

An overview of Cold-Boot Attack, related to

RSA and Factorization

SOURAV SEN GUPTA

Indian Statistical Institute, Kolkata



About this talk

Based on the work “*Reconstruction from Random Bits and Error Correction of RSA Secret Parameters*”, jointly done with



Santanu Sarkar

&

Subhamoy Maitra



This extends and supplements the work of *Heninger and Shacham* [Crypto 2009] and that of *Henecka, May and Meurer* [Crypto 2010].

Contents of this talk

- Cold-Boot attack - a brief introduction
- Application 1: Reconstruction of RSA secret parameters
 - Starting from the LSB side [Heneinger and Shacham, 2009]
 - Starting from the MSB side [this work]
- Application 2: Error-Correction of RSA secret parameters
 - Starting from the LSB side [Henecka, May and Meurer, 2010]
 - Starting from the MSB side [this work]
- Implications of Cold-Boot attack on RSA - a summary

COLD-BOOT ATTACK

a brief introduction

Cold-Boot Attack

What happens to your computer memory when the power is down?

Cold-Boot Attack

What happens to your computer memory when the power is down?

Contrary to popular assumption, DRAMs used in most modern computers retain their contents for several seconds after power is lost, even at room temperature and even if removed from a motherboard.

- Halderman et al. [USENIX 2008, Comm. ACM 2009]

Cold-Boot Attack

What happens to your computer memory when the power is down?

Contrary to popular assumption, DRAMs used in most modern computers retain their contents for several seconds after power is lost, even at room temperature and even if removed from a motherboard.

- Halderman et al. [USENIX 2008, Comm. ACM 2009]

Pieces of the puzzle

- Fact 1: Data remanence in RAM may be prolonged by cooling
- Fact 2: The memory can be dumped/copied through cold-boot
- Fact 3: Memory may retain sensitive cryptographic information

Cold Boot Attack

Cold boot attack reads partial information from the memory!

Cold Boot Attack

Cold boot attack reads partial information from the memory!

RSA stores $N, e, p, q, d, d_p, d_q, q^{-1} \bmod p$ in memory (PKCS#1)

Potential information retrieval

- Few random bits of the secret keys $p, q, d, d_p, d_q, q^{-1} \bmod p$
- All bits of secret keys, but with some probability of error

Cold Boot Attack

Cold boot attack reads partial information from the memory!

RSA stores $N, e, p, q, d, d_p, d_q, q^{-1} \bmod p$ in memory (PKCS#1)

Potential information retrieval

- Few random bits of the secret keys $p, q, d, d_p, d_q, q^{-1} \bmod p$
- All bits of secret keys, but with some probability of error

Question: Does this partial information help the attacker?

Partial Key Exposure attacks on RSA

RIVEST AND SHAMIR (Eurocrypt 1985)

N can be factored given $2/3$ of the LSBs of a prime.

COPPERSMITH (Eurocrypt 1996)

N can be factored given $1/2$ of the MSBs of a prime.

BONEH, DURFEE AND FRANKEL (Asiacrypt 1998)

N can be factored given $1/2$ of the LSBs of a prime.

HERRMANN AND MAY (Asiacrypt 2008)

N can be factored given a random subset of the bits (small contiguous blocks) in one of the primes.

Partial Key Exposure attacks on RSA

RIVEST AND SHAMIR (Eurocrypt 1985)

N can be factored given $2/3$ of the LSBs of a prime.

COPPERSMITH (Eurocrypt 1996)

N can be factored given $1/2$ of the MSBs of a prime.

BONEH, DURFEE AND FRANKEL (Asiacrypt 1998)

N can be factored given $1/2$ of the LSBs of a prime.

HERRMANN AND MAY (Asiacrypt 2008)

N can be factored given a random subset of the bits (small contiguous blocks) in one of the primes.

What if we know random bits?

RECONSTRUCTION

of RSA Secret Parameters

Reconstruction of RSA secret parameters

SITUATION

*Cold boot attack provides you with δ **fraction of random bits** in each secret parameter p, q, d, d_p, d_q , where $0 < \delta < 1$.*

PROBLEM: Can one correctly reconstruct these parameters?

