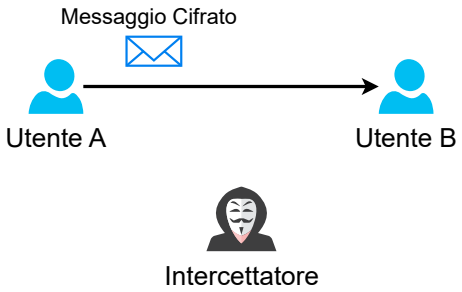
0.1 Introduzione

In questo corso andremo a vedere le basi della **Crittografia moderna**. In primis, dobbiamo capire cosa vuol dire "*Crittografia*". Partiamo dall'etimologia: dal greco ***kryptós*** (nascosto) – ***graphía*** (scrittura). In sostanza, la Crittografia è la disciplina che studia e analizza come inviare e ricevere messaggi **nascosti**. Con il termine "nascosti" si intende che solo ed esclusivamente la sorgente e il destinatario possono leggere il contenuto del messaggio, mentre qualsiasi altra persona non può.



In questa immagine l'utente A manda un messaggio criptato all'utente B; in questo modo solo A e B potranno leggere il contenuto del messaggio, mentre l'intercettatore, anche se riuscisse ad averne una copia, non sarebbe in grado di leggerlo (dato che è criptato).

La crittografia la usiamo tutti i giorni (anche involontariamente) con i nostri dispositivi elettronici. Un esempio è **WhatsApp**, che tramite una crittografia **End-to-End** (che avremo tempo di approfondire) permette di inviare messaggi in maniera sicura, in modo che nessun altro (nemmeno WhatsApp stesso!) possa leggere il messaggio che hai mandato al tuo amico. Ha anche utilità nell'autenticazione digitale e nei documenti elettronici: infatti, tutti i sistemi come **SPID** oppure **CIE** sfruttano la crittografia per funzionare. La crittografia viene utilizzata anche dalle case produttrici di console (come **Sony** per la **PlayStation**) per impedire di crackare le loro console. Di esempi ce ne sono a centinaia e avremo tempo per scoprirli tutti.

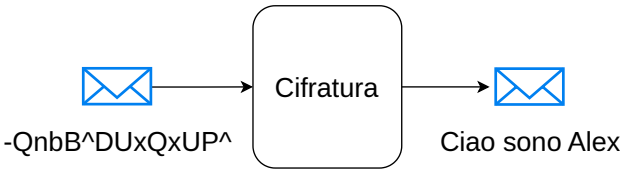
Anche se ho elencato molti esempi **digitali**, la crittografia è una disciplina basata sulla **matematica**. Infatti, tutti i sistemi crittografici si fondano su prove matematiche (come il **Logaritmo Discreto** e la **Fattorizzazione di numeri composti**) per funzionare. Mi piace definire la crittografia come una branca a metà strada tra matematica e informatica, poiché utilizza concetti matematici applicati a contesti informatici.

Un altro punto fondamentale da chiarire è che, anche se parleremo di sistemi moderni come **AES**, **RSA**, **Diffie-Hellman** e **ECC**, inventati tra il 1960 e il 1990 circa, in realtà la crittografia è molto più antica. Già dall'**Impero Romano** (753 a.C. – 476 d.C.) se ne parlava: chiaramente i sistemi erano molto più semplici di quelli odierni, ma all'epoca servivano per inviare messaggi all'esercito. In questo coro di cifrari "antichi" ne vedremo due, forse i più impattanti nella storia: il **Cifrario di Cesare**, che possiamo definire il primo sistema crittografico, e la macchina **Enigma**, che durante la Seconda Guerra Mondiale fu di fondamentale importanza per le truppe dell'Asse; gli alleati, grazie a **Alan Turing**, riuscirono a decifrarla, aiutando così la loro vittoria.

Fatte tutte le premesse del caso, iniziamo a parlare di crittografia; come prima cosa, vediamo e definiamo tutti i termini utilizzati in questo ambito.

Cifratura

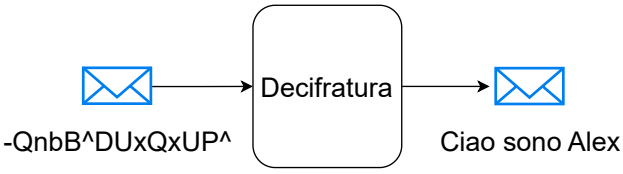
La cifratura di un messaggio è il processo che permette di **alterare** un messaggio in modo che nessun altro (eccetto chi lo invia e il destinatario) possa leggere il contenuto originario. La cifratura deve avvenire tramite un **algoritmo di cifratura** e con l'ausilio di una (o più) **chiavi**.



