensato da una firma più lunga. In particolare, supponendo di aver generato una OTS $\sigma(m)$ con la coppia di chiavi OTS K_i, PK_i, la **firma** sarà della forma

$$i \| \sigma(m) \| path_1 \| \ldots \| path_t,$$

dove i $path_j$ sono informazioni aggiuntive necessarie affinché il nodo rappresentante la chiave pubblica validi la PK_i ricostruita in fase di verifica.

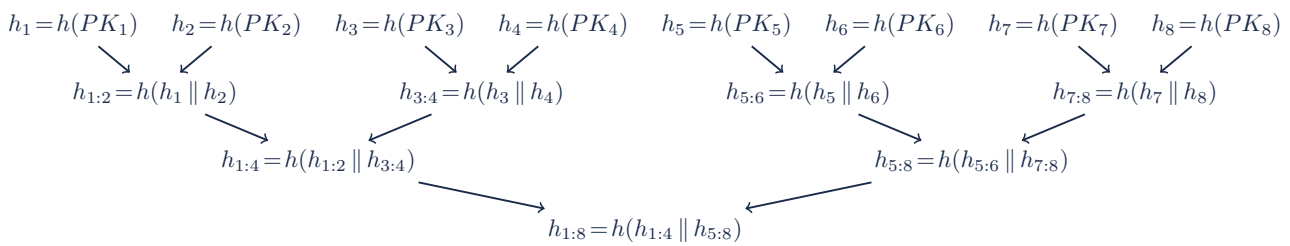


Figura 2 - Albero di Merkle di altezza 3

Nello specifico, i $path_j$ sono i valori di t nodi: per ogni livello, ognuno è il valore complementare a quello ottenibile a partire da PK_i, in modo che chiunque sia in grado di ricostruire il nodo radice e verificare che coincida con il valore pubblicato precedentemente. Considerando sempre l'esempio in Figura 2, se la firma è stata generata con la chiave OTS numero 5, allora in fase di verifica si ricostruisce PK_5 e il firmatario dovrà fornire:

- $path_1 = h_6$ così che si possa calcolare $h_{5:6} = h(h(PK_5) \,\|\, h_6)$;
- $path_2 = h_{7:8}$ così che si possa calcolare $h_{5:8}$;
- $path_3 = h_{1:4}$ così che si possa calcolare $h_{1:8}$ e verificare che sia uguale alla chiave pubblica.

Chiaramente la verifica è molto rapida, a scapito dei tempi di generazione della chiave pubblica. Si osserva inoltre come, affinché questa costruzione risulti sicura, sia fondamentale scegliere una funzione di hash h crittograficamente sicura e, in particolar modo, che risulti resistente alla seconda preimmagine (si faccia riferimento al documento dedicato [12]): se un attaccante fosse in grado di ottenere i digest che determinano i vari percorsi di ricostruzione della chiave pubblica, ma partendo da input diversi da quelli utilizzati dal firmatario, allora sarebbe capace di falsificare illecitamente una firma valida.

5.1.3 Struttura multi-albero

Nel caso in cui si intenda o sia necessario ridurre la complessità della rigenerazione dell'albero, da effettuare una volta esaurite le chiavi OTS, è possibile adottare una soluzione **gerarchica** o **multi-albero**. La struttura risultante è composta da più alberi di altezza t organizzati in d livelli in modo che:

- il livello ℓ-esimo con $0 \leq \ell \leq d$ contiene $2^{t\ell}$ alberi;
- la radice dell'unico albero al livello 0 è la chiave pubblica a lungo termine dello schema;
- le foglie di ogni livello corrispondono a chiavi OTS da utilizzare per firmare le radici degli alberi del livello successivo, quelle dell'ultimo livello sono per la firma del messaggio.

In questo modo, la lunghezza delle chiavi e della firma, così come i tempi di generazione e verifica della firma, restano praticamente uguali a quelli di uno schema con un singolo albero di altezza td.

Il vantaggio del multi-albero è che, una volta esaurite le chiavi OTS per la firma dei messaggi, è sufficiente aggiornare il corrispondente albero dell'ultimo livello e quindi si evita di dover rigenerare l'intera struttura.

5.1.4 Leighton-Micali Signature (LMS)

Lo schema di firma LMS [47, 48] utilizza le due componenti descritte apportando alcune modifiche.

Lo schema OTS, chiamato **LM-OTS**, presenta le seguenti caratteristiche:

- digest di 192 o 256 bit;
- blocchi di lunghezza w pari a 1, 2, 4 o 16;
- per garantire ulteriore sicurezza, ogni volta che viene calcolato un digest nella generazione della chiave pubblica OTS, si aggiunge all'input della funzione di hash h un prefisso contenente un valore I che identifica l'istanza (cioè la chiave a lungo termine) e gli indici relativi al numero della coppia di chiavi utilizzata, al blocco di chiave segreta e al numero di iterazione;
- in fase di firma, l'input per ottenere il digest iniziale ha un prefisso contenente il valore I, l'indice della chiave e un salt S generato casualmente, inserito poi in chiaro nella firma σ.

Per quanto riguarda l'albero di Merkle vale che:

- l'altezza è un valore t pari a 5, 10, 15, 20 o 25;
- si associa un'etichetta ad ogni nodo: questa vale 0 per la radice e cresce fino ad arrivare a $2^{t+1} - 1$ per il digest dell'ultima chiave pubblica OTS;
- nella generazione della chiave pubblica, ogni volta che si calcola un digest, si aggiunge all'input della funzione di hash un prefisso contenente il valore I e l'etichetta del nodo interessato. Inoltre, la chiave pubblica contiene il valore I che la identifica in chiaro.

Ricapitolando, lo schema LMS presenta:

- una **chiave pubblica** contenente la radice dell'albero di Merkle che raccoglie le chiavi OTS e l'identificativo I dell'istanza, più altri dati ausiliari;
- una **firma** composta dall'indice i della chiave OTS usata, dalla firma OTS $\sigma(m)$, dal salt S e dalle informazioni dell'albero di Markle necessarie per la verifica della validità, oltre ad alcuni dati ausiliari.

Per tutti i parametri raccomandati si faccia riferimento alla Tabella 4.

Schema	Sicurezza	Hash/XOF (h)	Lunghezza blocchi (w)	Blocchi di digest (b)	Blocchi totali (p)	Altezza albero (t)
LMS	192 bit	SHA-256/192, SHAKE256/192	1	192	200	5, 10, 15, 20, 25
			2	96	101	
			4	48	51	
			8	24	26	
	256 bit	SHA-256, SHAKE256	1	256	265	
			2	128	133	
			4	64	67	
			8	32	34	
XMSS	192 bit	SHA-256/192, SHAKE256/192	2	96	101	10, 16, 20
			4	48	51	
	256 bit	SHA-256, SHAKE256	2	128	133	
			4	64	67	

Tabella 4 - Parametri raccomandati per gli schemi di firma stateful basati su funzioni di hash

La versione multi-albero di LMS viene chiamata **HSS** (Hierarchical Signature System) e sono previsti da 1 a 8 livelli, ognuno con alberi di una tra le altezze supportate. Nello specifico, per HSS:

- la chiave pubblica è la concatenazione tra il numero d di livelli e la chiave pubblica del livello 0;
- la firma è composta dal valore di d, dalle $d-1$ chiavi pubbliche intermedie ognuna affiancata dalla rispettiva firma LMS sulla chiave del livello successivo e, infine, dalla firma LMS sul messaggio, ottenuta con la chiave dell'ultimo livello. Quindi, in fase di verifica, è possibile controllare la validità delle firme livello per livello.

