Introduzione alla Post-Quantum Cryptography

Matteo Bonini

Università degli Studi di Trento, CryptoLabTN

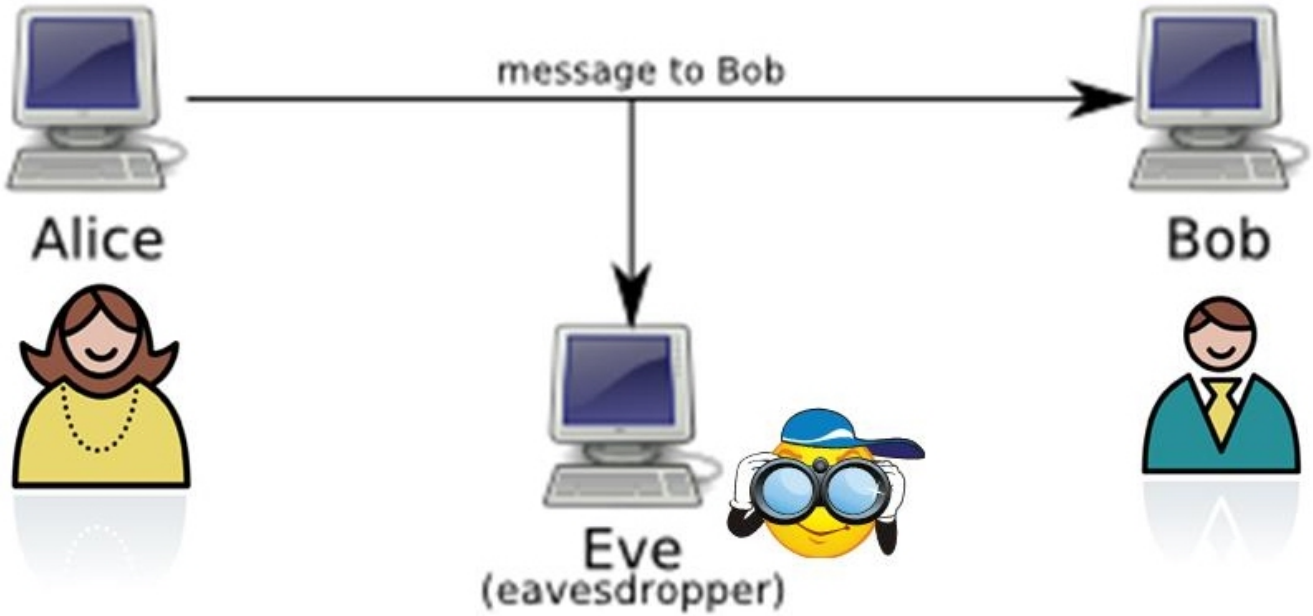
5 Aprile 2017

Crittografia

Crittografia = scienza/arte delle scritture segrete

Obiettivo della crittografia

Consentire a due utenti (Alice e Bob) di comunicare su un canale potenzialmente insicuro, senza permettere a una terza persona di comprendere il contenuto dei messaggi scambiati.



Crittografia a chiave pubblica

- Ci sono due chiavi distinte per ogni parte in gioco, una resa pubblica e una tenuta privata:
  1. Una chiave pubblica che viene utilizzata per le operazioni di cifratura.
  2. Una chiave privata che viene utilizzata per le operazioni di decifratura.
- Questo tipo di cifrari sono considerati **forti** perchè utilizzano funzioni **one way**, ovvero funzioni facili da calcolare ma praticamente impossibili da invertire senza la conoscenza della chiave segreta.
- Le funzioni one way sono costruite basandosi su problemi matematici che riteniamo **difficili** da risolvere, ovvero per i quali non si conosce un algoritmo di soluzione rapido.

Crittosistemi a chiave pubblica

I crittosistemi a chiave pubblica vengono utilizzati principalmente per svolgere due compiti:

1. Scambio di chiavi sicure per essere utilizzate con cifrari simmetrici.
2. Firma digitale.

Crittosistemi a chiave pubblica famosi:

- **RSA** (1977), che è basato sul problema della fattorizzazione dei numeri interi.
- **Crittografia a curve ellittiche**, che basa la sua sicurezza sulla difficoltà di eseguire certe operazioni sui punti di questa tipologia di curve.
- **Diffie-Hellman**, basato sulla difficoltà del logaritmo discreto su alcuni gruppi ciclici.

