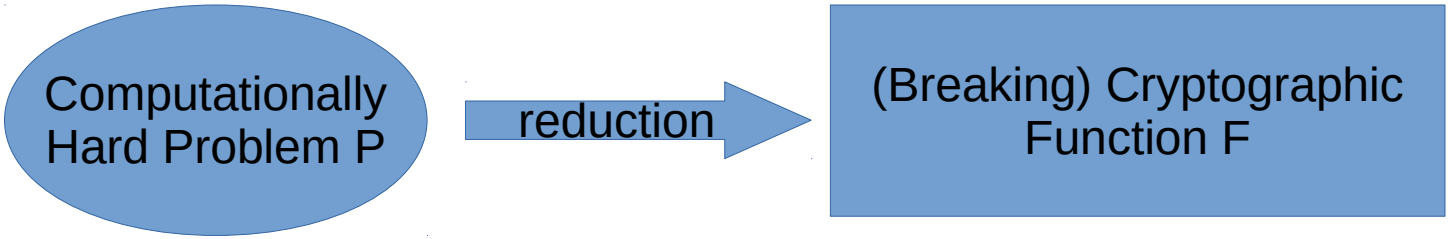
Faster Gaussian Sampling for Trapdoor Lattices (with any modulus)

March 2017

Daniele Micciancio (UCSD)
Nicholas Genise (UCSD)

Modern Cryptography

- Founded on Mathematics / Complexity Theory
  - Start from mathematical problem P that is computationally “hard” to solve
  - Build cryptographic function F that is “hard” to break
  - Prove security via a reduction:
    - if you can break F, then you can solve P



- “The happy side of Computational Complexity”

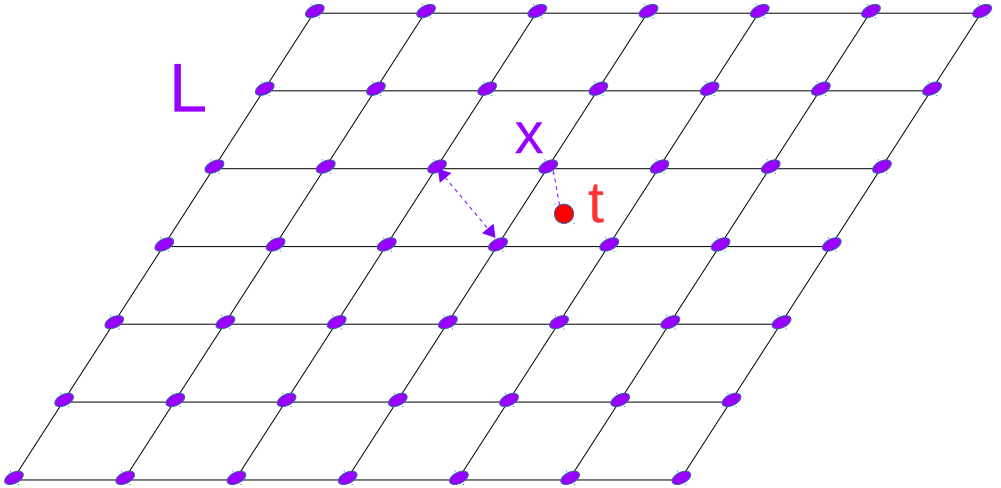
Cryptographic Hardness

- Requirements:
  - Hard for adversary: very hard to solve on the average even with small probability
  - Useful to users: algebraic structure to embed trapdoors, etc.
- Examples:
  - Factoring numbers
  - Discrete logarithm problem in finite fields and elliptic curves
- Balancing hardness with structure is difficult
  - e.g., the structure that makes factoring and discrete log problems useful, also opens the door to (polynomial time) quantum attacks.

Lattice Cryptography

- Computationally hard:
  - Average-case/worst-case connection
  - Strong pseudorandomness properties
  - Conjectured security against quantum attacks
- Useful/Efficient
  - Based on simple operations (vector addition and multiplication by small integers)
  - Powerful linear structure, with many applications
    - Public Key Encryption, Digital Signatures, Group Signatures, Identity Based Encryption, Fully Homomorphic Encryption, Attribute Based Encryption, Trapdoor Key Delegation, and much more

Lattices in Cryptography

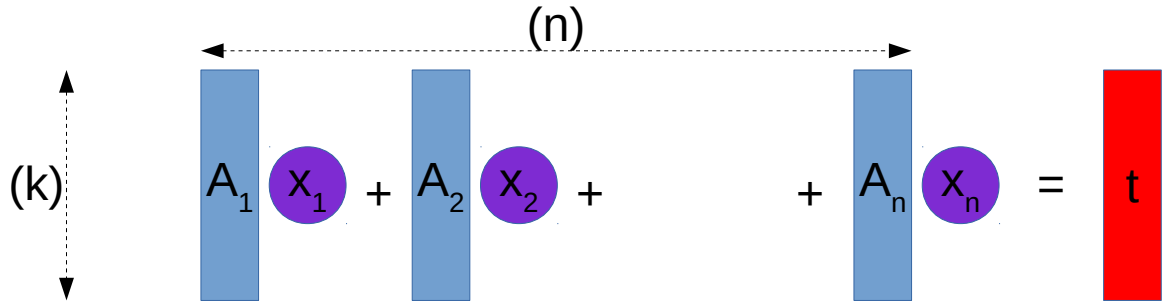


- Integer lattices
  - L: subgroup of Z^n
- *q*-ary lattices
  - Periodic modulo q
  - Usually described as