5.1.5 eXtended Merkle Signature Scheme (XMSS)

Anche XMSS [47, 49] si basa sulle componenti sopra introdotte, con alcune modifiche specifiche.
Come OTS si utilizza un miglioramento dello schema di Winternitz chiamato **WOTS$^+$**, caratterizzato da:

- digest di 192 o 256 bit;
- blocchi di lunghezza w pari a 2 o 4;
- nella generazione della chiave pubblica OTS e della firma, il calcolo dei digest si aggiunge all'input un prefisso dipendente da un valore $SEED$ utilizzato per generare la chiave a lungo termine e un indirizzo $ADRS$ associato alla singola istanza di WOTS$^+$. Inoltre, l'input viene offuscato calcolando lo XOR con una maschera chiamata bitmask, anch'essa dipendente da $SEED$ e $ADRS$.

La costruzione dell'albero di Merkle presenta le seguenti caratteristiche:

- l'altezza è un valore t pari a 10, 16, 20;
- come nella OTS, i digest vengono modificati con un prefisso dipendente da $SEED$ e $ADRS$ e due maschere, una per ogni digest in input;

Infine, nella generazione della firma XMSS, WOTS$^+$ non si applica direttamente al messaggio da firmare, ma al digest ottenuto a partire dalla concatenazione del messaggio con la radice dell'albero di Merkle e un valore casuale r, che si include in chiaro nella firma finale.
In sintesi, i dati che lo schema richiede di inviare sono:

- una **chiave pubblica** composta dalla radice dell'albero di Merkle che raccoglie le chiavi OTS e dal valore $SEED$, più altri dati ausiliari;
- una **firma** data dalla concatenazione dell'indice i della chiave OTS usata con il valore casuale r, la firma OTS $\sigma(m)$ e le informazioni dell'albero di Merkle necessarie per la verifica della validità.

I parametri raccomandati sono raccolti nella Tabella 4.
Inoltre, XMSS prevede anche una versione multi-albero chiamata **XMSS^{MT}** nella quale il possibile numero di livelli dipende dall'altezza degli alberi: 4, 8 o 12 se l'altezza è 5, seppur questa altezza non sia prevista per lo schema semplice, 2, 4 o 6 con 10 di altezza, oppure 2 o 3 se l'altezza è 20. In questo caso:

- la chiave pubblica è identica a quella di XMSS;
- la firma contiene l'indice i della chiave utilizzata per firmare il messaggio (appartenete a un albero dell'ultimo livello), il valore casuale r, le firme delle chiavi intermedie e quella del messaggio. In fase di verifica, partendo da quest'ultima firma, si risale livello per livello ricostruendo le chiavi intermedie fino a raggiungere quella finale da confrontare con la chiave pubblica.

5.2 Schema di firma stateless basato su funzioni di hash

Uno dei nuovi schemi di firma digitale standardizzati in seguito alla selezione del NIST sfrutta principi simili a quelli introdotti per LMS e XMSS, ma rientra nella tipologia **stateless** in quanto non è necessario tenere traccia delle chiavi utilizzate nelle istanze precedenti. Il nome con cui è stato presentato alla competizione del NIST è SPHINCS$^+$ [51], standardizzato poi come **SLH-DSA** (StateLess Hash-based DSA) [52].
Lo schema SLH-DSA utilizza due firme basate su hash: la prima, chiamata **FORS**, consente di cifrare con la stessa chiave una discreta quantità di messaggi; la seconda è lo schema di firma effimera **XMSS^{MT}**, che utilizza molteplici firme effimere WOTS$^+$ raccolte in multi-alberi di Merkle. Nello specifico, in base al messaggio da firmare si sceglie una tra le chiavi WOTS$^+$ che viene utilizzata per validare una specifica chiave FORS, con la quale si genera la firma del messaggio.

5.2.1 Schema di firma FORS

Lo schema FORS (Forest Of Random Subsets) permette di generare più firme con la stessa chiave, ma in numero limitato siccome la sicurezza si degrada con ogni utilizzo. Questo tipo di schemi viene chiamato **FTS** (Few Times Signature). Come per le firme OTS introdotte precedentemente, le chiavi vengono gestite tramite una funzione di hash e la struttura degli alberi di Merkle, in modo da fornire sicurezza e minimizzare lo spazio di memoria necessario.
Nello specifico, i parametri di FORS sono:

- una funzione di hash o XOF h con digest di n bit;
- un intero k rappresentante la quantità di alberi di Merkle da generare;
- un intero a che determina l'altezza di tali alberi.

Quindi, si applica h a ognuna di queste stringhe, ottenendo $k2^a$ digest che vengono utilizzati come foglie di k alberi di Merkle di altezza a. Le radici così generate vengono concatenate e il digest tramite h della stringa risultante è la **chiave pubblica** di FORS.
In fase di firma, un messaggio m lungo ka bit viene scomposto in k stringhe $m_1, \ldots, m_k$ di a bit ciascuna e la **firma** è la concatenazione delle k stringhe

$$k_{1,m_1} \| k_{2,m_2} \| \ldots \| k_{k,m_k},$$

prese dagli alberi di Merkle della chiave pubblica. Inoltre, si associa a ognuna di esse il rispettivo percorso di autenticazione come descritto nella sezione 5.1.2.
In questo modo, la fase di verifica consiste semplicemente nel risalire alla chiave pubblica, utilizzando gli alberi di Merkle per ricavare le radici.
Dal punto di vista della sicurezza, nonostante a ogni istanza di firma si pubblichi una chiave privata per ogni albero di Merkle, è possibile generare diverse firme prima di compromettere lo schema, sempre supponendo di aver adottato una funzione di hash resistente alla seconda preimmagine (si veda il documento dedicato [12]). Infatti, un attaccante può contraffare la firma di un messaggio solo se il suo digest è composto da blocchi contenuti in firme già pubblicate, ma ciò è estremamente improbabile se sono state prodotte solo un numero limitato di firme rispetto alla grandezza degli alberi.

5.2.2 Generazione delle chiavi SLH-DSA

Come già anticipato, in SLH-DSA il messaggio viene firmato tramite lo schema FORS e la chiave utilizzata viene autenticata tramite XMSS^{MT}, quindi con una delle chiavi WOTS$^+$ raccolte in un multi-albero. La struttura risultante è rappresentata in Figura 3.

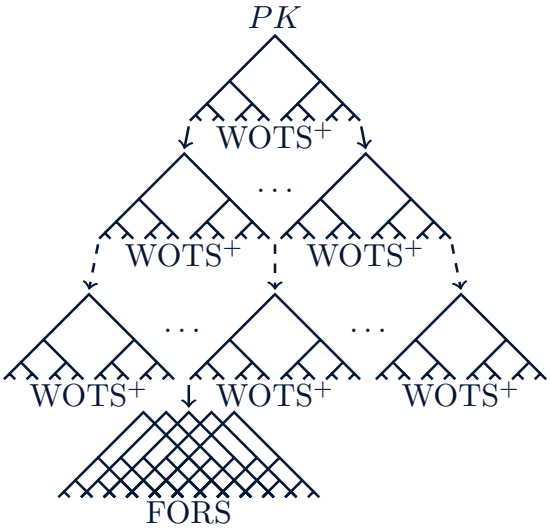


Figura 3 - Rappresentazione della combinazione di firme in SLH-DSA

I parametri sono quelli richiesti dalle singole componenti:

- la funzione di hash o XOF h da utilizzare, eventualmente troncando i digest alla lunghezza desiderata. Lo standard utilizza SHA-256 o SHAKE256.
- per WOTS$^+$ la lunghezza n del digest, i blocchi sono sempre di lunghezza $w = 4$;
- per il multi-albero di XMSS^{MT}, l'altezza t dei singoli alberi e il numero d di livelli;
- per FORS, il numero di alberi k e la loro altezza a.