Calcolatori

- Computer deterministico: computer funzionante in base alle leggi dell'elettromagnetismo (gli attuali computer). Funziona utilizzando i **bits** che hanno 2 possibili stati (acceso/spento).
- Computer quantistico: computer funzionante in base alle leggi della meccanica quantistica. Funziona utlizzando i **qubits** che hanno 4 possibili stati.

Sicurezza degli attuali crittosistemi a chiave pubblica

La sicurezza degli attuali crittosistemi a chiave pubblica è basata su problemi che consideriamo *molto difficili*, ovvero per i quali non si conosce un algoritmo polinomiale per la soluzione:

1. Il problema della fattorizzazione.
2. Il problema del logaritmo discreto.
3. Il problema del logaritmo discreto dei punti di una curva ellittica.

Algoritmo di Shor

Teorema (Shor, 1997)

Utilizzando un quantum computer è possibile fattorizzare i numeri interi in tempo polinomiale.

P.W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", SIAM J. Comput., pp. 1484 – 1509, 1997.
In realtà tutti i problemi scelti per la crittografia a chiave pubblica sono **deboli** per un computer quantistico (utilizzando versioni modificate dell'algoritmo di Shor).

Protezione dalla Quantum Computation

Come possiamo difenderci?

- **Quantum cryptography**: crittosistemi che si basano sulle stesse proprietà fisiche alla base della quantum computation.
- **Post-quantum cryptography**: crittosistemi a chiave pubblica in grado di resistere agli attacchi dei quantum computers, ma utilizzabili anche su calcolatori deterministici.

Problematiche

Importante

L'unica parte della crittografia che è messa in serio pericolo dall'avvento dei computer quantistici è la crittografia a chiave pubblica.
La crittografia simmetrica e le funzioni di hash si basano su problemi che sono resistententi agli algoritmi quantistici.

- La crittografia simmetrica basa la propria robustezza sulla difficoltà di risolvere sistemi polinomiali.
- Le funzioni di hash ripongono la loro robustezza sulla difficoltà di essere invertite e sulla difficoltà di trovare collisioni.

Questi due problemi sono **molto difficili** e per questo con un quantum computer la loro sicurezza non è compromessa.

Crittosistemi Post-Quantistici

- Crittografia con codici correttori di errori:
  - **McEliece-Niederreiter**
- Crittografia su reticoli
  - **NTRU**
  - **New Hope**
  - **Ring-LWE Signature**
- Crittografia con i polinomi
  - **Hidden Field Equations**
  - **Unbalanced Oil and Vinegar Cryptosystems**
- Hash-based cryptography
  - **Merkle signature scheme**
- Crittografia su isogenie di curve ellittiche supersingolari

Code-Based Cryptography

R.J. McEliece, " A public key cryptosystem based on algebraic coding theory", Technical report, Jet Propulsion Lab DSN Progress Report, 1978.

H. Niederreiter, "Knapsack-type cryptosystem and algebraic coding theory", Problems of Control and Information Theory , 15(2): 159 – 166, 1986.

Codici correttori d'errori

Codice Lineare

Un **codice lineare** $[n, k]_q$ C su $\mathbb{F}_q$ è un sottospazio di $\mathbb{F}_q^n$ di dimensione k, $0 \leq k \leq n$. n è la **lunghezza** del codice, k la **dimensione**.

Se C è un $[n, k]_q$ codice, allora ogni matrice G di dimensione $k \times n$, le cui righe sono una base per C come k-spazio vettoriale, è detta **matrice generatrice** di C.

$$v \in \mathbb{F}_q^k \Longrightarrow v \cdot G \in C$$

Codici correttori d'errori

Distanza di Hamming

Presi $u, v \in \mathbb{F}_q^n$ definiamo la loro **distanza di Hamming** come:

$$d(u, v) := \#\{\, i \,|u_i \neq v_i, 1 \leq i \leq n\}$$

Preso un codice C diciamo che la sua **distanza minima** è:

$$d := \min_{u,v \in C, u \neq v} d(u, v)$$

C viene detto un $[n, k, d]_q$-codice.

Teorema

Sia C un $[n, k, d]_q$-codice allora:

- C rileva fino a $d - 1$ errori.
- C corregge fino a $t := \lfloor \frac{d-1}{2} \rfloor$ errori.

Decodifica per massima verosimiglianza

La decodifica per massima verosimiglianza consiste nei passaggi seguenti

1. Si confronta la parola ricevuta con *tutte* le parole del codice.
2. Si prende come risultato la parola con distanza minima rispetto a quella ricevuta.

Importante

La decodifica per massima verosimiglianza ha costo **esponenziale**.