    L = {x: Ax=0 mod q} (A in Z^{kxn})
- Lattice problems:
  - Find “small” nonzero solution x of Ax=0 mod q
  - Find “small” solution x to Ax=t mod q
- Basic crypto operation: vector addition modulo (small) q

Simple, still hard

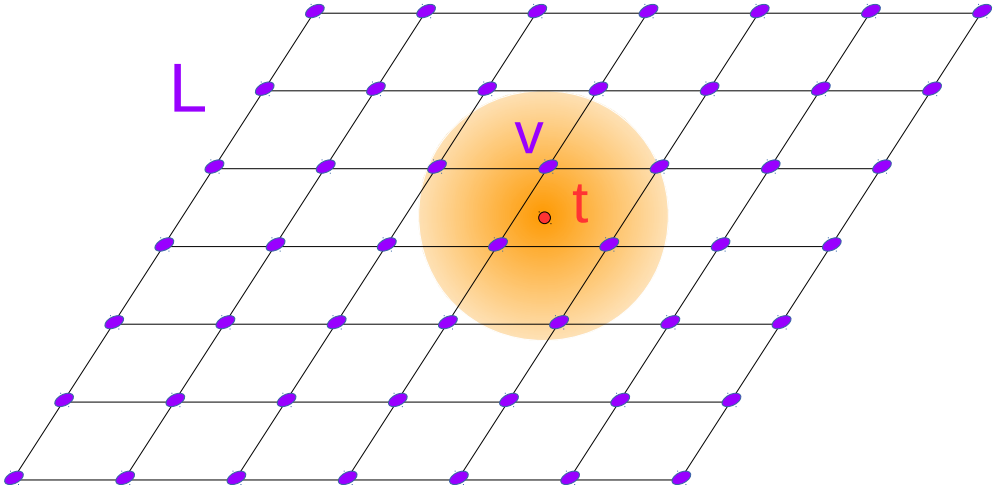
- Problem:
  - Find “small” solution x to Ax=t mod q
- Integer solutions are easy to find by Gauss elimination, but solution is not “small”
- Finding binary solutions x in $\{0,1\}^n$:
  - Equivalent to subsetsum in the group Z^k



- Lattice crypto: small euclidean length ||x||<q/poly(n)

Today: Lattice Sampling Problem

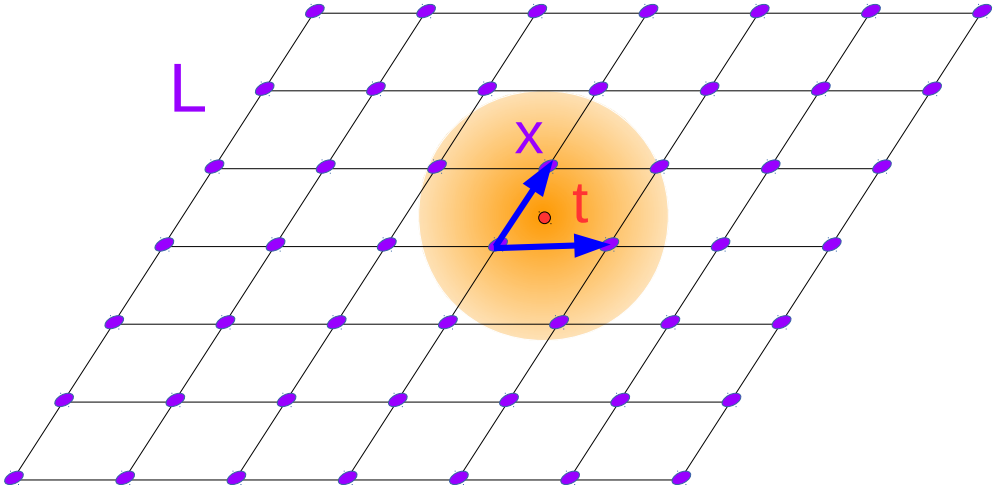
- Input:
  - Lattice L (mod q)
  - trapdoor information T
  - Target vector t



- Output:
  - Lattice point v in L with gaussian distribution around target t
- Applications:
  - Used by most advanced lattice cryptographic primitives, allowing some form of trapdoor “delegation”

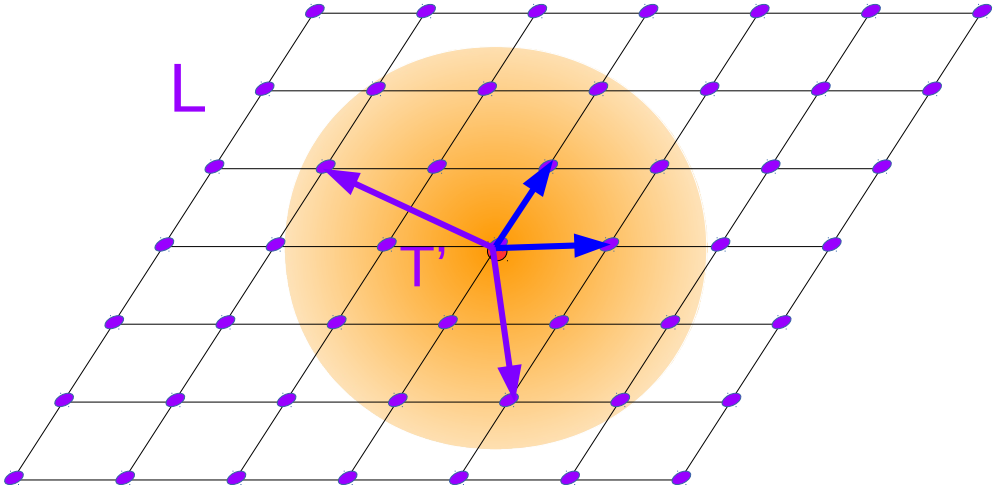
Lattice Trapdoors

- Lattice L (mod q)
- Example trapdoor: T
  - Linearly independent
  - Short euclidean length
- Using T:
  - Small solution to Ax=t (mod q) can be found by “rounding” t wrt T
- Applications:
  - Decryption operation in Public-Key Encryption
  - “Hash and Sign” signatures