La **chiave pubblica** è semplicemente quella di XMSS^{MT}, cioè la radice PK dell'albero al livello 0, mentre la **chiave privata** è composta dalle chiavi private delle firme FORS e WOTS$^+$, o semplicemente il seed utilizzato per generarle.

5.2.3 Generazione e verifica della firma SLH-DSA

Quando si intende firmare un messaggio, per prima cosa si applica una funzione di hash o XOF in modo che il digest abbia la corretta lunghezza, ottenuta specificamente come

$$td + ka,$$

dove il primo addendo è la quantità di bit dedicati alla scelta di una chiave FORS tra quelle associate alle foglie finali del multi-albero di XMSS^{MT}.
Infatti, $t(d-1)$ bit servono per identificare uno degli $2^{t(d-1)}$ alberi appartenenti all'ultimo livello, mentre t bit identificano una tra le sue 2^t foglie, utilizzata per autenticare una specifica chiave pubblica FORS. Il secondo addendo rappresenta la quantità di bit che verranno firmati tramite lo schema FORS.
Una volta identificata la chiave pubblica e la rispettiva chiave privata da utilizzare, si applica lo schema FORS agli ultimi ka bit del digest e la **firma** SLH-DSA risultante è composta da:

- la firma FORS sul digest del messaggio;
- la firma WOTS$^+$ sulla chiave pubblica FORS;
- le firme WOTS$^+$ sulle radici coinvolte in XMSS^{MT};
- i percorsi associati per la validazione delle varie firme.

La verifica avviene semplicemente ricostruendo la chiave pubblica PK a partire dalle informazioni contenute nella firma, seguendo prima le operazioni richieste da FORS e poi quelle di XMSS^{MT}.

5.2.4 Sicurezza di SLH-DSA

Per i valori dei parametri raccomandati si veda la Tabella 5. In particolare, i parametri di SLH-DSA sono organizzati nei livelli di sicurezza 1, 3 e 5, definiti dal NIST come comparabili alla sicurezza di AES con chiavi da 128, 192 e 256 bit, rispettivamente. Per ognuno di questi livelli, esistono due possibili insiemi di parametri che definiscono due varianti, **small** (s) e **fast** (f), che prediligono la dimensione dei dati e velocità di computazione, rispettivamente. Per qualsiasi scelta tra i parametri standardizzati, si prevede di poter eseguire in modo sicuro fino a 2^{64} istanze di firma.

Sicurezza	Lunghezza digest (n)	Variante	Altezza WOTS$^+$ (t)	Livelli XMSS^{MT} (d)	Alberi FORS (k)	Altezza FORS (a)
1	128	small (s)	9	7	14	12
		fast (f)	3	22	33	6
3	192	small (s)	9	7	17	14
		fast (f)	3	22	33	8
5	256	small (s)	8	8	22	14
		fast (f)	4	17	35	9

Tabella 5 - Parametri raccomandati per SLH-DSA

5.3 Schemi di firma basati su reticoli

Oltre agli schemi basati su hash, il NIST ha selezionato due schemi di firma post-quantum che basano la propria sicurezza su problemi derivanti da strutture matematiche chiamate **reticoli** (lattices, in inglese).
Nonostante entrambe le soluzioni siano state ampiamente studiate dalla comunità scientifica, la pubblicazione dello standard di uno dei due schemi, chiamato Falcon [53], è stata posticipata per problematiche relative alla sicurezza della sua implementazione.
L'altro schema, CRYSTALS-Dilithium [54], è invece diventato rapidamente uno standard chiamato **ML-DSA** (Module-Lattice-based DSA) [55] e, grazie alle sue caratteristiche molto competitive sia in termini di dimensioni di dati sia di prestazioni, ha riscosso fin da subito molto successo nel processo di transizione agli schemi post-quantum.

5.3.1 Sicurezza e problemi computazionali

Un reticolo è una struttura algebrica definita come un sottogruppo discreto dello spazio dei vettori reali. Gli elementi di un reticolo, che possono essere chiamati vettori come gli elementi dello spazio, possono essere sommati e sottratti rimanendo sempre all'interno del reticolo.
Di conseguenza, uno specifico reticolo può essere descritto efficientemente specificando solamente alcuni dei suoi vettori che consentono di generare qualsiasi altro vettore, e tale insieme viene chiamato **base**. Inoltre, è possibile definire una funzione di **lunghezza** e identificare un vettore che risulta il più corto tra tutti quelli del reticolo. Tuttavia, trovare questo vettore speciale è un processo computazionalmente difficile a meno di conoscere una particolare base del reticolo. Questo problema viene chiamato **Shortest Vector Problem (SVP)** ed è il motivo per cui si parla di schemi crittografici basati su reticoli.
Nello specifico, lo schema di firma ML-DSA non basa la propria sicurezza direttamente sulla complessità dello SVP, ma su quella di due problemi a esso riconducibili, entrambi formulati su numeri interi modulo un primo q, cioè in $\mathbb{Z}_q$.
Il primo problema è il cosiddetto **Learning With Errors (LWE)**: data una matrice $\boldsymbol{A} \in \mathbb{Z}_q^{k \times l}$ e un vettore $\boldsymbol{t} \in \mathbb{Z}_q^k$, trovare la soluzione $\boldsymbol{s} \in \mathbb{Z}_q^l$ del sistema di equazioni

$$\boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} = \boldsymbol{t},$$

con $\boldsymbol{e} \in \mathbb{Z}_q^k$ da interpretare come l'introduzione volontaria di un piccolo errore. In particolare, è possibile dimostrare matematicamente che un'istanza generica di LWE risulta tanto complessa quanto un'istanza difficile di SVP [56], per cui si ritiene che basare la sicurezza di uno schema sul LWE sia una scelta molto solida.
Il secondo problema alla base della sicurezza di ML-DSA è conosciuto come **Short Integer Solution (SIS)** e consiste nel trovare un vettore $\boldsymbol{x} \in \mathbb{Z}_q^l$ sufficientemente piccolo che risolva il sistema

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{0},$$

per una data matrice $\boldsymbol{A} \in \mathbb{Z}_q^{k \times l}$.
Anche in questo caso, un'istanza generica di SIS risulta tanto complessa quanto un'istanza difficile di SVP [57] e quindi la sicurezza basata su SIS è considerata molto solida.

Andando più nel particolare, la sicurezza di ML-DSA si basa su particolari istanze di LWE e SIS che, invece degli interi $\mathbb{Z}_q$, utilizzano un più generico insieme R_q (dotato di due operazioni che rispettano particolari proprietà) e i problemi associati sono Module LWE (MLWE) e Module SIS (MSIS).

5.3.2 Generazione delle chiavi ML-DSA

Il livello di sicurezza desiderato definisce i valori assunti dai parametri dello schema, dati da:

- lo spazio $R_q = \mathbb{Z}_q[x]/(x^{256}+1)$ con $q = 2^{23} - 2^{13} + 1$, contenente i polinomi con variabile x, coefficienti in $\mathbb{Z}_q$ e grado al più 255;
- le dimensioni (k, l) della matrice associata al LWE;
- gli estremi γ, η che definiscono gli intervalli per i coefficienti dei polinomi in vettori specifici;
- un estremo α per le approssimazioni;
- un intero τ per indicare il massimo numero di zeri nei coefficienti del polinomio che rappresenta la firma;
- una funzione di hash o XOF $h(m, \ell)$ che restituisce un digest di m lungo ℓ bit. Lo standard utilizza SHAKE256 e un parametro λ pari alla sicurezza contro le collisioni.