Decodifica con sindromi

Il codice C può essere definito o da una sua matrice generatrice G o tramite una matrice $(n-k) \times n$, detta **matrice di parità**, che chiameremo H:

$$\forall x \in \mathbb{F}_q^n \quad Hx^t = 0 \iff x \in C$$

Sia C un $[n, k, d]_q$-codice e siano $c, e, y \in \mathbb{F}_q^n$ rispettivamente la parola trasmessa, l'errore e il vettore ricevuto, allora $c + e = y$: Dato y, applichiamo ad esso la matrice di parità H:

$$Hy^t = H(x+e)^t = Hx^t + He^t = He^t$$

Il vettore $s = Hy^t \in \mathbb{F}_q^{n-k}$ è detto **sindrome**

Teorema

Se il numero di errori è correggibile, allora esiste un unico vettore di errore corrispondente alla sindrome $s = He^t$.

Decodifica con sindromi

Quindi per decodificare con utilizzando le sindromi si deve creare una tabella in questo modo:

1. Si prendono tutti i vettori di errore singolo, $w(e) = 1$, e si calcolano tutti le sindromi corrispondenti.
2. Si fa lo stesso per tutti i vettori di errore doppio.
3. Si continua fino a raggiungere i vettori di errori tali che $w(e) = t$, con t capacità correttiva del codice.

Per decodificare si guarda alla tabella creata e si recupera l'errore associato alla sindrome che si è ottenuta, poi si sottrae l'errore alla parola per ottenere il messaggio originale.

Importante

Il costo computazionale della decodifica con sindromi è **esponenziale**, anche se generalmente è minore della decodifica per confronti.

Difficoltà della decodifica di un codice qualsiasi

Teorema

Il problema della decodifica di un **generico** codice lineare è un problema **NP-hard** (ovvero riconducibile polinomialmente a un problema NP-completo).

Questo non significa che preso un qualsiasi codice lineare non esista per quello specifico codice un algoritmo polinomiale per la decodifica. Esistono classi di codici per cui si conoscono algoritmi per la decodifica polinomiali (e.g. Reed-Solomon, BCH, Goppa).

Code-Based Cryptography

- Chiave Privata: $K_{priv} := (S, G, P)$ dove:
  1. S è una matrice $k \times k$ invertibile su $\mathbb{F}_q$
  2. G è una matrice generatrice di un $[n, k, d]_q$-codice C dotato di un algoritmo di decodifica rapida
  3. P è una matrice di permutazione $n \times n$.
- Chiave Pubblica: $K_{pub} := SGP$

Nel crittosistema di McEliece (1978) C è un codice di **Goppa** binario. Originariamente Neiderreiter aveva proposto di usare un'altra classe di codici lineari, i codici GRS su $\mathbb{F}_2^m$, ma questa variante si è dimostrata debole.

Codifica

Supponiamo che Alice voglia inviare a Bob il messaggio x. Alice codificherà x in x_0 nel seguente modo:

$$x \longrightarrow x_0 = xSGP + e$$

dove e è un vettore **random** in $\mathbb{F}_q^n$ di peso t, la capacità correttiva del codice scelto.

Importante

e deve essere scelto random, altrimenti è possibile attaccare in maniera molto semplice il crittosistema!

Decodifica

Bob per leggere ciò che Alice gli ha inviato decodificherà x_0 in questa maniera:

$$x_0 \longrightarrow x_0 P^{-1} = xSG + eP^{-1}$$

Importante

$w(eP^{-1}) = w(e) = t$ visto che P è una matrice di permutazione.

Bob quindi può applicare l'algoritmo di decodifica rapida che C possiede Bob per ottenere xS, dal quale poi ricaverà x semplicemente moltiplicandolo per S^{-1}, dato che S è una matrice invertibile.

Sicurezza del crittosistema di McEliece

Vantaggi

- Dopo più di 30 anni il crittosistema di McEliece (nella sua forma originale) rimane sostanzialmente inviolato. La sua sicurezza si basa su due fattori:
  1. problema della decodifica di codici lineari random (NP-hard).
  2. difficoltà nel recuperare la chiave privata, o almeno una equivalente.
- Inoltre il crittosistema di McEliece ha anche una velocità di codifica-decodifica superiore ad RSA.

Svantaggi

- Chiave pubblica molto grande in rapporto al grado di sicurezza fornito e basso rate d'informazione trasmesso.
- Possibilità di lanciare di attacchi in parallelo.