In questo caso, il messaggio "**Ciao sono Alex**" tramite una cifratura è diventato **-QnbBˆDUxQxUPˆ**; se qualcuno riuscisse a intercettare il messaggio cifrato, non capirebbe nulla.

Decifratura

La decifratura è il passaggio **inverso** della cifratura, nel senso che permette di tornare al messaggio originale partendo dal messaggio cifrato. Chiaramente, bisogna usare lo **stesso algoritmo di cifratura** e, soprattutto, la **stessa chiave**, che deve essere conosciuta solo da chi invia il messaggio e da chi lo deve ricevere.



Iniziamo a utilizzare un po' di nozionismo matematico: il meccanismo di cifratura e decifratura può essere paragonato a una **funzione**, poiché entrambe prendono una variabile in input e restituiscono un valore in output. Possiamo quindi definire la cifratura come

$$c = f(m)$$

Dove c è il messaggio cifrato, m è il messaggio originale e f è la "funzione di cifratura". Dato ciò, la "funzione di decifratura" sarà definita come

$$m = f^{-1}(c)$$

Questo può essere dedotto dalla seguente equazione:

$$m = f^{-1}(f(m))$$

Tranquilli, per ora abbiamo terminato con il nozionismo matematico.

Chiave

Una chiave è una qualsiasi **stringa** o, più semplicemente, un **numero**; la caratteristica principale è che una chiave deve **rimanere privata**, perché permette di cifrare e decifrare i messaggi. Se qualcuno riuscisse a rintracciare la vostra chiave privata, potrebbe leggere tutti i messaggi che inviate e ricevete. L'idea della chiave in crittografia è simile a quella di una **password**: allo stesso modo, se qualcuno vi ruba la password, può accedere al vostro account. In realtà, la chiave può anche essere qualcosa di più complesso, come un **punto nel piano cartesiano** (usato nella *Elliptic Curve Cryptography*).

Quindi, per evitare confusione, correggiamo la definizione precedente dicendo che una chiave è un qualsiasi **dato**, oppure un **insieme di dati**, che deve rimanere **segreto**.

In realtà, vedremo verso metà corso che esiste anche una cosiddetta **chiave pubblica**, ovvero una chiave come quella che abbiamo definito fino a ora, ma che **chiunque può conoscere**. Se vi sembra strano o controintuitivo, quando lo vedremo tutto sarà chiaro.

Rotto

Un sistema crittografico si definisce **rotto** qualora sia possibile decifrare un messaggio criptato senza conoscere la chiave. Un sistema rotto, chiaramente, non può essere utilizzato, perché chiunque riuscirebbe a decifrare il messaggio. Un esempio di sistema rotto è il **DES** (che vedremo nel capitolo "Cifrari a Blocchi"). Il DES è stato inventato nel 1976 e, all'inizio, era molto usato; il problema era che utilizzava una chiave a lunghezza fissa: 54 bit. Ad oggi, purtroppo, una chiave a 54 bit è soggetta ad attacchi **brute-force**[1], e per questo oggi il DES non può più essere usato per cifrare ed è stato sostituito dall'**AES**.

0.1.1 Principio di Kerckhoffs

Ora che abbiamo iniziato a familiarizzare con i primi termini della crittografia, possiamo comprendere il principio fondante della disciplina: il **Principio di Kerckhoffs**.

Teorema 1: Principio di Kerckhoffs

La sicurezza di un sistema crittografico deve dipendere unicamente dalla chiave segreta e non dalla segretezza dell'algoritmo stesso.

Sostanzialmente, Kerckhoffs afferma che non deve essere segreto **l'algoritmo di cifratura**; la forza di un sistema crittografico deriva invece dalla difficoltà di romperlo, e non dalla segretezza dell'algoritmo.

Per questo motivo, oggi sappiamo perfettamente quali algoritmi utilizzano i vari siti e le app; non è un rischio conoscere il metodo con cui viene criptato un messaggio, perché la sicurezza risiede nella segretezza della chiave, che, chiaramente, deve rimanere privata.