Reconstruction of RSA secret parameters

SITUATION

Cold boot attack provides you with δ fraction of random bits in each secret parameter p, q, d, d_p, d_q , where $0 < \delta < 1$.

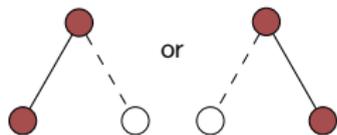
PROBLEM: Can one correctly reconstruct these parameters?

- HENINGER AND SHACHAM (Crypto 2009)
Reconstruction of secret parameters from the LSB side
- MAITRA, SARKAR AND SEN GUPTA (Africacrypt 2010)
First attempt at reconstruction from the MSB side (known blocks)
- SARKAR, SEN GUPTA AND MAITRA (this talk)
Reconstruction from the MSB side with known random bits

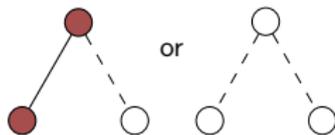
HENINGER AND SHACHAM (Crypto 2009)

- Reconstruction of parameters given δ fraction of random bits.
- Idea: The relation $p[i] \oplus q[i] = (N - p_{i-1}q_{i-1})[i]$ gives a chance for improvised branching and pruning in the search tree

Either $p[i]$ or $q[i]$ is known



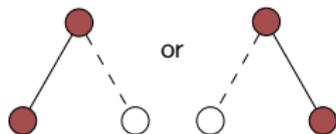
Both $p[i]$ and $q[i]$ are known



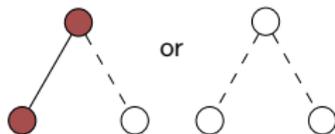
HENINGER AND SHACHAM (Crypto 2009)

- Reconstruction of parameters given δ fraction of random bits.
- Idea: The relation $p[i] \oplus q[i] = (N - p_{i-1}q_{i-1})[i]$ gives a chance for improvised branching and pruning in the search tree

Either $p[i]$ or $q[i]$ is known



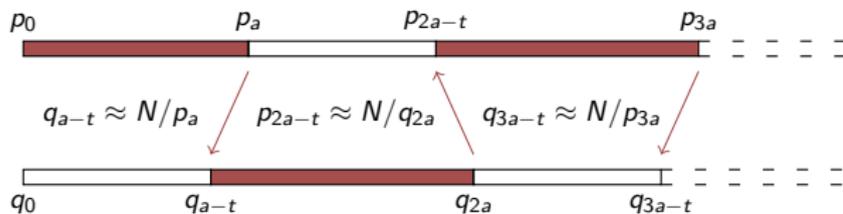
Both $p[i]$ and $q[i]$ are known



- Result: One can factor N in time $\text{poly}(e, \log_2 N)$, given
 - $\delta \geq 0.27$ fraction of random bits of p, q, d, d_p, d_q , or
 - $\delta \geq 0.42$ fraction of random bits of p, q, d , or
 - $\delta \geq 0.57$ fraction of random bits of p, q .

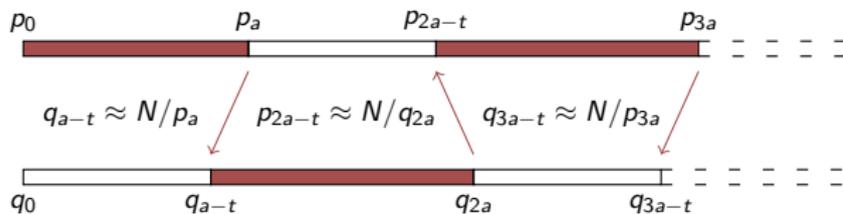
MAITRA ET AL. (Africacrypt 2010)

- Reconstruction of parameters from the MSB side given small blocks of the parameters are known.
- Intuition for primes p, q :



MAITRA ET AL. (Africacrypt 2010)

- Reconstruction of parameters from the MSB side given small blocks of the parameters are known.
- Intuition for primes p, q :



- Result: One can factor N in time $O(\log^2 N)$ with considerable probability of success given $< 70\%$ bits of the primes (together).