Paragoni di sicurezza

Per una sicurezza a 128-bit si deve prendere un $[2'960, 2'288, 113]_2$-codice di Goppa e la dimensione della chiave pubblica è di **$1'534'896$** bits.

Year	Sym-metric Key Size	Lower bound for $\log_2 S(n,k,t)$	McEliece para-meters (n,k,t) and public key size (kB)	RSA Key Size and SDL[a] Field Size	Elliptic Curve Key Size (bits) for $c=0$	Elliptic Curve Key Size (bits) for $c=18$	Infeasible number of MIPS-years	Corresponding number of years on a 2.4 GHz Intel Core 2 Quad Q6600
2009	77	83	(1634, 1217, 39) 62	1323 1024	145	157	$8.52 \cdot 10^{11}$	$1.94 \cdot 10^{7}$
2010	78	84	(1635, 1197, 41) 64	1369 1056	146	160	$1.45 \cdot 10^{12}$	$3.30 \cdot 10^{7}$
2011	79	85	(1652, 1203, 42) 66	1416 1088	148	163	$2.47 \cdot 10^{12}$	$5.61 \cdot 10^{7}$
2012	80	87	(1687, 1226, 43) 69	1464 1120	149	165	$4.19 \cdot 10^{12}$	$9.52 \cdot 10^{7}$
2013	80	88	(1702, 1219, 45) 72	1513 1184	151	168	$7.14 \cdot 10^{12}$	$1.62 \cdot 10^{8}$
2014	81	89	(1770, 1306, 43) 74	1562 1216	152	172	$1.21 \cdot 10^{13}$	$2.75 \cdot 10^{8}$
2015	82	90	(1823, 1368, 42) 76	1613 1248	154	173	$2.07 \cdot 10^{13}$	$4.70 \cdot 10^{8}$
2016	83	91	(1833, 1356, 44) 79	1664 1312	155	177	$3.51 \cdot 10^{13}$	$7.98 \cdot 10^{8}$
2017	83	92	(1845, 1356, 45) 81	1717 1344	157	180	$5.98 \cdot 10^{13}$	$1.36 \cdot 10^{9}$
2018	84	93	(1877, 1387, 45) 83	1771 1376	158	181	$1.02 \cdot 10^{14}$	$2.32 \cdot 10^{9}$
2019	85	95	(1951, 1481, 43) 85	1825 1440	160	185	$1.73 \cdot 10^{14}$	$3.93 \cdot 10^{9}$
2020	86	96	(1955, 1463, 45) 88	1881 1472	161	188	$2.94 \cdot 10^{14}$	$6.68 \cdot 10^{9}$
2021	86	97	(1983, 1479, 46) 91	1937 1536	163	190	$5.01 \cdot 10^{14}$	$1.14 \cdot 10^{10}$
2022	87	98	(2013, 1508, 46) 93	1995 1568	164	193	$8.52 \cdot 10^{14}$	$1.94 \cdot 10^{10}$
2023	88	99	(2018, 1491, 48) 96	2054 1632	166	197	$1.45 \cdot 10^{15}$	$3.30 \cdot 10^{10}$
2024	89	101	(2104, 1596, 46) 99	2113 1696	167	198	$2.47 \cdot 10^{15}$	$5.61 \cdot 10^{10}$
2025	89	102	(2106, 1576, 48) 102	2174 1728	169	202	$4.20 \cdot 10^{15}$	$9.55 \cdot 10^{10}$
2026	90	103	(2135, 1604, 48) 104	2236 1792	170	205	$7.14 \cdot 10^{15}$	$1.62 \cdot 10^{11}$
2027	91	104	(2157, 1614, 49) 107	2299 1856	172	207	$1.21 \cdot 10^{16}$	$2.75 \cdot 10^{11}$
2028	92	105	(2198, 1654, 49) 110	2362 1888	173	210	$2.07 \cdot 10^{16}$	$4.70 \cdot 10^{11}$
2029	93	106	(2220, 1664, 50) 113	2427 1952	175	213	$3.52 \cdot 10^{16}$	$8.00 \cdot 10^{11}$
2030	93	108	(2241, 1673, 51) 116	2493 2016	176	215	$5.98 \cdot 10^{16}$	$1.36 \cdot 10^{12}$
2032	95	110	(2344, 1784, 50) 122	2629 2144	179	222	$1.73 \cdot 10^{17}$	$3.93 \cdot 10^{12}$
2034	96	112	(2440, 1877, 50) 129	2768 2272	182	227	$5.01 \cdot 10^{17}$	$1.14 \cdot 10^{13}$
2036	98	115	(2496, 1920, 51) 135	2912 2400	185	232	$1.45 \cdot 10^{18}$	$3.30 \cdot 10^{13}$
2038	99	117	(2440, 1776, 59) 144	3061 2528	188	239	$4.20 \cdot 10^{18}$	$9.55 \cdot 10^{13}$
2040	101	119	(2521, 1854, 59) 151	3214 2656	191	244	$1.22 \cdot 10^{19}$	$2.77 \cdot 10^{14}$
2042	103	122	(2623, 1964, 58) 158	3371 2784	194	248	$3.52 \cdot 10^{19}$	$8.00 \cdot 10^{14}$
2044	104	124	(2662, 1979, 60) 165	3533 2944	197	255	$1.02 \cdot 10^{20}$	$2.32 \cdot 10^{15}$
2046	106	126	(2691, 1973, 63) 173	3700 3072	200	260	$2.95 \cdot 10^{20}$	$6.70 \cdot 10^{15}$
2048	107	129	(2798, 2088, 62) 181	3871 3232	203	265	$8.53 \cdot 10^{20}$	$1.94 \cdot 10^{16}$
2050	109	131	(2804, 2048, 66) 189	4047 3392	206	272	$2.47 \cdot 10^{21}$	$5.61 \cdot 10^{16}$