La coppia di chiavi ML-DSA è quindi data da:

- **chiave privata** contenente due vettori $\boldsymbol{s} \in R_q^l, \boldsymbol{e} \in R_q^k$ composti da polinomi a coefficienti interi compresi tra $-\eta$ e η, estremi inclusi;
- **chiave pubblica** composta dalla matrice $\boldsymbol{A} \in R_q^{k \times l}$ e dal vettore $\boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} = \boldsymbol{t} \in R_q^k$.

Formalmente, la matrice viene salvata in modo da velocizzare le operazioni tra vettori, sfruttando una trasformazione matematica chiamata **Number Theoretic Transform (NTT)**. Sempre per ottimizzare le prestazioni, il vettore $\boldsymbol{t}$ viene pubblicato in una versione ridotta in cui si rimuovono i 13 bit meno significativi da ogni coefficiente.

5.3.3 Generazione della firma ML-DSA

Lo schema di ML-DSA è ispirato alla firma di Schnorr [22] in quanto utilizza la trasformata di Fiat-Shamir [58] per ottenere uno schema di firma a partire da un protocollo interattivo per provare la conoscenza di un segreto. In particolare, tale protocollo prevede tre passaggi:

- Alice vuole provare a Bob di conoscere $\boldsymbol{s} \in R_q^l$. Genera $\boldsymbol{y} \in R_q^l$ piccolo, calcola $\boldsymbol{w} = \boldsymbol{A} \cdot \boldsymbol{y}$, esclude la parte minore di α ottenendo un'approssimazione $\tilde{\boldsymbol{w}}$ e la invia a Bob;
- Bob genera un elemento di sfida $c \in R_q$ piccolo;
- in risposta Alice invia $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}$ e Bob deve verificare che sia un vettore sufficientemente piccolo con $\boldsymbol{A} \cdot \boldsymbol{z} - c\boldsymbol{t} \approx \tilde{\boldsymbol{w}}$. Questo dovrebbe corrispondere a prendere i bit meno significativi di

$$\boldsymbol{A} \cdot (\boldsymbol{y} + c\boldsymbol{s}) - c(\boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}) = \boldsymbol{A} \cdot \boldsymbol{y} - c\boldsymbol{e},$$

  in cui il secondo addendo dovrebbe essere sufficientemente piccolo da poter ricostruire l'approssimazione iniziale $\tilde{\boldsymbol{w}}$. Se questo vale, allora è altamente probabile che Alice conosca $\boldsymbol{s}$, e la prova è stata fornita senza doverlo pubblicare.

L'algoritmo di generazione della firma ML-DSA si ottiene rendendo il protocollo appena descritto non interattivo. Nello specifico, lo schema funziona in quanto la trasformata di Fiat-Shamir consente di generare in fase di verifica la stessa istanza del protocollo costruita in fase di firma. Per applicare la firma a un messaggio m si eseguono i seguenti passaggi:

- si calcola $H = h(h(\boldsymbol{A} \,\|\, \boldsymbol{t}, 512) \,\|\, m, 512)$;
- si genera casualmente un vettore $\boldsymbol{y} \in R_q^l$ con coefficienti interi compresi tra $-\gamma + 1$ e γ, estremi inclusi, da cui si ottiene $\boldsymbol{A} \cdot \boldsymbol{y} \approx \tilde{\boldsymbol{w}}$ come descritto precedentemente;
- si calcola $\tilde{c} = h(H \,\|\, \tilde{\boldsymbol{w}}, 2\lambda)$;
- utilizzando $\tilde{c}$ nell'inizializzazione di un generatore casuale, si genera $c \in R_q$ in modo che i suoi coefficienti valgano ± 1 e quelli non nulli siano al più τ;
- si calcola $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}$;
- la **firma** è data dalla coppia $(\tilde{c}, \boldsymbol{z})$.

Nelle specifiche, oltre a includere vari controlli per confermare la correttezza dei valori calcolati, la firma comprende anche un ulteriore vettore informativo da utilizzare nell'algoritmo di verifica per ricostruire $\tilde{\boldsymbol{w}}$ quando la chiave pubblica contiene la versione ridotta di $\boldsymbol{t}$.

5.3.4 Verifica della firma ML-DSA

L'algoritmo di verifica della firma controlla che i valori ricevuti rispettino i vincoli posti dal protocollo interno. Data la chiave pubblica $(\boldsymbol{A}, \boldsymbol{t})$ e il messaggio m, per validare la firma $(\tilde{c}, \boldsymbol{z})$:

- si calcola $H = h(h(\boldsymbol{A} \,\|\, \boldsymbol{t}, 512) \,\|\, m, 512)$;
- utilizzando $\tilde{c}$ si genera lo stesso $c \in R_q$ ottenuto al passo 4 della generazione della firma;

- si calcola $\boldsymbol{w'} = \boldsymbol{A} \cdot \boldsymbol{z} - c\boldsymbol{t}$ e si approssima a $\tilde{\boldsymbol{w}}'$ escludendo la parte minore di α;
- la firma è valida se i polinomi in $\boldsymbol{z}$ hanno coefficienti minori di $\gamma - \eta\tau$ e vale che $\tilde{\boldsymbol{w}} = \tilde{\boldsymbol{w}}'$.

Come già anticipato, se la chiave pubblica fornisce $\boldsymbol{t}$ in versione ridotta, allora la firma deve contenere un ulteriore vettore di informazione che consenta di ricostruire $\tilde{\boldsymbol{w}}$.

5.3.5 Sicurezza di ML-DSA

Per i valori dei parametri raccomandati si veda la Tabella 6. In particolare, i parametri di ML-DSA sono organizzati nei livelli di sicurezza 2, 3 e 5, definiti dal NIST come comparabili alla sicurezza di SHA-256 (rispetto alle collisioni) e di AES con chiavi da 192 e 256 bit, rispettivamente.

5.4 Schemi ibridi

Nonostante l'approfondito studio da parte della comunità scientifica e l'impegno dedicato dagli enti di standardizzazione nel definire nuovi schemi di firma sicuri, solo la prova del tempo potrà fornire una maggiore fiducia nel loro utilizzo. Infatti, anche schemi ritenuti sicuri da molti anni potrebbero rivelarsi vulnerabili a causa dell'inaspettata scoperta di nuove tecniche di crittoanalisi. Ne è un esempio il cifrario Rainbow, candidato alla standardizzazione del NIST arrivato al terzo round di selezione, la cui sicurezza si basava su un problema di sistemi multivariati, che si è rivelato insicuro solo dopo 16 anni dalla sua prima pubblicazione [59].

Nelle fasi iniziali della transizione, la soluzione più sicura e promettente è adottare schemi di firma **ibridi** ottenuti utilizzando contemporaneamente uno schema post-quantum e uno classico. In questo modo, da una parte, si guadagna la sicurezza contro attacchi quantistici, mentre dall'altra si ha la sicurezza classica di uno schema ampiamente utilizzato e studiato da molti decenni. In particolare, per schemi basati su primitive con sicurezza consolidata come le funzioni di hash, l'ibridazione può essere opzionale. Al contrario, schemi che basano la propria sicurezza sulla complessità di nuovi problemi matematici, come **ML-DSA**, al momento richiedono ibridazione.

Per ottenere uno schema di firma ibrido è sufficiente affiancare a uno schema post-quantum uno degli schemi classici tra quelli descritti nel capitolo 3. Esistono diverse strategie per l'ibridazione in base ai contesti e ai requisiti [60]. In generale, si fornisce resistenza ad attacchi sia classici che quantistici in quanto la contraffazione ha successo solo riuscendo a falsificare entrambe le firme. Il livello di sicurezza dello schema ibrido è pari al minimo dei livelli di sicurezza delle due firme che lo compongono e quindi è importante scegliere parametri allo stesso livello.