Random Bits: Reconstruction of p, q

CONTEXT

- We know δ fraction of random bits of both primes p, q
- The goal is to reconstruct prime p from this knowledge

Random Bits: Reconstruction of p, q

CONTEXT

- We know δ fraction of random bits of both primes p, q
- The goal is to reconstruct prime p from this knowledge

STEP 0. Guess Routine

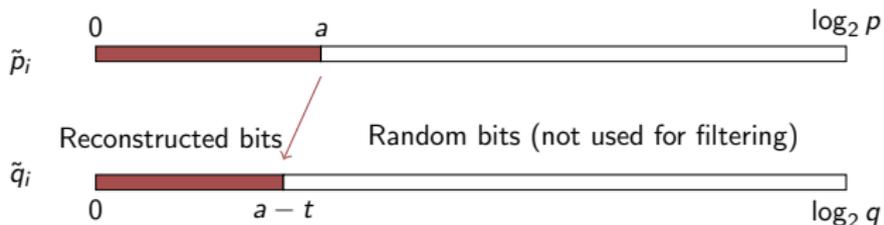
- Generate all $2^{a(1-\delta)}$ options for the first window (a MSBs) in p
- Pad the remaining by 0's, and store in an array A , say.



Random Bits: Reconstruction of p, q

STEP 1. For each option $\tilde{p}_i \in A$,

- Reconstruct first $(a - t)$ MSBs of q using $\tilde{q}_i = \lfloor \frac{N}{\tilde{p}_i} \rfloor$
- Store these options in an array B , say.
- Offset t comes as division is not 'perfect'



Random Bits: Reconstruction of p, q

STEP 2. Filter Routine

- If for some known bit $q[l]$ of q , the corresponding bit in q_i does not match, discard \tilde{q}_i from B , and hence \tilde{p}_i from A .
- If all the known bits of q match with those of \tilde{q}_i , retain \tilde{p}_i .

Filtered $A = \{\tilde{p}_1, \tilde{p}_2, \dots, \tilde{p}_x\}$ where $x = |A| < 2^{a(1-\delta)}$

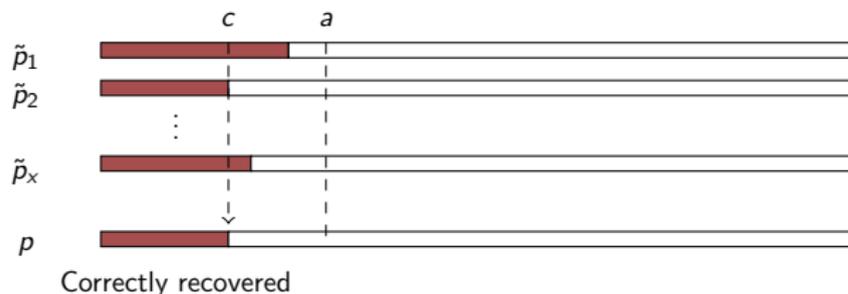
HOPE: Options in A reduce considerably after filtering.

Random Bits: Reconstruction of p, q

STEP 3.

- Each option in A has some correctly recovered block of MSBs.
- Find the initial contiguous common portion out of the options

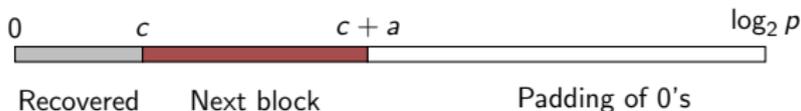
$$\tilde{p}_1[l] = \tilde{p}_2[l] = \dots = \tilde{p}_x[l] \quad \text{for all } 1 \leq l \leq c, \text{ not for } c < l \leq a$$



Random Bits: Reconstruction of p, q

ITERATE. Slide the Window

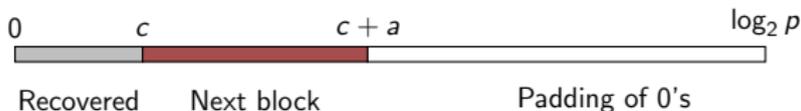
- Take next window of a bits of p starting at the $(c + 1)$ -th MSB
- Repeat Guess and Filter routines using first $(c + a)$ MSBs of p .