Lattice-Based Cryptography

M. Ajtai, "Generating hard instances of lattice problems", In Proc. of 28th STOC, pp. 99-108, ACM 1996.

0. Goldreich, S. Goldwasser, S. Halevi, "Public-key cryptosystem from lattice reductions problem", In Proc. of Crypto '97, volume 1294 of LNCS pp. 112-131, IACR, Springer-Verlag, 1997.

J. Hofftein, J. Pipher, J.H. Silverman, "NTRU: a ring based public key cryptosystem", In Proc. of ANTSIII , volume 1423 pp. 267-288, Springer-Verlag 1997..

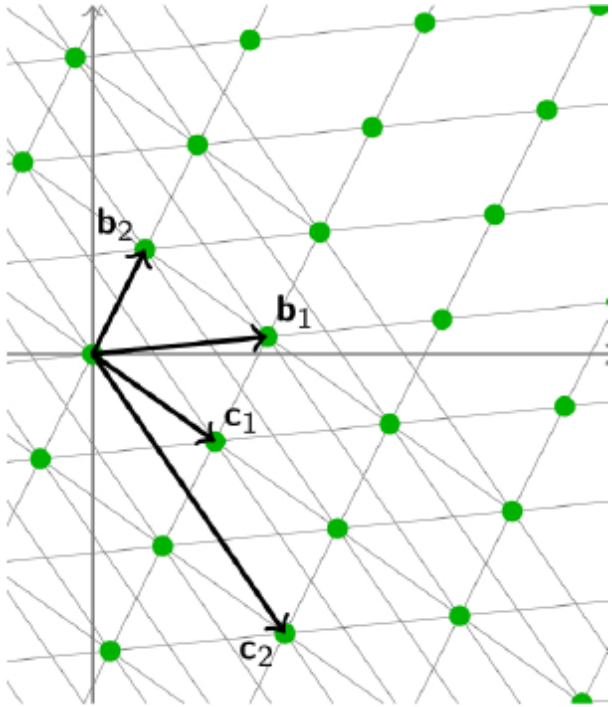
Reticoli

Un **reticolo** (o ***lattice***) in $\mathbb{Z}^n$ è l'insieme di tutte le combinazioni lineari intere dei vettori di una base intera $(b_1, \ldots, b_n)$:

$$\mathcal{L} := \sum b_i \cdot \mathbb{Z} = \{B \cdot x |\ x_i \in \mathbb{Z}, b_i \in \mathbb{Z}^n\}$$

- n si dice **dimensione**.
- $B = [b_1, \ldots, b_n] \in \mathbb{Z}^{n \times n}$

Ogni reticolo ha **molte** basi.