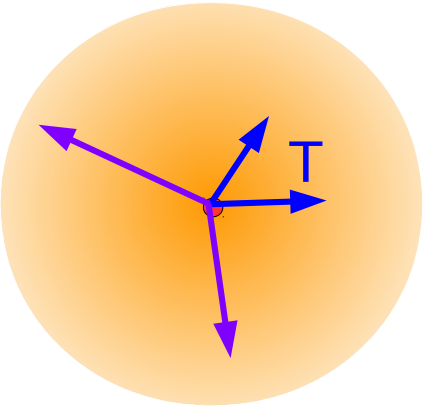


Application of Gaussian Sampling

- Lattice L (mod q)
- Trapdoor: T
  - Linearly independent
  - Short euclidean length
- Trapdoor quality:
  - $||T|| = \max \{||t_i||: i=1..n\}$
- Use T to generate a “weaker” trapdoor T’
  - $||T'|| > ||T||$, but still short
  - Can be used for restricted operations

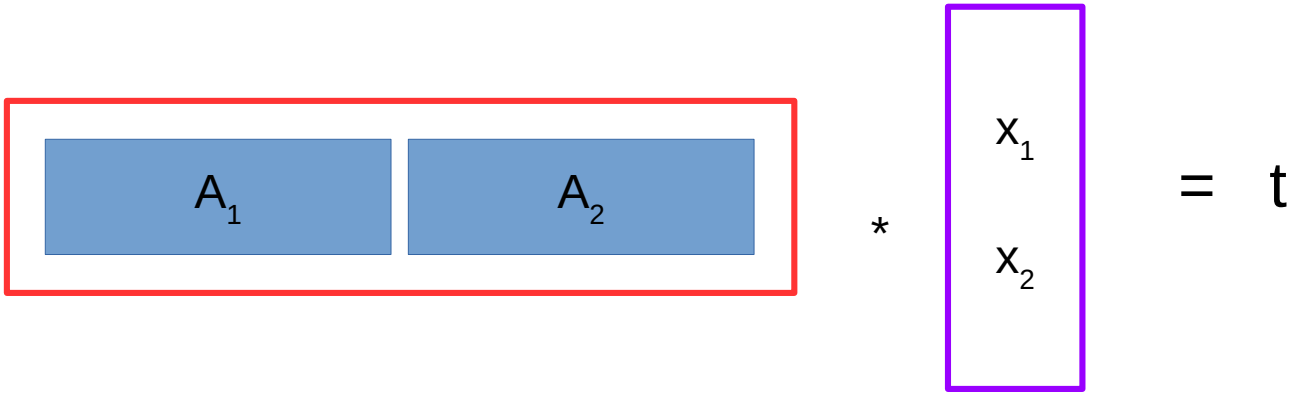


Why using Gaussians?



- Efficiency:
  - product of n independent 1-dimensional gaussians
  - Generating n-dimensional gaussian reduces to generating n samples from distribution with small support
- Security:
  - Spherically symmetric
  - Does not depend on the geometry of T
  - Samples can be revealed without leaking information about trapdoor T

Structured sets of keys/trapdoors

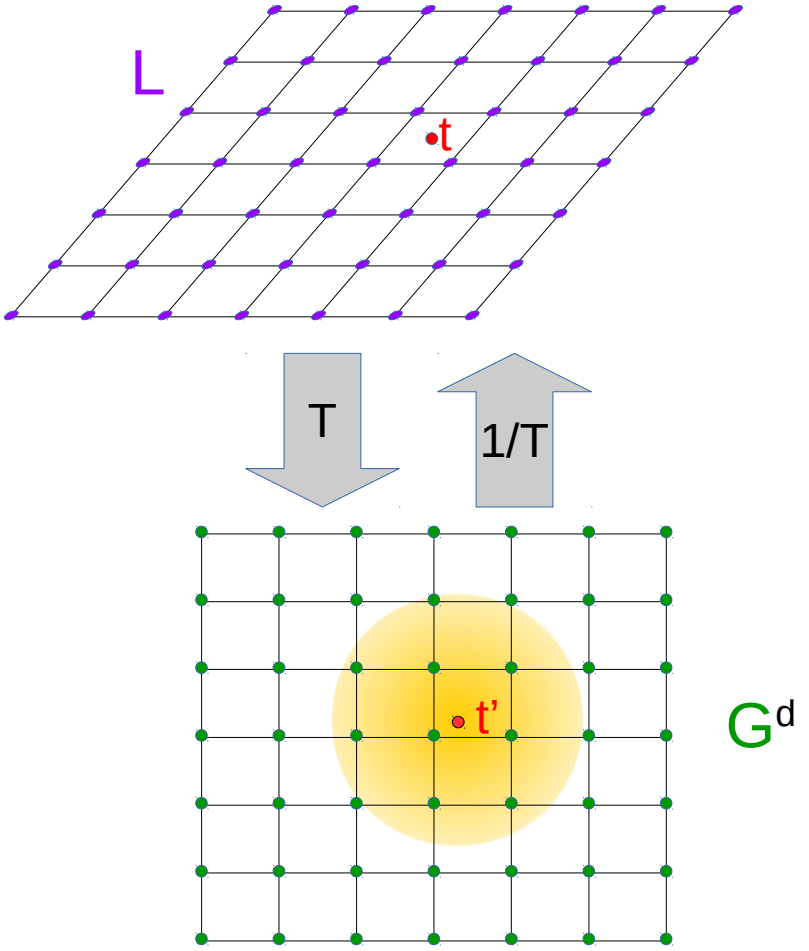


- Given B=[A_1,A_2] and t, sample {x | Bx=t mod q}
- Two independent trapdoors:

  A_1T_1=0 mod q A_2T_2=0 mod q
- Sample x=(x_1,x_2) using A_1:
  - Choose x_2 with gaussian distribution from Z^n
  - Choose x_1 from {x_1 | $A_1x_1 = t - A_2x_2$}
- Similarly for A_2: distribution on x=(x_1,x_2) is the same

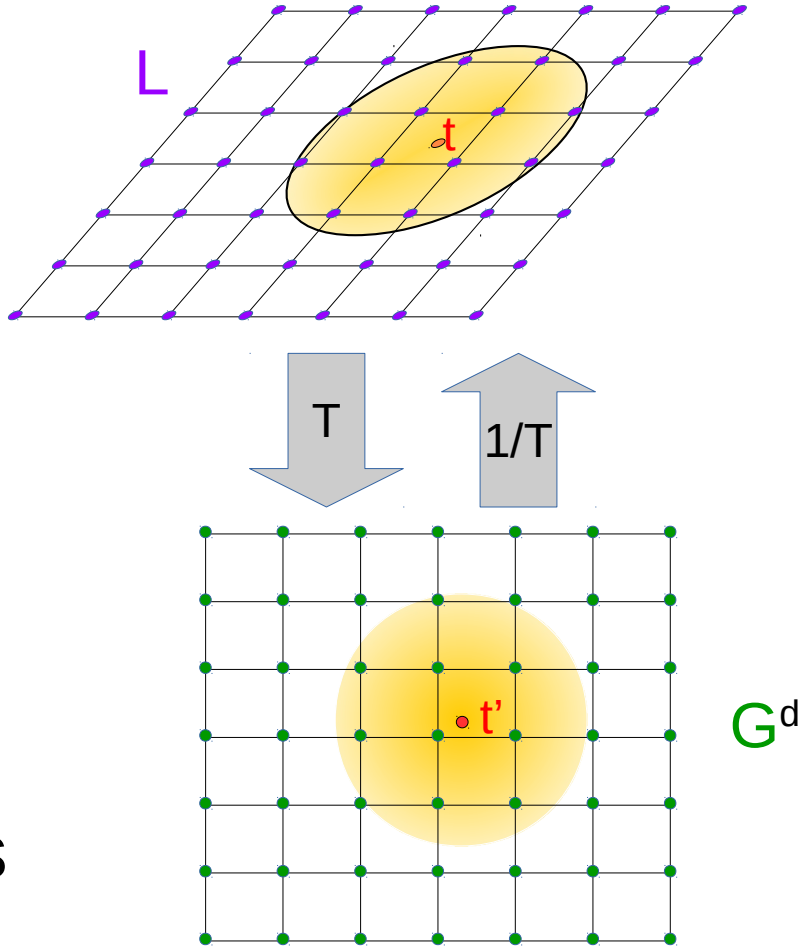
MP12 Trapdoors and Sampling

- Trapdoor T: L → G^d
- Building blocks:
  - T, 1/T: easy to compute
  - G: easy to sample
- Steps:
  - Map t to t’
  - Sample x’ in G^d around t’
  - Map x’ back to x
-