Random Bits: Reconstruction of p, q

ITERATE. Slide the Window

- Take next window of a bits of p starting at the $(c + 1)$ -th MSB
- Repeat Guess and Filter routines using first $(c + a)$ MSBs of p .



Continue till we get top half of prime p .

Then use Coppersmith's method to factor N efficiently!

Random Bits: Sliding Window Technique

Intuition for the General Algorithm:

1. Fit a window of length a at the top of prime p
2. Find out how many bits we know within this window
3. Guess the remaining unknown bits within the window of a bits
4. Filter through the guesses using the partial information known about the bits of all other secret parameters q, d, d_p, d_q
5. Slide the window forward and continue the same process

Experimental Results

Known	δ	Blocksize a	Offset t	Probability	Time (sec)
p, q	63	30	5	0.3	96
p, q	62	35	5	0.8	379
p, q, d	50	28	6	1.0	831
p, q, d	47	30	6	1.0	10402
p, q, d, d_p, d_q	40	25	6	0.9	2447
p, q, d, d_p, d_q	38	25	6	1.0	3861

We could factor N with considerable success probability, given

- $\delta \geq 0.38$ fraction of random bits of p, q, d, d_p, d_q , or
- $\delta \geq 0.47$ fraction of random bits of p, q, d , or
- $\delta \geq 0.62$ fraction of random bits of p, q .

Comparison with Heninger-Shacham

Heninger-Shacham: LSB side reconstruction with random bits known

Our work: MSB side reconstruction with random bits known

Bits known from	Heninger Shacham	Our result	
		Theory	Experiment
p, q	59%	64%	62%
p, q, d	42%	51%	47%
p, q, d, d_p, d_q	27%	37%	38%

Comparison with Heninger-Shacham

Heninger-Shacham: LSB side reconstruction with random bits known

Our work: MSB side reconstruction with random bits known

Bits known from	Heninger Shacham	Our result	
		Theory	Experiment
p, q	59%	64%	62%
p, q, d	42%	51%	47%
p, q, d, d_p, d_q	27%	37%	38%

How do you know the bits for sure?

ERROR CORRECTION

of RSA Secret Parameters

Error Correction of RSA Parameters

SITUATION

*Cold boot attack provides you with **all the bits** in p, q, d, d_p, d_q , but each known bit has a **certain probability** $0 < \gamma < 1$ **of being wrong**.*

PROBLEM: Can one correct the errors in these parameters?

Error Correction of RSA Parameters

SITUATION

Cold boot attack provides you with all the bits in p, q, d, d_p, d_q , but each known bit has a certain probability $0 < \gamma < 1$ of being wrong.

PROBLEM: Can one correct the errors in these parameters?

- HENECKA, MAY AND MEURER (Crypto 2010)
Correct reconstruction of secret parameters from the LSB side
- SARKAR, SEN GUPTA AND MAITRA (this talk)
Correct reconstruction of secret parameters from the MSB side

Error Correction Algorithm

CONTEXT

- We know all bits of parameters with error probability γ
- Goal is to correct this error and reconstruct prime p correctly

Error Correction Algorithm

CONTEXT

- We know all bits of parameters with error probability γ
- Goal is to correct this error and reconstruct prime p correctly

STEP 0. Guess Routine

- Generate all 2^a options for the first a MSBs of p and store in A

Error Correction Algorithm

CONTEXT

- We know all bits of parameters with error probability γ
- Goal is to correct this error and reconstruct prime p correctly

STEP 0. Guess Routine

- Generate all 2^a options for the first a MSBs of p and store in A

STEP 1.