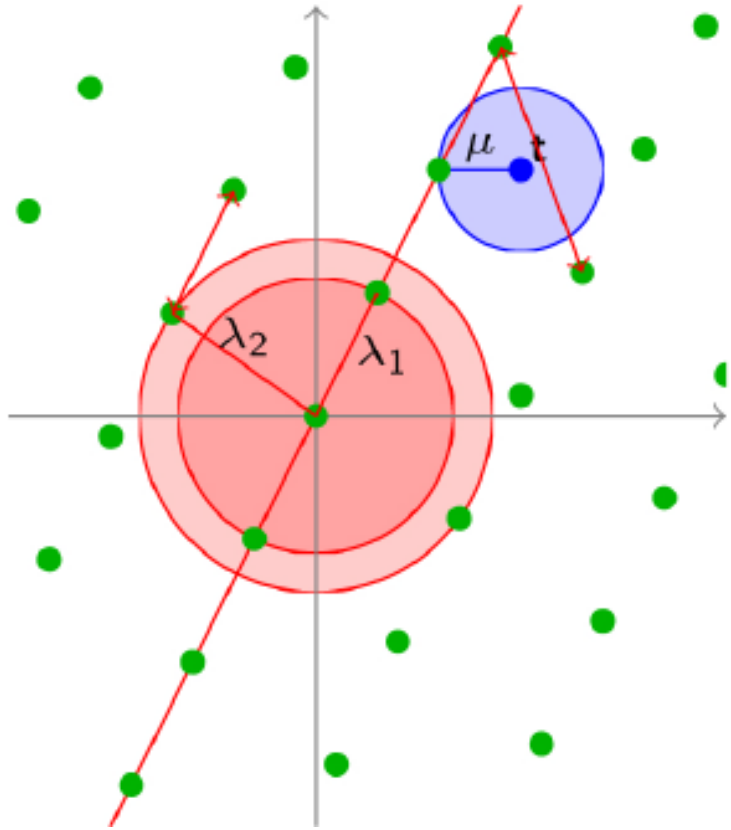
Reticoli

- Distanza minima:

$$\lambda_1 = \min_{x,y \in \mathcal{L},\, x \neq y} ||x - y||$$
$$= \min_{x \in \mathcal{L},\, x \neq 0} ||x||$$

- Distance function:

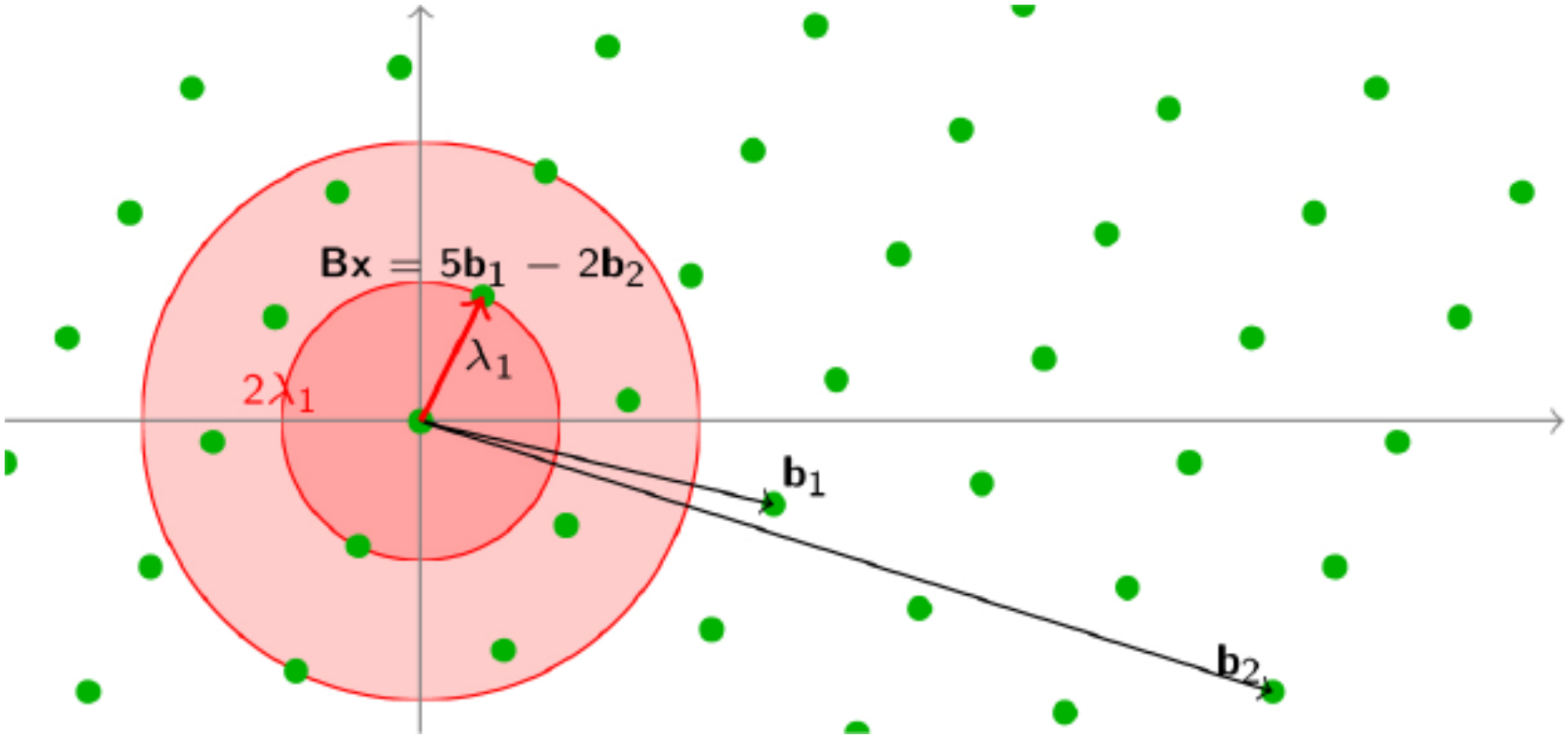
$$\mu(t, \mathcal{L}) = \min_{x \in \mathcal{L}} ||t - x||$$