Per esempio, secondo Kerckhoffs, se tu e un tuo amico volete creare un sistema per scambiarvi messaggi segreti, non potete semplicemente utilizzare un sistema debole mantenendolo segreto a tutti gli altri; se qualcuno riuscisse a scoprire l'algoritmo, potrebbe leggere tutti i vostri messaggi. i messaggi.

[1] Attacchi in cui si provano tutte le possibili combinazioni di una chiave; richiedono molto tempo, ma per chiavi molto piccole (come il DES) possono funzionare

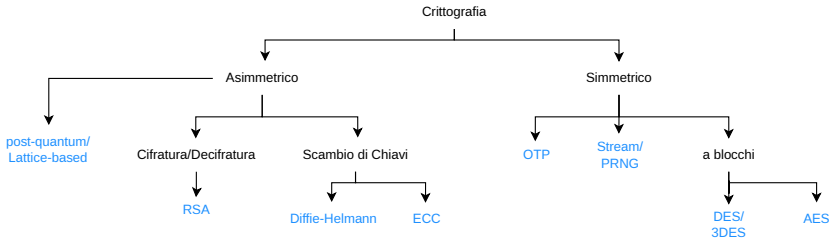
0.1.2 Il problema dello scambio della Chiave

Prima di cominciare a parlare della classificazione dei sistemi crittografici, è necessario affrontare il problema dello scambio della chiave. Ripetendo quanto visto finora, la crittografia studia come due utenti possano scambiarsi messaggi in modo che nessun altro ne possa leggere il contenuto. Abbiamo capito che il messaggio viene cifrato e poi decifrato tramite un algoritmo. Inoltre, un algoritmo ha bisogno di una chiave per cifrare i messaggi, e la chiave deve essere posseduta solo da chi invia il messaggio e da chi lo deve ricevere; nessun altro dovrebbe conoscerla, altrimenti anche altri utenti potrebbero decifrare i messaggi. Tuttavia, non abbiamo ancora considerato come i due utenti possano scambiarsi una chiave comune o, comunque, concordare su una chiave da usare nel sistema.

In questa immagine, l'utente A ha generato una chiave da usare per cifrare i messaggi, ma deve trovare un modo per inviarla a B (così che lui possa decifrare i messaggi di A) senza che l'intercettatore riesca a intercettarla. Questo problema è stato risolto tramite i sistemi **asimmetrici**; come ciò sia possibile lo vedremo quando li studieremo nel dettaglio. Per ora, vi basta sapere che lo scambio della chiave è risolto da questi tipi di sistemi.

0.1.3 Le prime Classificazioni

I sistemi crittografici si dividono in diverse sottocategorie, ognuna con le proprie caratteristiche. Per comprenderle meglio, vediamo subito una mappa riassuntiva di tutte le categorie e poi le commenteremo una ad una. Perciò, ecco a voi la mappa:



La prima grande distinzione nella crittografia moderna è quella tra sistemi **simmetrici** e **asimmetrici**. Un sistema simmetrico utilizza **una sola chiave**, che deve rimanere sempre privata, mentre i sistemi asimmetrici hanno **due chiavi: una privata e una pubblica**. I due sistemi sono complementari l'uno all'altro e, ora, vedremo i principali vantaggi e svantaggi di entrambi.

Sistemi Simmetrici:

- I sistemi simmetrici sono veloci dal punto di vista computazionale, nel senso che i computer possono eseguire rapidamente gli algoritmi di cifratura e decifratura.

  Questi algoritmi utilizzano combinazioni di operazioni booleane (come lo **XOR**) e operazioni su matrici, calcoli che i computer sanno eseguire in maniera eccellente. In alcuni casi, è anche possibile parallelizzare alcune fasi per velocizzare ulteriormente il processo.

- Tuttavia, i sistemi simmetrici non risolvono il problema dello scambio della chiave.

Sistemi Asimmetrici:

- I sistemi asimmetrici risolvono il problema dello scambio della chiave, nel senso che non è necessario che i due utenti abbiano precedentemente condiviso una chiave segreta.

- I sistemi asimmetrici sono più lenti dal punto di vista computazionale, perché devono eseguire operazioni su numeri molto grandi (parliamo di numeri con fino a 600 cifre!).