Sicurezza	Dimensione matrice (k, l)	Estremo γ	Estremo η	Estremo α	Numero di zeri (τ)	Sicurezza hash (λ)
2	4, 4	2^{17}	2	$(q-1)/44$	39	128
3	6, 5	2^{19}	4	$(q-1)/16$	49	192
5	8, 7	2^{19}	2	$(q-1)/16$	69	256

Tabella 6 - Parametri raccomandati per ML-DSA

L'implementazione di ML-DSA tramite componenti **hardware** presenta complicazioni da tenere in considerazione: per prima cosa, è necessario costruire funzioni di campionamento che generino vettori della corretta lunghezza e, in secondo luogo, serve una componente dedicata alla NTT. In alcuni contesti specifici, le componenti hardware utilizzate attualmente non prevedono tali funzioni e quindi, per adottare ML-DSA, è necessario sostituirle.

6 Confronto tra gli schemi

In questa sezione si intende sottolineare le differenze tra gli schemi trattati nei capitoli precedenti tramite confronti di dimensioni dei dati trasmessi e prestazioni degli algoritmi di ogni schema per i vari livelli di sicurezza.

6.1 Dimensioni dei dati trasmessi

Nelle Tabelle 7 e 8 sono raccolte, rispettivamente, le **dimensioni in byte** delle chiavi pubbliche e delle firme generate dagli schemi trattati.
I valori presentati sono ottenuti teoricamente a partire dalle specifiche descritte nei capitoli precedenti. In particolare, per consentire la comparazione, tali valori sono suddivisi nei 3 livelli di sicurezza standard corrispondenti ad AES con chiavi di 128, 192 e 256 bit, seppur alcuni schemi presentino configurazioni intermedie contrassegnate con *. Nello specifico, i parametri più sicuri per EdDSA corrispondono a 224 bit di sicurezza mentre il primo livello di sicurezza per ML-DSA è comparabile a quella data dalla difficoltà di trovare una collisione per SHA-256, che corrisponde a una sicurezza intermedia tra 128 e 192 bit.
Le prime tre righe riguardano gli schemi classici, mentre le restanti corrispondono agli schemi post-quantum.
Per quanto riguarda le **chiavi pubbliche** in Tabella 7 si può osservare che:

- gli schemi basati su curve ellittiche e su hash presentano le dimensioni più contenute;
- RSASSA-PSS ha dimensioni intermedie, tra 10 e 30 volte quelle degli schemi al punto precedente;
- ML-DSA ha le dimensioni maggiori, pari a circa 40 volte quelle degli schemi al primo punto.

Considerando invece la Tabella 7 riguardante le **firme** generate, si può notare che:

- le firme basate su curve ellittiche sono le più contenute;
- RSASSA-PSS ha dimensioni tra 6 e 15 volte quelle degli schemi al punto precedente;
- ML-DSA ha dimensioni di circa 35 volte quelle degli schemi al primo punto;
- gli schemi stateful non presentano parametri per 128 bit di sicurezza ma hanno molteplici configurazioni per gli altri due livelli, per cui per ognuno si raccolgono gli estremi delle possibili dimensioni. L'incremento rispetto alle dimensioni al primo punto va indicativamente da 8 a 600 volte, per cui risulta necessario trovare un compromesso tra requisiti e prestazioni.
- SLH-DSA presenta incrementi da 120 a 230 volte per la variante small e da 260 a 390 volte per la versione fast. Anche in questo caso si richiede un compromesso, ma i valori minimi restano i maggiori tra tutti gli schemi.

Schema	128 bit	192 bit	256 bit
RSASSA-PSS	416	992	1 952
ECDSA	32	48	64
EdDSA	32	57*	-
LMS	-	48	56
HSS	-	52	60
XMSS	-	52	68
XMSS^{MT}	-	52	68
SLH-DSA (s)	32	48	64
SLH-DSA (f)	32	48	64
ML-DSA	1 312*	1 952	2 592

Tabella 7 - Dimensioni in byte della chiave pubblica per gli schemi presentati

Schema	128 bit	192 bit	256 bit
RSASSA-PSS	384	960	1 920
ECDSA	64	96	128
EdDSA	64	114*	-
LMS	-	780 - 5 436	1 292 - 9 324
HSS	-	784 - 43 828	1 296 - 74 988
XMSS	-	1 492 - 2 932	2 500 - 4 932
XMSS^{MT}	-	2 955 - 30 560	4 963 - 53 032
SLH-DSA (s)	7 856	16 224	29 792
SLH-DSA (f)	17 088	35 664	49 856
ML-DSA	2 420*	3 309	4 627

Tabella 8 - Dimensioni in byte della firma generata dagli schemi presentati

*Parametri con sicurezza superiore a quella della colonna.

6.2 Prestazioni degli algoritmi

Le Tabelle da 9 a 11 descrivono, rispettivamente, le **prestazioni** degli algoritmi di generazione delle chiavi, generazione della firma e verifica.

Come per la sezione precedente, gli schemi vengono comparati a parità di livello di sicurezza, seppur EdDSA e ML-DSA prevedono un livello superiore nei casi contrassegnati con *. Nuovamente, le prime tre righe riguardano gli schemi classici, mentre le restanti corrispondono agli schemi post-quantum.

Considerando il continuo sviluppo delle implementazioni per gli schemi post-quantum, si intende fornire una valutazione puramente indicativa delle prestazioni, la quale verrà aggiornata costantemente per valutare anche contesti specifici. Per questo motivo, al momento, le misurazioni sono state effettuate tramite la funzione "speed" di OpenSSL v3.5.5 che restituisce la velocità degli algoritmi in numero di esecuzioni al secondo. Tale libreria, seppur all'avanguardia, non contiene implementazioni di LMS e XMSS che quindi non vengono inclusi nel confronto.

Per rendere i dati indipendenti dalla macchina utilizzata, le velocità raccolte sono state normalizzate dividendole per il valore minimo in ogni colonna. Le tabelle contengono quindi, per ogni algoritmo specifico, i **rapporti delle velocità** rispetto allo schema meno efficiente nell'esecuzione di tale algoritmo (che avrà rapporto 1.00). Implementazioni dedicate o macchine con diversa potenza possono risultare in variazioni sui risultati della comparazione.

Per quanto riguarda la **generazione delle chiavi**, operazione che solitamente viene eseguita una sola volta per utente, la funzione di OpenSSL non fornisce valori per gli schemi basati su curve ellittiche che per questo risultano non disponibili (n.d.). In generale, dalla Tabella 9 si può osservare che:

- lo schema meno efficiente è RSASSA-PSS;
- SLH-DSA nella versione small ha prestazioni poco migliori del precedente, ma nella versione fast il miglioramento è consistente;
- ML-DSA risulta lo schema più efficiente.

Tuttavia, gli schemi basati su curve ellittiche possono essere considerati i più efficienti in assoluto, in quanto la generazione delle chiavi prevede solamente il campionamento di un numero casuale da moltiplicare per il punto base di una curva standard.

L'algoritmo di **generazione della firma** è il più importante tra le tre fasi dello schema in quanto deve essere eseguito molteplici volte e spesso costituisce un collo di bottiglia per le prestazioni. Da quanto si evince dalla Tabella 10 si ha che:

- lo schema meno efficiente è SLH-DSA, specialmente nella versione small che si incentra sul diminuire la dimensione della firma. La versione fast risulta più efficiente ma comunque inferiore a tutti gli altri schemi;
- RSASSA-PSS, nonostante sia uno schema classico, risulta poco efficiente;
- ML-DSA ha ottime prestazioni nonostante sia post-quantum;
- gli schemi basati su curve ellittiche sono i più efficienti.