- For each $\tilde{p}_i \in A$, reconstruct first a MSBs of all others:

$$\tilde{q}_i = \left\lfloor \frac{N}{\tilde{p}_i} \right\rfloor, \quad \tilde{d}_{p_i} = \left\lfloor \frac{k_p \tilde{p}_i}{e} \right\rfloor, \quad \tilde{d}_{q_i} = \left\lfloor \frac{k_q \tilde{q}_i}{e} \right\rfloor, \quad \tilde{d}_{2_i} = \left\lfloor \frac{k(N - \tilde{p}_i - \tilde{q}_i)}{e} \right\rfloor$$

Error Correction Algorithm

STEP 2. Filter Routine

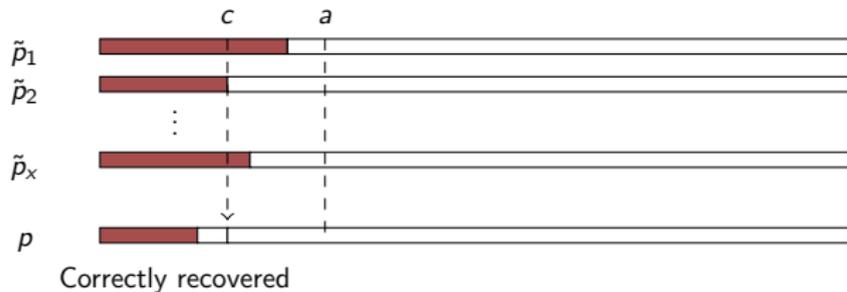
- Find the Hamming distances of $\tilde{p}_i, \tilde{q}_i, \tilde{d}_{2i}, \tilde{d}_{p_i}, \tilde{d}_{q_i}$ with the available (erroneous) values p', q', d'_2, d'_p, d'_q
- The sum of all Hamming distances is a measure for error.
 - If this sum is less than a predefined threshold, we retain \tilde{p}_i in A
 - Otherwise we discard \tilde{p}_i from A to reduce the options

HOPE: The choice of threshold is good enough to reduce options.

Error Correction Algorithm

STEP 3.

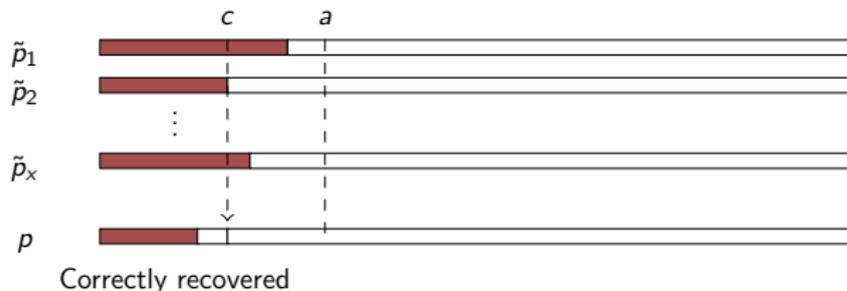
- Find initial contiguous c initial MSBs common to all options.
- Expected $(c - t)$ bits of p are correctly recovered.



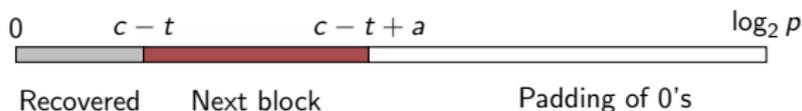
Error Correction Algorithm

STEP 3.

- Find initial contiguous c initial MSBs common to all options.
- Expected $(c - t)$ bits of p are correctly recovered.



ITERATE. Follow the 'sliding window' mechanism as before



Error Correction: Sliding Window Technique

Intuition for the General Algorithm:

1. Fit a window of length a at the top of prime p
2. Guess all bits of p within the window of a bits
3. Filter through the guesses by comparing reconstructed values and available erroneous values of all secret parameters p, q, d, d_p, d_q
4. Slide the window forward and continue the same process

Experimental Results

Erroneous approximations of p and q				
Error γ (%)	Blocksize a	Threshold	Success Prob	Time T (sec)
3	15	5	0.8	4300

Erroneous approximations of p , q and d known				
Error γ (%)	Blocksize a	Threshold	Success Prob	Time T (sec)
5	12	5	1.0	612
7	14	7	1.0	5995

Erroneous approximations of p , q , d , d_p and d_q known				
Error γ (%)	Blocksize a	Threshold	Success Prob	Time T (sec)
10	10	9	0.7	523
11	12	10	0.8	969
13	14	13	1.0	3780
15	15	14	0.8	4009

We could successfully correct errors up to 15% in practice.

Comparison with Henecka et al.