Queste sono, intanto, le prime differenze tra sistemi asimmetrici e simmetrici, e si nota che sono complementari: difficilmente, infatti, nei progetti si utilizza solo uno dei due, perché è preferibile impiegare entrambi. Per esempio, il protocollo **HTTPS**, che serve per inviare le pagine web in modo cifrato, crea una chiave per un sistema **simmetrico**, ma la chiave viene cifrata tramite un sistema **asimmetrico**. In questo modo, i due utenti ottengono la chiave in **maniera sicura** (perché è stata inviata tramite un sistema asimmetrico), mentre la comunicazione vera e propria utilizza un sistema simmetrico, poiché è più **veloce**.

Sistemi Simmetrici - OTP

Tra le due categorie, i sistemi simmetrici sono i primi che vedremo, poiché sono tendenzialmente più semplici. Come si nota dal grafico, i simmetrici si suddividono in altre tre categorie: **OTP**, **Stream** e **a Blocchi**. Il primo che analizzeremo è **OTP** (*One Time Pad*), l'unico sistema definito **perfettamente sicuro**, ovvero tale per cui, partendo dal messaggio cifrato senza la chiave, è impossibile risalire al messaggio originale. Tutti gli altri sistemi, se sottoposti ad attacchi **brute-force**, permettono di risalire al messaggio originale senza la chiave di cifratura. Chiaramente, tali attacchi sono praticamente infattibili perché richiederebbero anni per essere completati; tuttavia, nell'ipotesi di disporre di un computer infinitamente potente, il cifrario OTP sarebbe l'unico **impossibile** da decifrare senza la chiave.

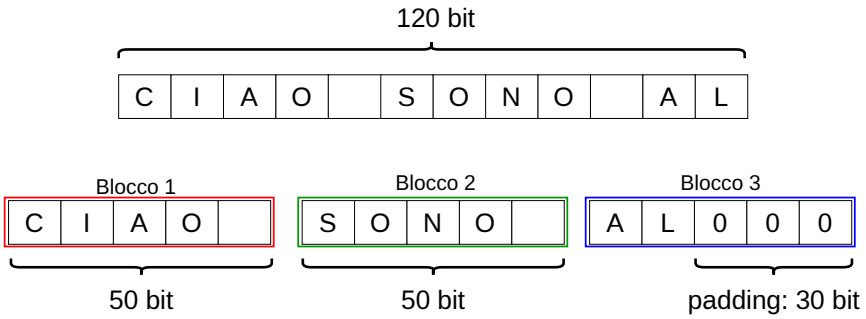
Dopo questa definizione, potreste pensare che si potrebbe usare sempre e solo l'**OTP**, ma purtroppo ha un limite che lo rende poco praticabile: è necessario cambiare la chiave dopo ogni cifratura. Infatti, se due messaggi vengono cifrati con la stessa chiave, tramite delle **criptoanalisi** è possibile risalire sia alla chiave sia ai messaggi originali. Se qualcuno pensasse di generare sempre nuove chiavi, la gestione diventerebbe eccessivamente complessa e insostenibile per la comunicazione.

Sistemi Simmetrici - PRNG / Stream

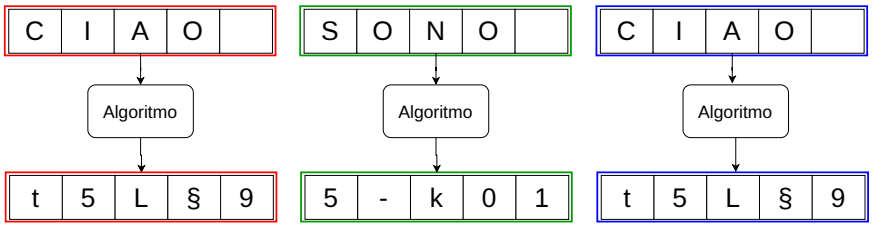
La categoria successiva è quella degli **Stream**, che però vedremo solo brevemente, poiché non sono molto utilizzati. Gli Stream funzionano generando un **flusso** (da cui il nome *Stream*) di **numeri casuali**, che vengono poi utilizzati per cifrare i messaggi tramite l'**OTP**. Il problema è che la generazione di numeri casuali per i computer è **impossibile**, poiché i computer sono sistemi **deterministici**. Per produrre numeri per questo sistema, si utilizzano algoritmi detti **PRNG** (*Pseudo Random Number Generator*), che cercano di generare numeri apparentemente casuali ma che presentano comunque correlazioni tra loro; con determinati attacchi, è possibile prevedere i numeri successivi dello stream anche senza conoscere la chiave. Per questo motivo, questa tipologia di cifrari oggi non viene quasi mai utilizzata.