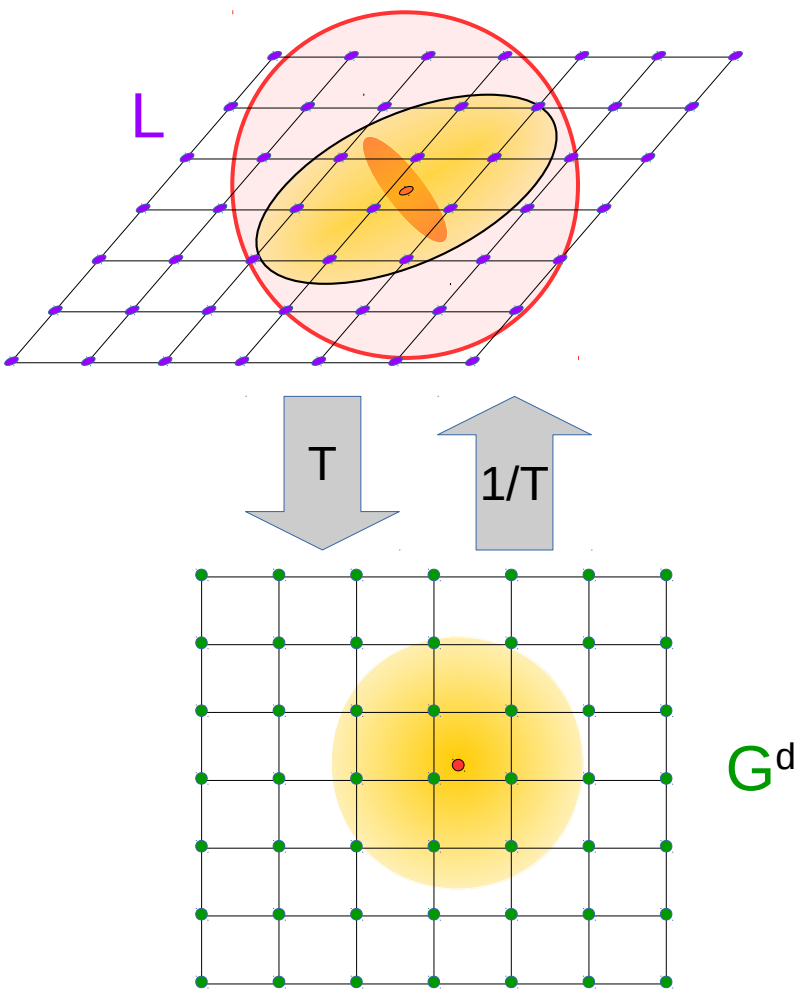
MP12 Trapdoors and Sampling

- Trapdoor T: L → G^d
- Building blocks:
  - T, 1/T: easy to compute
  - G: easy to sample
- Steps:
  - Map t to t’
  - Sample x’ in G^d around t’
  - Map x’ back to x
- Problem: Distribution of x is no longer spherical



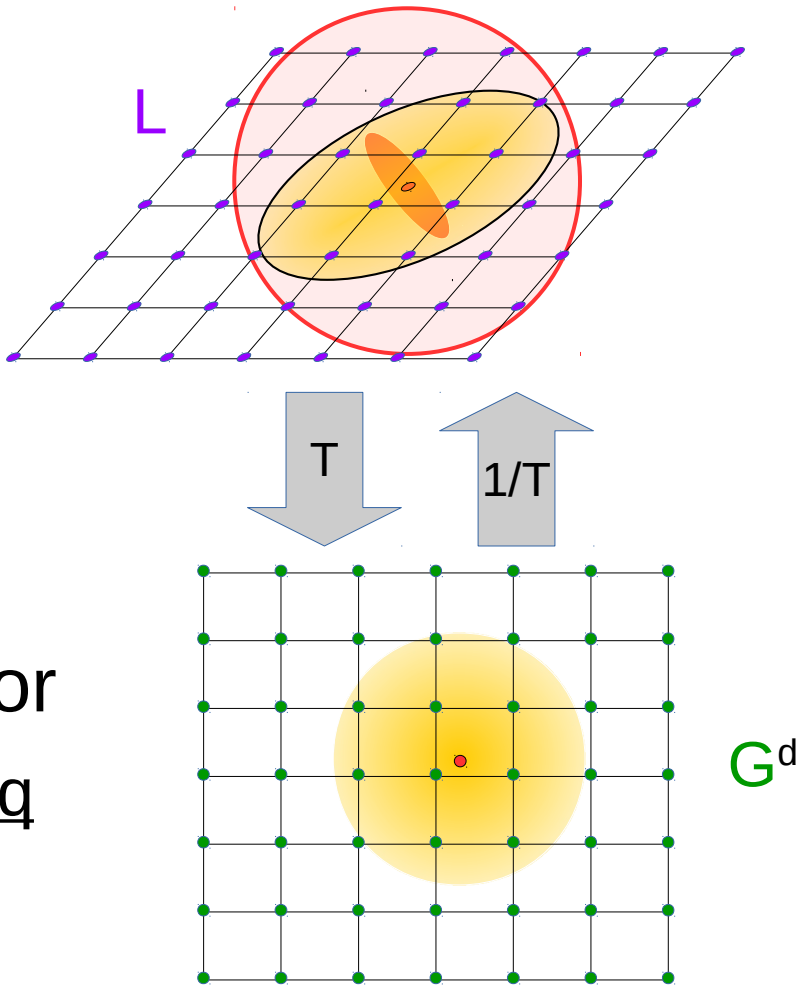
MP12 Trapdoors and Sampling

- Trapdoor T: L → G^d
- Building blocks:
  - T, 1/T: easy to compute
  - G: easy to sample
- Steps:
  - Map t to t’
  - Sample x’ in Gd around t’
  - Map x’ back to x
- Solution: Add corrective perturbation e to t before mapping/sampling



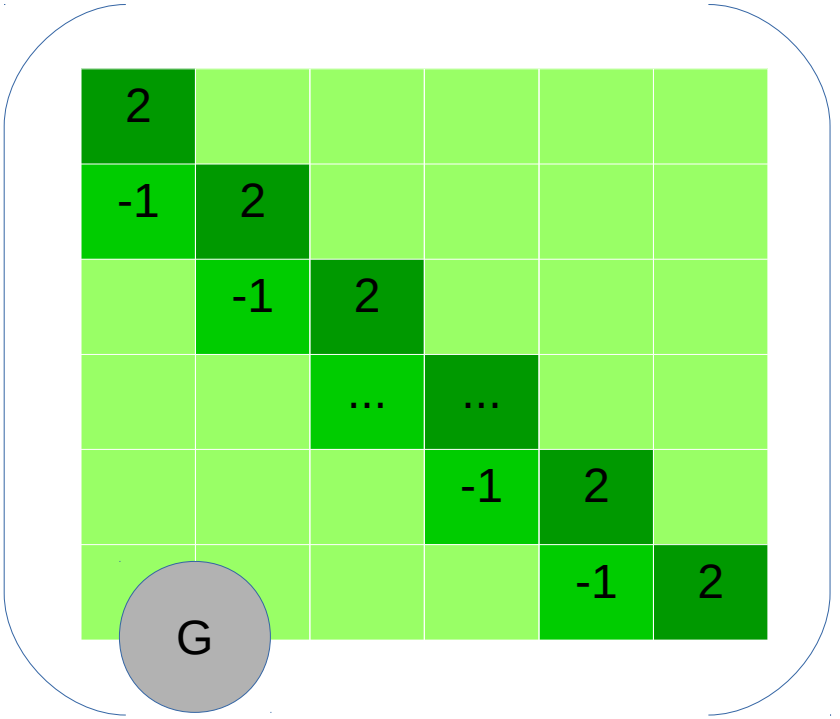
MP12: Summary

- Trapdoor T: L → G^d
- Corrective perturbation: e
- Building blocks:
  - T, 1/T: easy to compute
  - G: easy to sample
  - Generate perturbation: e
- **Our work**:new algorithms for
  - Efficient G-sampling for any q
  - Efficient perturbation generation in ring lattices