Shortest Vector Problem

Definizione SVP (Shortest Vector Problem)

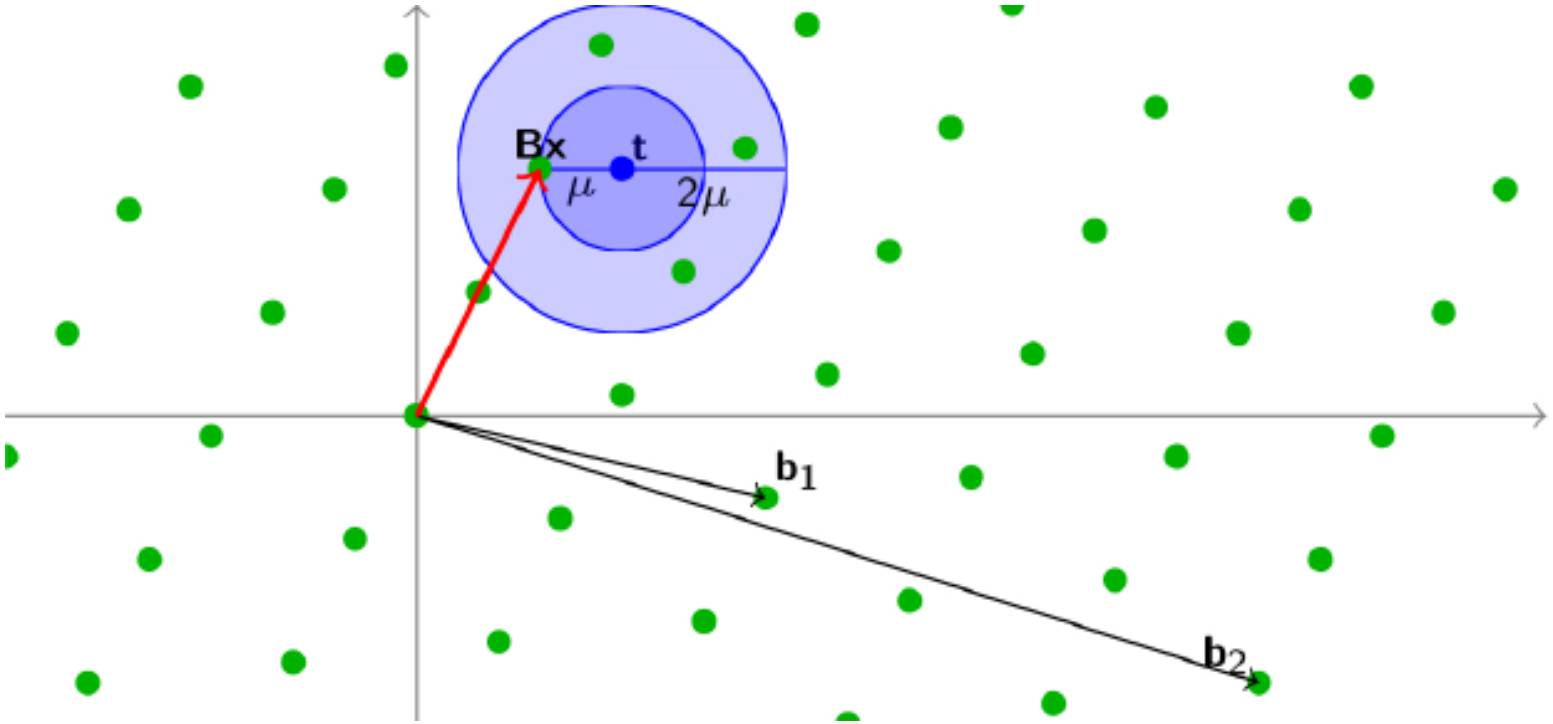
Data una base $B \in \mathbb{Z}^{n \times n}$ di $\mathcal{L}$, trovare il vettore non nullo del reticolo Bx (con $x \in Z^n \setminus \{0\}$) di lunghezza al più $||Bx|| \leq \lambda_1$.