Sistemi Simmetrici - A Blocchi

Finalmente arriviamo ai veri sistemi simmetrici: quelli a blocchi. Questi cifrari sono quelli utilizzati oggigiorno per la loro sicurezza e **velocità**. I cifrari a blocchi funzionano suddividendo il messaggio in blocchi di n bit. Supponiamo di voler cifrare un messaggio di 120 bit e di avere un cifrario a blocchi che opera su blocchi da 50 bit. In questo caso, il nostro messaggio **sarà diviso in 3 blocchi:** il primo da 50 bit, il secondo da 50 bit e l'ultimo da 20 bit. Se l'ultimo blocco non raggiunge la dimensione prevista dal cifrario, vengono aggiunti degli zeri (o un qualsiasi **padding**) in modo che raggiunga la lunghezza di 50 bit.



Questo meccanismo di creare blocchi è necessario perché ogni blocco viene poi trasformato in una **matrice**. Per questo motivo, è importante che ogni blocco abbia una dimensione definita, poiché gli algoritmi sfruttano operazioni su matrici per cifrare il messaggio. Allo stesso modo, il messaggio cifrato sarà anch'esso una matrice della stessa dimensione, che verrà poi riconvertita in messaggio. È importante comprendere che, poiché l'algoritmo cifra un blocco alla volta, lo stesso blocco in input produce sempre lo stesso blocco cifrato. Ciò significa che, se due blocchi all'interno del messaggio originale sono identici, avranno lo stesso blocco cifrato.



I blocchi cifrati vengono poi riuniti per formare il messaggio cifrato. Per evitare che due blocchi identici producano lo stesso output (cosa che potrebbe aiutare a risalire al messaggio originale), sono stati ideati i **modi di funzionamento dei cifrari a blocchi**, che vedremo con calma e in dettaglio più avanti.

La forza di questi cifrari è che utilizzano soltanto operazioni **booleane** (**and**, **or**, **xor** e **not**) e operazioni **su matrici**, il che li rende molto veloci, poiché queste operazioni sono altamente ottimizzate nei computer odierni.

I principali algoritmi a blocchi sono **DES** (*Data Encryption Standard*) e **AES** (*Advanced Encryption Standard*). Il DES nacque nel 1976 e, per i vent'anni successivi, fu lo standard per i cifrari a blocchi. Nel 1999, però, dei ricercatori riuscirono a **rompere** il cifrario, rendendolo insicuro a causa della breve lunghezza della chiave, che permetteva un attacco brute-force. Al suo posto arrivò **AES** nel 1998, che ad oggi è ancora considerato un sistema sicuro. Inoltre, fu creato il **3DES** (*Triple DES*), che consiste nel cifrare un messaggio tre volte con il DES; questo stratagemma permette ancora oggi di usare il DES in forma sicura, anche se oggi si utilizza quasi esclusivamente il 3DES e non più il DES singolo.

Sistemi Asimmetrici - RSA

Passando all'altro lato del nostro schema arriviamo ai sistemi **asimmetrici**. Come abbiamo gia detto, questi sistemi hanno come caratteristica principale che non hanno solamente una chiave ma ne hanno **una privata e una pubblica**. RSA (che è l'acronimo dei 3 crittografi che lo hanno inventato) è un algoritmo che permette di creare le due chiavi in maniera sicura, e permette anche ci **cifrare e decifrare** (cosa che negli algoritmi a **Scambi di chiave** non è contemplata). RSA sfrtutta la difficoltà di **Fattorizzare numeri grandi** (che come dicevamo stiamo parlando di numeri a 600 cifre circa). Infatti ad oggi, l'unico modo che abbiamo per fattorizzare un numero è provare a dividere tutti i numeri minori di esso. Quindi per i computer odierni non è sostenibile un calcolo così impegnativo.

Algorithm 1: Fattorizzazione di un numero n

```
Input: n
Output: list
list ← [];
for k ← 2 to n − 1 do
    if n mod k = 0 then
        list.append(k);
return list;
```