Infine, i risultati sulle prestazioni per la **verifica della firma** sono raccolti in Tabella 11. Si può osservare che gli schemi hanno prestazioni tra loro comparabili ma, nello specifico:

- lo schema meno efficiente è nuovamente SLH-DSA, in particolare nella versione fast, mentre la versione short risulta poco più efficiente;
- gli schemi su curve ellittiche si comportano bene per 128 bit di sicurezza ma perdono di efficienza per i livelli più alti;
- ML-DSA ha in media le seconde prestazioni migliori;
- lo schema più efficiente è RSASSA-PSS, anche se a 256 bit di sicurezza risulta poco efficiente a causa dei numeri molto grandi da elaborare.

Per fornire un ulteriore confronto visivo tra le prestazioni degli schemi trattati, le Figure da 4 a 6 raccolgono le velocità ottenute tramite la funzione "speed" di OpenSSL v3.5.5 per generazione delle chiavi, generazione della firma e verifica, rispettivamente. In questo caso, le velocità non sono state normalizzate rispetto a quella minore come fatto nelle tabelle precedentemente descritte. Tuttavia, per poter apprezzare le differenze tra i vari ordini di grandezza, i grafici rappresentano i valori di esecuzioni al secondo in scala logaritmica.

Schema	128 bit	192 bit	256 bit
RSASSA-PSS	1,00	1,00	1,00
ECDSA (secp)	n.d.	n.d.	n.d.
EdDSA	n.d.	n.d.	-
SLH-DSA (SHA2-s)	3,66	9,50	621,76
SLH-DSA (SHA2-f)	233,83	601,62	9 868,07
ML-DSA	1 552,95*	3 277,14	101 707,38

Tabella 9 - Prestazioni dell'algoritmo di generazione delle chiavi per diversi schemi in rapporto con RSASSA-PSS

Schema	128 bit	192 bit	256 bit
RSASSA-PSS	168,26	152,94	2,87
ECDSA (secp)	13 523,92	3 596,52	2 528,91
EdDSA	8 453,38	3 655,39*	-
SLH-DSA (SHA2-s)	1,00	1,00	1,00
SLH-DSA (SHA2-f)	20,87	23,64	10,81
ML-DSA	593,46*	753,12	597,82

Tabella 10 - Prestazioni dell'algoritmo di generazione della firma per diversi schemi in rapporto con SLH-DSA (SHA2-s)

Schema	128 bit	192 bit	256 bit
RSASSA-PSS	22,29	21,24	1,59
ECDSA (secp)	11,92	3,54	3,25
EdDSA	8,61	7,29*	-
SLH-DSA (SHA2-s)	2,94	2,54	1,85
SLH-DSA (SHA2-f)	1,00	1,00	1,00
ML-DSA	8,15*	8,77	5,73

Tabella 11 - Prestazioni dell'algoritmo di verifica della firma per diversi schemi in rapporto con SLH-DSA (SHA2-f)

*Parametri con sicurezza superiore a quella della colonna.

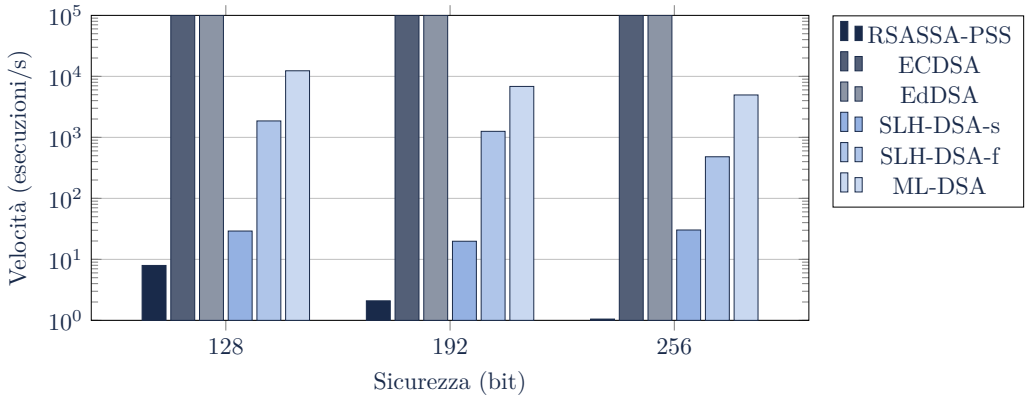


Figura 4 - Velocità dell'algoritmo di generazione delle chiavi per diversi schemi, in scala logaritmica

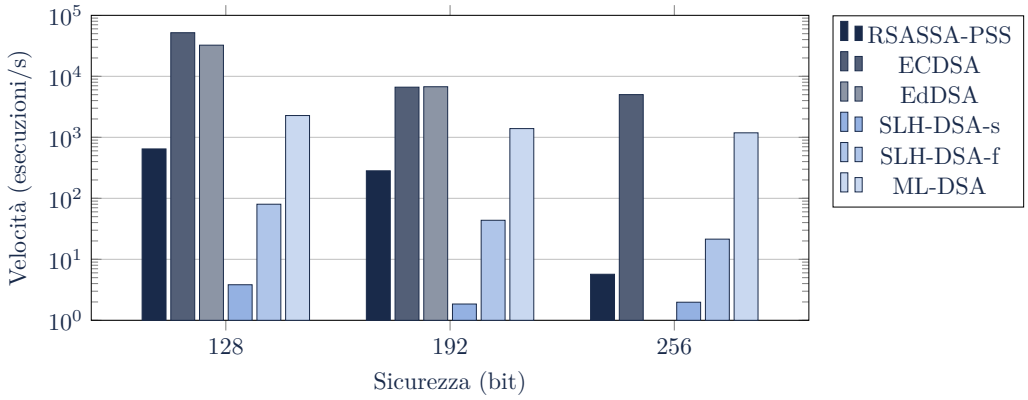


Figura 5 - Velocità dell'algoritmo di generazione della firma per diversi schemi, in scala logaritmica

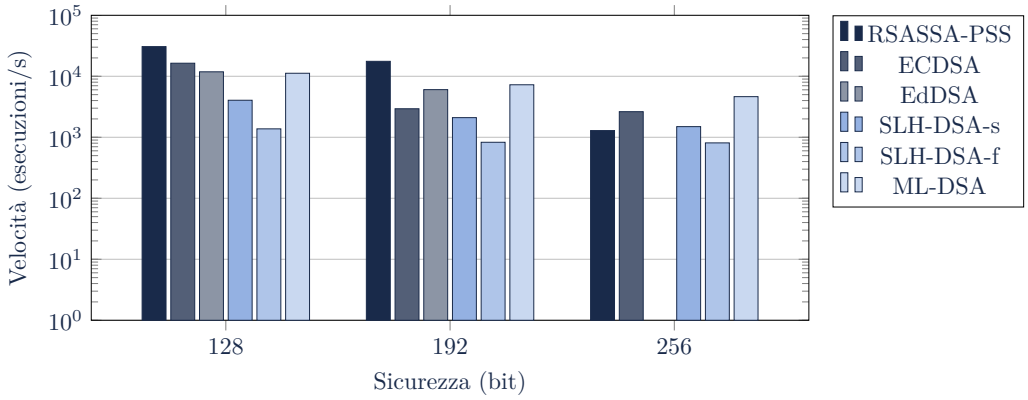


Figura 6 - Velocità dell'algoritmo di verifica della firma per diversi schemi, in scala logaritmica

7 Conclusioni

Quanto specificato nei capitoli precedenti intende fornire le informazioni necessarie per scegliere lo schema di firma da adottare. Questo dipende fortemente dal contesto di applicazione, in quanto ognuno degli schemi considerati presenta i propri vantaggi e svantaggi in termini di memoria e prestazioni, oltre che diversi livelli di sicurezza.
Le configurazioni degli schemi sono state organizzate in raccomandate e accettate, quest'ultime a loro volta suddivise in post-quantum e classiche.