Closest Vector Problem

Definizione CVP (Closest Vector Problem)

Data una base $B \in \mathbb{Z}^{n \times n}$ di $\mathcal{L}$ ed un vettore $t \in \mathbb{Z}^n$, trovare il vettore del reticolo Bx più vicino a t, i.e. trovare un vettore intero $x \in \mathbb{Z}^n$ tale che $||Bx - t|| \leq \mu(t, \mathcal{L})$.



Problemi su reticoli

- Le versioni esatte di questi problemi sono NP-hard, quindi possono essere utilizzate per creare crittosistemi resistenti.
- Algoritmi di riduzione classici (LLL, BKZ): gli algoritmi di riduzione producono "vettori relativamente corti" in tempo polinomiale.
  A. K. Lenstra, H. W. Jr. Lenstra, L. Lovász, "Factoring polynomials with rational coefficients, Mathematische Annalen 261 (4) : 515-534, 1982.
- Ridurre un reticolo vuol dire trovare una "buona base", una base che permetta di approssimare problemi come lo SVP e il CVP, oppure di risolverli in modo esatto usando ulteriori procedimenti.

Multivariate Cryptography

T. Matsumoto, H. Imai, "A class of asymmetric cryptosystems based on polynomials over finite rings", IEEE International Symposium on Information Theory, Abstract of Papers, pp.131-132, September 1983.

J. Patarin, "Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms", Eurocrypt'96, Springer Verlag, pp. 33-48.

Schema generale

Supponiamo che Alice voglia trasmettere a Bob il messaggio $m = (m_1, \ldots, m_n)$.

Bob costruisce un sistema di l polinomi a coefficienti in $\mathbb{F}_q$, A, che **sa risolvere**:

$$A := \begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ f_2(x_1, \ldots, x_n) = 0 \\ \vdots \\ f_l(x_1, \ldots, x_n) = 0 \end{cases}$$

Il sistema A costituisce la chiave pubblica di Bob, la sua chiave segreta è il metodo con cui costruisce il sistema.

Codifica e decodifica

- Alice valuta m nelle equazioni del sistema calcolando $f_i(m1, \ldots, m_n)$ per $i = 1, \ldots, l$.
- Alice manda a Bob $(f_1(m_1, m_2, \ldots, m_n), \ldots, f_l(m_1, m_2, \ldots, m_n))$.
- Bob sa risolvere A e calcola il messaggio iniziale.

La sicurezza di questo crittosistema dipende dalla difficoltà della risoluzione del sistema per chi non lo ha costruito. Un modo di costruirlo è quello di usare un sistema di equazioni con grado almeno 2.

Questa famiglia di schemi sono basati su *una rappresentazione oscura* dei polinomi.

T. Matsumoto, H. Imai, H. Harashima, H. Miyakawa, "Asymmetric cryptosystems using obscure representations of enciphering functions", Natl. Conf. Re. On Inf. Syst., IECE Japan, S8-5, 1983..

Sicurezza

Per attaccare un crittosistema costruito su polinomi possiamo:

- tentare di ricostruire direttamente la chiave segreta.
- utilizzare le Basi di Groebner per risolvere il sistema A.

Importante

Risolvere un sistema di n equazioni in n variabili di grado 2 su un piccolo campo è molto spesso impraticabile: già con $n \approx 100$ non ci sono metodi migliori della ricerca esaustiva.

Questo stesso problema è quello che rende sicuro rispetto ai computer quantistici l'attuale crittografia simmetrica (e.g. AES)

Sfide attuali

- Il **NIST** (National Institute of Standards and Technology) ha lanciato recentemente la gara per avere uno standard di crittografia post-quantistica: http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/pqcrypto-2016-presentation.pdf
- **Google** sta iniziando a introdurre la crittografia post-quantistica nei suoi programmi: https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html

La crittografia post-quantum è uno degli argomenti più "caldi" del momento!

GRAZIE
PER L'ATTENZIONE

CryptoLabTN