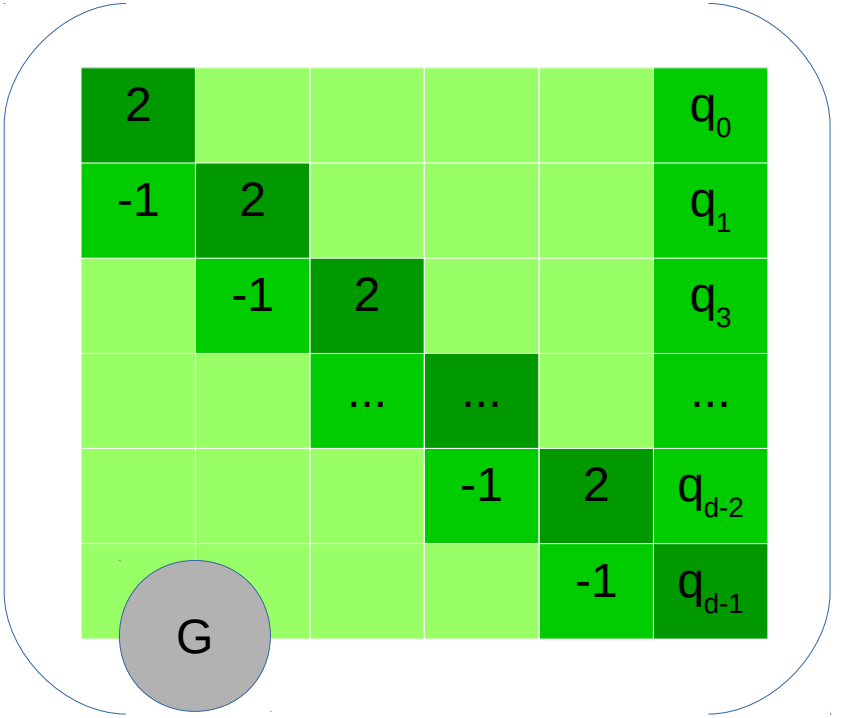
MP12: G Lattice modulo $q=2^d$

- G Basis:
  - Sparse
  - (Lower) Triangular
- Preimage Sampling:
  - Orthogonalize: G*
  - Sample d independent 1-dim Gaussians.
  - Running time: O(d)
- Remark:
  - Works for any $q=b^d$

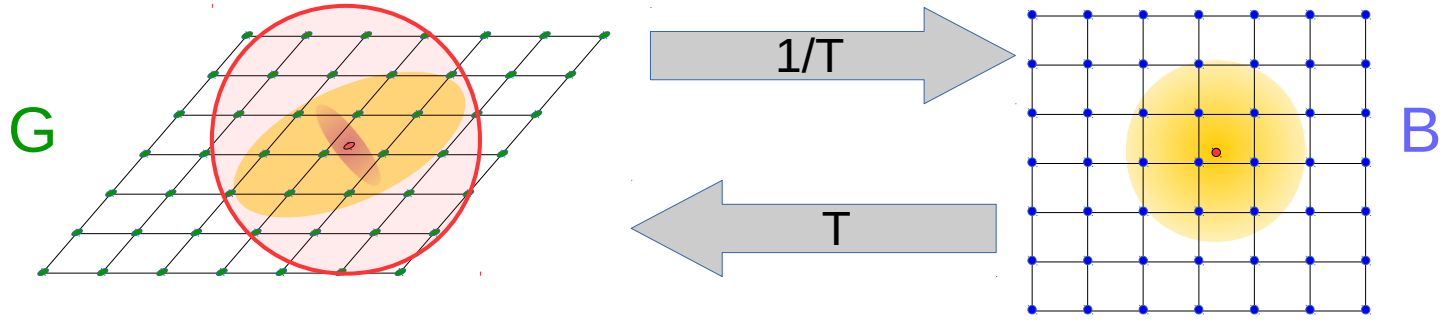


G Lattice: modulo $q=\Sigma_{i<d}\ q_i 2^i$

- G Basis:
  - Sparse
  - But not Triangular!
- G* is dense
  - Requires $O(d^3)$ time and $O(d^2)$ storage to compute
  - Sampling: $O(d^2)$ time
- In some applications:
  - $q=\Sigma_{i<d}\ q_i 2^i$
  - $q=2^{O(n)},\ d=O(n)$