7.1 Schemi di firma post-quantum raccomandati

In generale, si raccomanda di sostituire al più presto qualsiasi schema di firma classico con una soluzione post-quantum tra quelle raccolte in Tabella 12.
Per l'utilizzo di SLH-DSA, a parità di livello di sicurezza, la scelta tra la variante small (s) e la variante fast (f) deve essere effettuata tenendo in considerazione il contesto specifico di applicazione e la priorità tra dimensione della firma e prestazioni della sua generazione, rispettivamente. In entrambi i casi, l'ibridazione è opzionale.
Per quanto riguarda ML-DSA, data la novità del problema matematico alla base della sicurezza, si raccomanda di eseguire un'**ibridazione** affiancando uno schema classico con pari livello di sicurezza. Come osservato nella sezione dedicata, le implementazioni hardware di ML-DSA richiedono una componente dedicata alla NTT che al momento risulta altamente complicata da realizzare in modo corretto e affidabile.
Le configurazioni degli schemi post-quantum con livello di sicurezza inferiore a 192 bit non sono raccomandate in quanto il processo di sostituzione e integrazione si prospetta oneroso e 128 bit di sicurezza potrebbero non essere sufficienti nel futuro prossimo.

7.2 Schemi di firma post-quantum accettati

Gli schemi di firma digitale post-quantum accettati sono raccolti nella Tabella 13.
Tra questi si includono i primi livelli di sicurezza di SLH-DSA e ML-DSA, non raccomandati per le motivazioni fornite nella sezione precedente ma comunque ritenuti sicuri.
Inoltre si accettano LMS e XMSS nelle versioni ad albero singolo, seppur per i parametri specifici si rimanda a una valutazione del contesto di applicazione. Si osservi come tali soluzioni presentino limitazioni date dalla tipologia stateful, che richiede una gestione delle chiavi oculata e dedicata.
Come per SLH-DSA, l'ibridazione è opzionale.

7.3 Schemi di firma classici accettati

Gli schemi classici sono vulnerabili agli attacchi quantistici e pertanto è necessario provvedere il prima possibile alla transizione verso le soluzioni post-quantum raccomandate in Tabella 12.
Di conseguenza, le configurazioni raccolte in Tabella 14 risultano accettate solo per le fasi intermedie della transizione e sono considerate valide opzioni da affiancare agli schemi post-quantum per ottenere soluzioni ibride.

Schema	Parametri	Sicurezza	Note
SLH-DSA	SHA2-192s	192 bit	s = dimensione firma ridotta f = generazione firma veloce Ibridazione opzionale
	SHA2-192f		
	SHAKE-192s		
	SHAKE-192f		
	SHA2-256s	256 bit	
	SHA2-256f		
	SHAKE-256s		
	SHAKE-256f		
ML-DSA	6, 5	192 bit	Ibridazione raccomandata Componenti hardware dedicate
	8, 7	256 bit	

Tabella 12 - Schemi di firma post-quantum raccomandati con relativi parametri

Schema	Parametri	Sicurezza	Note
SLH-DSA	SHA2-128s	128 bit	s = dimensione firma ridotta f = generazione firma veloce Ibridazione opzionale
	SHA2-128f		
	SHAKE-128s		
	SHAKE-128f		
ML-DSA	4, 4	128 bit*	Ibridazione raccomandata Componenti hardware dedicate
LMS	SHA-256/192	192 bit	Gestione chiavi complessa Ibridazione opzionale
	SHAKE256/192		
	SHA-256	256 bit	
	SHAKE256		
XMSS	SHA-256/192 SHAKE256/192	192 bit	
	SHA-256	256 bit	
	SHAKE256		

Tabella 13 - Schemi di firma post-quantum accettati con relativi parametri

*Il primo livello di sicurezza per ML-DSA è comparabile a quella data dalla difficoltà di trovare una collisione per SHA-256, che corrisponde a una sicurezza intermedia tra 128 e 192 bit.

Schema	Parametri	Sicurezza	Note
RSASSA-PSS	$n \geq 3072$	128 bit	Vulnerabili agli attacchi quantistici Raccomandati per ibridazione
ECDSA	secp256r1	128 bit	
	brainpoolP256r1		
	secp384r1	192 bit	
	brainpoolP384r1		
	secp521r1	256 bit	
	brainpoolP512r1		
EdDSA	Curve25519	128 bit	
	Curve448	224 bit	

Tabella 14 - Schemi di firma classici accettati con relativi parametri

Bibliografia

[1] ACN. *Introduzione alla Crittografia e alle Linee Guida*. Linee Guida Funzioni Crittografiche, 2026. URL: https://www.acn.gov.it/portale/crittografia.

[2] W. Diffie e M. E. Hellman. «New directions in cryptography». In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[3] R. L. Rivest, A. Shamir e L. Adleman. «A method for obtaining digital signatures and public-key cryptosystems». In: *Communications of the ACM* 21.2 (1978), pp. 120–126. DOI: 10.1145/359340.359342.

[4] S. Goldwasser, S. Micali e R. L. Rivest. «A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks». In: *SIAM Journal on Computing* 17.2 (1988), pp. 281–308. DOI: 10.1137/0217017.

[5] ACN. *Crittografia post-quantum e quantistica - Preparazione alla minaccia quantistica*. Documenti informativi, 2024. URL: https://www.acn.gov.it/portale/crittografia.

[6] ITU-T. *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*. X.509. 2019. URL: https://www.itu.int/rec/t-rec-x.509.

[7] P. 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.

[8] A. K. Lenstra e H. W. Lenstra. *The Development of the Number Field Sieve*. Springer, 1993. DOI: 10.1007/BFb0091534.

[9] K. Moriarty et al. *PKCS #1: RSA Cryptography Specifications Version 2.2*. RFC 8017. 2016. DOI: 10.17487/RFC8017.

[10] NIST. *Digital Signature Standard (DSS)*. FIPS 186-5. U.S. Department of Commerce, 2023. DOI: 10.6028/NIST.FIPS.186-5.

[11] ISO. *Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms*. ISO/IEC 9796-2. 2010. URL: https://www.iso.org/standard/54788.html.

[12] ACN. *Funzioni di Hash*. Linee Guida Funzioni Crittografiche, 2026. URL: https://www.acn.gov.it/portale/crittografia.

[13] T. ElGamal. «A public key cryptosystem and a signature scheme based on discrete logarithms». In: *IEEE Transactions on Information Theory* 31.4 (1985), pp. 469–472. DOI: 10.1109/TIT.1985.1057074.

[14] S. Vanstone. «Responses to NIST's proposal». In: *Communications of the ACM* 35.7 (1992). Comunicato da John Anderson, pp. 50–52. DOI: 10.1145/129902.129905.

[15] ISO. *IT Security techniques – Digital signatures with appendix*. ISO/IEC 14888-3. 2018. URL: https://www.iso.org/standard/76382.html.

[16] BSI. *Technical Guideline TR-03111 Elliptic Curve Cryptography (ECC)*. 2018. URL: https://www.bsi.bund.de/dok/TR-03111-en.

[17] R. Lidl e H. Niederreiter. *Introduction to Finite Fields and their Applications*. Cambridge University Press, 1994. DOI: 10.1017/CBO9781139172769.

[18] L. Chen et al. *Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters*. SP 800-186. NIST, 2023. DOI: 10.6028/NIST.SP.800-186.

[19] M. Lochter e J. Merkle. *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation*. RFC 5639. 2010. DOI: 10.17487/RFC5639.

[20] A. Langley, M. Hamburg e S. Turner. *Elliptic Curves for Security*. RFC 7748. 2016. DOI: 10.17487/RFC7748.

[21] T. Pornin. *Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)*. RFC 6979. 2013. DOI: 10.17487/RFC6979.