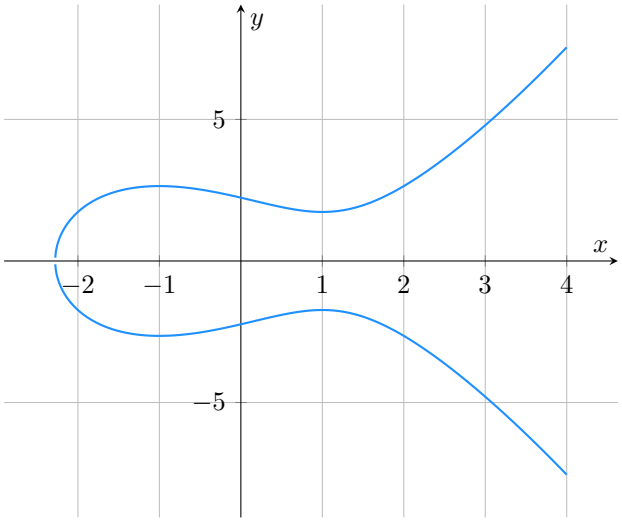
Sistemi Asimmetrici - Scambio di Chiavi

Tra gli algoritmi asimmetrici abbiamo anche quelli che io ho chiamato **Scambio di Chiave**, i quali si occupano esclusivamente di **generare una chiave condivisa tra due utenti**, che poi sarà utilizzata con un algoritmo simmetrico come l'**AES** o il **DES**. Questo approccio consente di combinare i vantaggi dei sistemi simmetrici (**velocità**) con quelli dei sistemi asimmetrici (**sicurezza**).

Tra questi algoritmi, i due principali sono: **Diffie-Hellman** (dal nome dei ricercatori che lo hanno inventato) e **ECC** (*Elliptic Curve Cryptography*). Diffie-Hellman, come vedremo, è molto semplice e sfrutta un meccanismo di potenze e moduli, mentre ECC è più particolare: utilizza infatti una curva ellittica definita da un'equazione del tipo seguente:

$$y^2 = x^3 + ax + b$$

che ha come grafico:



L'idea alla base di ECC è quella di **modulare la curva secondo un numero primo** p e di scegliere un punto casuale su di essa, sul quale eseguire una particolare operazione, la **somma tra punti**, che vedremo con calma, poiché risulta particolarmente complessa.

Ad ogni modo, sia Diffie-Hellman che ECC sfruttano la difficoltà di calcolare il **logaritmo discreto**. In altre parole, per noi e per i computer è semplice risolvere la seguente equazione:

$$2^x = 9$$

Infatti, per risolverlo basta calcolare il logaritmo.

$$x = log_2 9$$

Il discorso cambia però quando applichiamo questo ragionamento ai moduli (che vedremo meglio in seguito; se non li avete mai visti, potete saltare al prossimo paragrafo). Infatti, se consideriamo un' equazione del genere:

$$2^x \equiv 9 \mod 17$$

È complicatissimo risolverla, perché non è più possibile applicare il logaritmo a causa del modulo. Come per la fattorizzazione, ad oggi non esistono algoritmi che permettano di risolvere queste equazioni in tempi ragionevoli.

Post-Quantum

Ultimo appunto prima di partire con i cifrari antichi: voglio fare una piccola introduzione sui **Cifrari Post-Quantum**, perché gli algoritmi **RSA**, **Diffie-Hellman** e **ECC** sono attualmente molto difficili da rompere con i computer moderni, ma è possibile che, con i computer **quantistici**, possano essere violati anche in pochi secondi. Infatti, i computer quantistici sono particolarmente adatti a risolvere problemi su cui si basano questi algoritmi: ad esempio, possono fattorizzare numeri enormi o calcolare il logaritmo discreto in tempi brevissimi.

Tuttavia, ad oggi i computer quantistici sono ancora in fase di sviluppo e c'è ancora molta strada da fare: per esempio, il numero più grande mai fattorizzato con un computer quantistico è **21**. Non si può sapere come saranno i computer quantistici in futuro; per questo motivo, i maggiori ricercatori stanno lavorando per creare nuovi cifrari **difficili da rompere anche per computer quantistici**.

Tra questi, quelli che stanno ricevendo maggiore attenzione dalla comunità scientifica sono i **Lattice-Based**. Si tratta di algoritmi che sfruttano i **reticoli** (in inglese *Lattice*), ovvero "piani cartesiani" a n dimensioni, ad esempio $\mathbb{Z}^n$.

Si possono generare dei vettori linearmente indipendenti a n dimensioni. A partire da questi, è possibile definire un punto e determinare il percorso più semplice per raggiungerlo usando i nuovi vettori. Ad ogni modo, questi cifrari sono particolarmente complessi, ma rappresentano anche il futuro della crittografia. Per questo motivo, in questo corso non li tratteremo, ma consiglio a tutti di studiarli autonomamente.