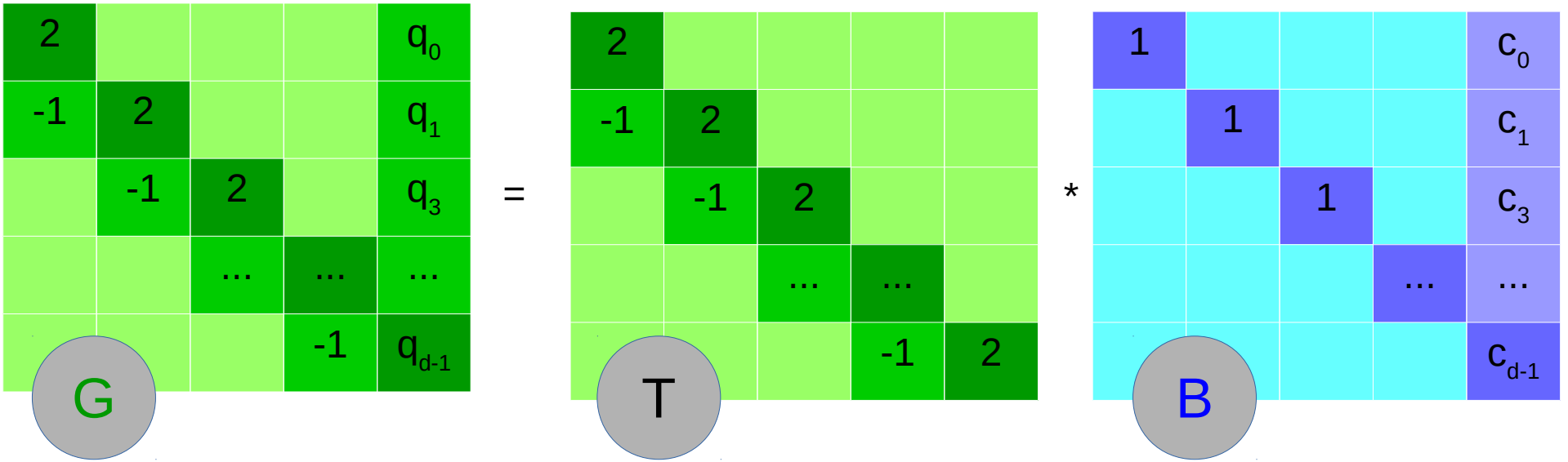


Our G sampler for $q=\Sigma_{i<d}\ q_i 2^i$



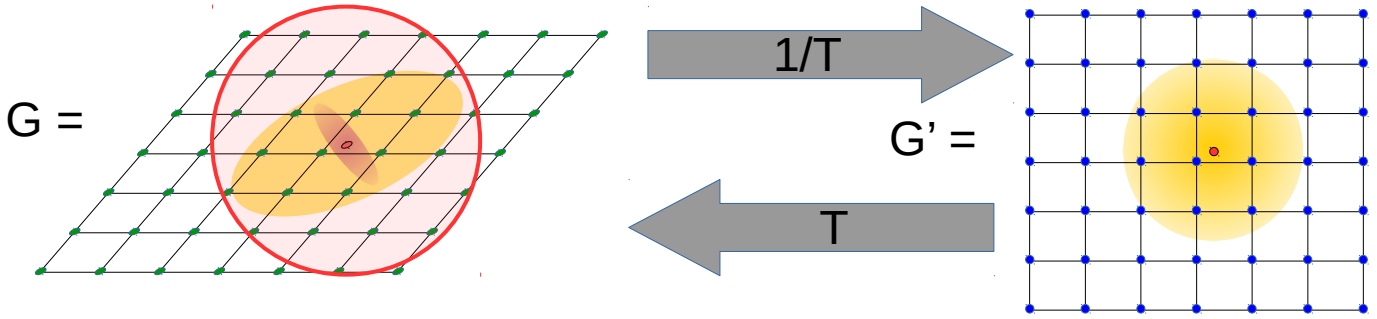
- Transfor G lattice to even simper lattice B
- Requirements:
  - T,1/T: easy to compute
  - B: easy preimage sampling (sparse, triangular)
  - Perturbation: easy to generate

Transformation and Sampling:



- B: sparse, (upper) triangular
  - Efficient Preimage Sampling: O(d) time, no storage
- Transformation T: sparse, triangular
  - Efficient to compute: O(d) time, no storage
- Inverse Transformation 1/T: dense, but …
  - Still easy to compute by back substitution: O(d) time, no storage

G/B Perturbation



- Need to generate perturbations with covariance

$$C = s^2 I - TT^{tr} =$$

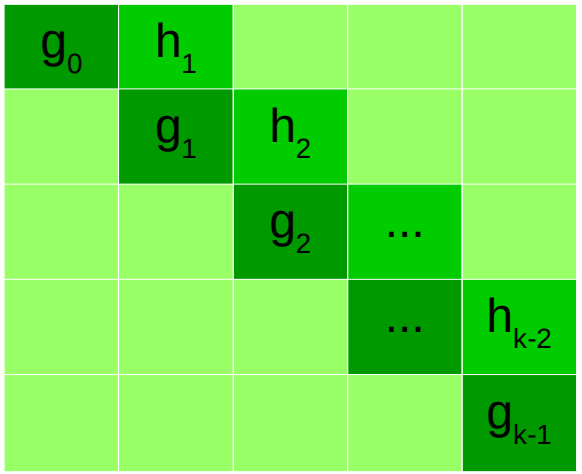
s^2-4	2			
2	s^2-5	2		
	2	s^2-5	2	
		...	...	2
			2	s^2-5

- Uses Cholesky decomposition of $C = LL^{tr}$

Cholesky Decomposition

- $C=LL^{tr}$, where L is (upper or lower) triangular
- Can be computed numerically in $O(d^3)$ time
- Closed formula for upper triangular with s=3:

$L =$



$g_0^2 = 3 + 2 / d$

$g_i^2 = 2 + 2 / (d-i)$

$h_{i+1}^2 = 2 – 2 / (d-i)$

- Sparse, Triangular, easy to compute with O(d) operations on O(log d)-bit numbers!