[22] C. P. Schnorr. «Efficient signature generation by smart cards». In: *Journal of Cryptology* 4 (1991), pp. 161–174. DOI: 10.1007/BF00196725.

[23] I. L. S. Josefsson. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032. 2017. DOI: 10.17487/RFC8032.

[24] R. S. Lehman. «Factoring Large Integers». In: *Mathematics of Computation* 28.126 (1974), pp. 637–646. DOI: 10.2307/2005940.

[25] J. M. Pollard. «Theorems on factorization and primality testing». In: *Mathematical Proceedings of the Cambridge Philosophical Society* 76.3 (1974), pp. 521–528. DOI: 10.1017/S0305004100049252.

[26] H. C. Williams. «A $p + 1$ Method of Factoring». In: *Mathematics of Computation* 39.159 (1982), pp. 225–234. DOI: 10.2307/2007633.

[27] Y. Lu, R. Zhang e D. Lin. «Factoring RSA Modulus with Known Bits from Both p and q: A Lattice Method». In: *Network and System Security (NSS)*. Lecture Notes in Computer Science. 2013, pp. 393–404. DOI: 10.1007/978-3-642-38631-2_29.

[28] D. Boneh, G. Durfee e Y. Frankel. «An Attack on RSA Given a Small Fraction of the Private Key Bits». In: *Advances in Cryptology - ASIACRYPT '98*. Lecture Notes in Computer Science. 1998, pp. 25–34. DOI: 10.1007/3-540-49649-1_3.

[29] D. Coppersmith. «Finding a Small Root of a Univariate Modular Equation». In: *Advances in Cryptology - EUROCRYPT '96*. Lecture Notes in Computer Science. 1996, pp. 155–165. DOI: 10.1007/3-540-68339-9_14.

[30] M. 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.

[31] A. K. Lenstra et al. *Ron was wrong, Whit is right*. IACR Cryptology ePrint Archive. 2012. URL: https://eprint.iacr.org/2012/064.

[32] D. Bleichenbacher. «Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1». In: *Advances in Cryptology - CRYPTO '98*. Lecture Notes in Computer Science. 1998, pp. 1–12. DOI: 10.1007/BFb0055716.

[33] Y. Oiwa, K. Kobara e H. Watanabe. «A New Variant for an Attack Against RSA Signature Verification Using Parameter Field». In: *Public Key Infrastructure (EuroPKI)*. Lecture Notes in Computer Science. 2007, pp. 143–153. DOI: 10.1007/978-3-540-73408-6_10.

[34] P. C. Kocher. «Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems». In: *Advances in Cryptology - CRYPTO '96*. Lecture Notes in Computer Science. 1996, pp. 104–113. DOI: 10.1007/3-540-68697-5_9.

[35] D. Brumley e D. Boneh. «Remote timing attacks are practical». In: *Computer Networks* 48.5 (2005), pp. 701–716. DOI: 10.1016/j.comnet.2005.01.010.

[36] S. Bhattacharya e D. Mukhopadhyay. «Who Watches the Watchmen?: Utilizing Performance Monitors for Compromising Keys of RSA on Intel Platforms». In: *Cryptographic Hardware and Embedded Systems (CHES)*. Lecture Notes in Computer Science. 2022, pp. 248–266. DOI: 10.1007/978-3-662-48324-4_13.

[37] A. Pellegrini, V. Bertacco e T. Austin. «Fault-based attack of RSA authentication». In: *Design, Automation & Test in Europe (DATE)*. 2010, pp. 855–860. DOI: 10.1109/DATE.2010.5456933.

[38] A. J. Menezes, P. C. V. Oorschot e S. A. Vanstone. *Handbook of applied cryptography (1st edition)*. CRC Press, 1997. DOI: 10.1201/9780429466335.

[39] S. Pohlig e M. Hellman. «An improved algorithm for computing logarithms over GF(p) and its cryptographic significance». In: *IEEE Transactions on Information Theory* 24.1 (1978), pp. 106–110. DOI: 10.1109/TIT.1978.1055817.

[40] A. E. Western e J. C. P. Miller. *Indices and primitive roots*. Vol. 9. Royal Society Mathematical Tables. Cambridge University Press, 1968.

[41] D. Coppersmith. «Fast evaluation of logarithms in fields of characteristic two». In: *IEEE Transactions on Information Theory* 30.4 (1984), pp. 587–594. DOI: 10.1109/TIT.1984.1056941.

[42] L. M. Adleman e J. Demarrais. «A Subexponential Algorithm for Discrete Logarithms Over all Finite Fields». In: *Mathematics of Computation* 61.203 (1993), pp. 1–15. DOI: 10.2307/2152932.

[43] NIST. *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[44] NIST. *PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates*. 2022. URL: https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4.

[45] D. Cooper et al. *Recommendation for Stateful Hash-Based Signature Schemes*. SP 800-208. NIST, 2020. DOI: 10.6028/NIST.SP.800-208.

[46] L. K. Grover. «A fast quantum mechanical algorithm for database search». In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. 1996, pp. 212–219. DOI: 10.1145/237814.237866.

[47] ISO. *Information security – Digital signatures with appendix – Part 4: Stateful hash-based mechanisms*. ISO/IEC 14888-4. 2024. URL: https://www.iso.org/standard/80492.html.

[48] D. McGrew, M. Curcio e S. Fluhrer. *Leighton-Micali Hash-Based Signatures*. RFC 8554. 2019. DOI: 10.17487/RFC8554.

[49] A. Huelsing et al. *XMSS: eXtended Merkle Signature Scheme*. RFC 8391. 2018. DOI: 10.17487/RFC8391.

[50] R. C. Merkle. «A Digital Signature Based on a Conventional Encryption Function». In: *Advances in Cryptology - CRYPTO '87*. Lecture Notes in Computer Science. 1987, pp. 369–378. DOI: 10.1007/3-540-48184-2_32.

[51] J.-P. Aumasson et al. *SPHINCS$^+$ - Submission to the NIST post-quantum project*. NIST submission. 2017. URL: https://sphincs.org/data/sphincs+-specification.pdf.

[52] NIST. *Stateless Hash-Based Digital Signature Standard*. FIPS 205. U.S. Department of Commerce, 2023. DOI: 10.6028/NIST.FIPS.205.

[53] P.-A. Fouque et al. *Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU*. NIST submission. 2020. URL: https://falcon-sign.info/falcon.pdf.

[54] L. Ducas et al. *CRYSTALS-Dilithium - Algorithm Specifications and Supporting Documentation*. NIST submission. 2017. URL: https://pq-crystals.org/dilithium/data/dilithium-specification.pdf.

[55] NIST. *Module-Lattice-Based Digital Signature Standard*. FIPS 204. U.S. Department of Commerce, 2023. DOI: 10.6028/NIST.FIPS.204.

[56] O. Regev. «On lattices, learning with errors, random linear codes, and cryptography». In: *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*. STOC '05. 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.

[57] M. Ajtai. «Generating hard instances of lattice problems (extended abstract)». In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC)*. 1996, pp. 99–108. DOI: 10.1145/237814.237838.

[58] A. Fiat e A. Shamir. «How To Prove Yourself: Practical Solutions to Identification and Signature Problems». In: *Advances in Cryptology - CRYPTO '86*. Lecture Notes in Computer Science. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[59] W. Beullens. «Breaking Rainbow Takes a Weekend on a Laptop». In: *Advances in Cryptology - CRYPTO 2022*. Lecture Notes in Computer Science. 2022, pp. 464–479. DOI: 10.1007/978-3-031-15979-4_16.

[60] N. Bindel et al. *Hybrid signature spectrums*. RFC draft. 2026. URL: https://datatracker.ietf.org/doc/draft-ietf-pquip-hybrid-signature-spectrums/.