G Lattice Sampling: Summary

- New algorithm for gaussian sampling in G-lattices with arbitrary modulus $q=\Sigma_{i<d}\ q_i 2^i$
  - New algorithm also generalizes to arbitrary b
  - Just as efficient as MP12 algorithm for power $q=b^d$

	Preproc. Time	Preproc. Size	On-line Time
[MP12] $q=b^d$	0	0	O(d)
[MP12] any q	$O(d^3)$	$O(d^2)$	$O(d^2)$
[New] any q	0	0	O(d)

- Already implemented and running in PALISADE lattice library

Generating the correction term

- Recall: $\mathrm{cov}(x) = E[x * x^{tr}]$
  - Start from spherical sample x in G lattice
  - Apply trapdoor transformation T
  - Result has covariance

    $\mathrm{cov}(Tx) = Tx * (Tx)^{tr} = T\,(x * x^{tr})\,T^{tr} = T * T^{tr}$
- Fix: add perturbation e with complementary covariance

  $C = s^2 I - T * T^{tr}$
  - $\mathrm{cov}(Tx + e) = \mathrm{cov}(Tx) + \mathrm{cov}(e) = T * T^{tr} + C = s^2 I$
- Technicality:
  - Instead of adding e to Tx …
  - Sample Tx around e

Generating correction terms:

- Problem: generate vectors with covariance

$$C = s^2 I - T * T^{tr}$$

- Standard Solution:
  - Compute Cholesky decomposition $C = LL^{tr}$
  - Generate sperical gaussian x, compute Lx, and round each coordinate
- Performance: $O((dn)^2)$ even after $O((dn)^3)$ preprocessing
- Cannot do much better because T takes $O(dn^2)$ time just to read

Faster perturbations for algebraic lattices

- Use matrix T with special structure:
  - Each block of T has anticirculant structure
  - Equivalently, use matrix T over ring $R=Z[x]/(x^n+1)$
  - If $n=2^k$, still hard lattice problem, worst-case/average-case connection, etc.
- Now T takes only O(dn) storage, but we still have a problem:
  - Cholesky decomposition $C = LL^{tr}$ destroys the ring structure

Previous solution

- Ducas and Nguyen [DN12]
  - If C is over $R=Z[x]/(x^n+1)$,
  - then C=SS for a symmetrix matrix S over R, and
  - S can be computed using Newton iteration
- Asymptotically efficient, but rather complex
  - Requires computing S over the reals, and
  - Gaussian rounding each coordinate of Sx to an integer
- We propose an alternative, more direct method

Main idea

- Compute Cholesky decomposition only implicitly, in a sequence of recursive stages
- Each stage: Use **“block”** version of **Cholesky**
  - Sample x with covariance $A=A^{tr}$
  - Sample y with covariance $D-BA^{-1}B^{tr}$

    (Shur complement of A)
  - Output $(I, BA^{-1})x + (0,I)y$ with covariance

$$\begin{bmatrix} I \\ BA^{-1} \end{bmatrix} \cdot A \cdot \begin{bmatrix} I & A^{-tr}B^{tr} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} \cdot D \cdot \begin{bmatrix} 0 & I \end{bmatrix} = \begin{bmatrix} A & B^{tr} \\ B & D \end{bmatrix} = C$$

Stage 1: $R_n^{d+2} \rightarrow R_n^2$

- Input: $$C = s^2 I - \begin{bmatrix} I \\ T \end{bmatrix} \cdot \begin{bmatrix} I & T^{tr} \end{bmatrix} = \begin{bmatrix} s^2 I & -T^{tr} \\ -T & s^2 I - TT^{tr} \end{bmatrix}$$
- Recursive calls:
  - x ← Perturbation(A = $s^2 I$)
  - y ← Perturbation(D-BA^{-1}B^{tr} = $s^2 I - (1+1/s^2)TT^{tr}$)
- Output: (x,y-Tx/s^2)
- Cost:
  - Sample x in R_n^d (Easy!!!)
  - Recursive call for y in R_n^2
  - Compute product Tx with T in $R_n^{2 \times d}$ (Fast, using FFT!)

Stage 2: $R_n^2 \rightarrow R_n + R_n$

- Input: $C = \begin{bmatrix} a & b^{tr} \\ b & d \end{bmatrix} \in R_n^{2\times 2}$
- Recursive calls:
  - x ← Perturbation(a)
  - y ← Perturbation($d-ba^{-1}b^{tr}$)
- Output: (x,y-(ba^{-1})x)
- Cost:
  - 2 recursive calls in R_n
  - 1 product in R_n (Fast, using FFT)

Stage 3+: $R_n \rightarrow R_{n/2}^2$

- Use ring embedding: $c \in R_n \rightarrow \begin{bmatrix} a & b^{tr} \\ b & d \end{bmatrix} \in R_{n/2}^{2\times 2}$

- Proceed as in step 2:
  - $R_{n/2}^2 \rightarrow 2 \text{ x } R_{n/2}$
- Recurse:
  - $1\text{x } R_n \rightarrow 2\text{x } R_{n/2} \rightarrow 4\text{x } R_{n/4} \rightarrow \ldots \rightarrow n\text{x } R_{n/n} = n$
- Remarks:
  - Ring structure is gradually destroyed, when it is no longer need
  - Total cost: O(n log(n))

Ring Perturbations: Summary

- New algorithm to generate perturbations in trapdoor lattices on 2-power cyclotomic rings

	Preproc. Time	Preproc. Size	On-line Time
Naive	$O((dn)^3)$	$O((dn)^2)$	$O((dn)^2)$
[BB13]	$O(dn^3)$	$O(dn^2)$	$O(dn^2)$
[New]	0	0	$O(dn)$

- Performance similar to [BB13]+[DN12], but no need to use high precision numerical iterations,

Thank you!

For details, paper draft available on my webpage

http://www.cse.ucsd.edu/~daniele