Henecka et al.: Error correction from the LSB side

Our result: Error Correction from the MSB side

Bits known from	Henecka et al		Our result	
	Theory	Experiment	Theory	Experiment
p, q	8.4%	-	8%	3%
p, q, d	16.0%	-	16%	7%
p, q, d, d_p, d_q	23.7%	17%	23%	15%

IMPLICATIONS

of cold-boot attack

Summary

RECONSTRUCTION (LSB / MSB)

One requires approximately

- 60% random bits for p, q
- 45% random bits for p, q, d
- 30% random bits for p, q, d, d_p, d_q

ERROR CORRECTION (LSB / MSB)

One can correct up to

- 8% bit-error for p, q
- 16% bit-error for p, q, d
- 23% bit-error for p, q, d, d_p, d_q

Summary

RECONSTRUCTION (LSB / MSB)

One requires approximately

- 60% random bits for p, q
- 45% random bits for p, q, d
- 30% random bits for p, q, d, d_p, d_q

ERROR CORRECTION (LSB / MSB)

One can correct up to

- 8% bit-error for p, q
- 16% bit-error for p, q, d
- 23% bit-error for p, q, d, d_p, d_q

cold-boot attack
often offers
more than this!

THANK YOU

FOR YOUR KIND ATTENTION



Learning with Errors Problem

Mahabir Prasad Jhanwar

C R RAO AIMSCS
Hyderabad Central University Campus
mahavir.jhawar@gmail.com

January 14, 2012

Outline of the talk

- 1 LWE problem
- 2 Hardness of LWE
- 3 Cryptographic Applications

Outline of the talk

- 1 LWE problem
- 2 Hardness of LWE
- 3 Cryptographic Applications

Definition

- Fix a size parameter $n \geq 1$, a modulus $q \geq 2$, and an “error” probability distribution $\chi : \mathbb{Z}_q \rightarrow \mathbb{R}^+$ on \mathbb{Z}_q .
- For a $s \in_R \mathbb{Z}_q^n$, let $A_{s,\chi}$ be a probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by
 - choosing a vector $a \in_R \mathbb{Z}_q^n$,
 - choosing $e \in_\chi \mathbb{Z}_q$
 - and outputting the pair $(a, \langle a, s \rangle + e \bmod q)$
- We say that an algorithm solves $\text{LWE}_{q,\chi}$ if,
 - for any $s \in \mathbb{Z}_q^n$,
 - given an **arbitrary** number of independent samples from $A_{s,\chi}$
 - it outputs s with high probability.

Definition

- Fix a size parameter $n \geq 1$, a modulus $q \geq 2$, and an “error” probability distribution $\chi : \mathbb{Z}_q \rightarrow \mathbb{R}^+$ on \mathbb{Z}_q .
- For a $s \in_R \mathbb{Z}_q^n$, let $A_{s,\chi}$ be a probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by
 - choosing a vector $a \in_R \mathbb{Z}_q^n$,
 - choosing $e \in_\chi \mathbb{Z}_q$
 - and outputting the pair $(a, \langle a, s \rangle + e \bmod q)$
- We say that an algorithm solves $\text{LWE}_{q,\chi}$ if,
 - for any $s \in \mathbb{Z}_q^n$,
 - given an **arbitrary** number of independent samples from $A_{s,\chi}$
 - it outputs s with high probability.

Definition

- Fix a size parameter $n \geq 1$, a modulus $q \geq 2$, and an “error” probability distribution $\chi : \mathbb{Z}_q \rightarrow \mathbb{R}^+$ on \mathbb{Z}_q .
- For a $s \in_R \mathbb{Z}_q^n$, let $A_{s,\chi}$ be a probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by
 - choosing a vector $a \in_R \mathbb{Z}_q^n$,
 - choosing $e \in_\chi \mathbb{Z}_q$
 - and outputting the pair $(a, \langle a, s \rangle + e \bmod q)$
- We say that an algorithm solves $\text{LWE}_{q,\chi}$ if,
 - for any $s \in \mathbb{Z}_q^n$,
 - given an **arbitrary** number of independent samples from $A_{s,\chi}$
 - it outputs s with high probability.

Definition

- Fix a size parameter $n \geq 1$, a modulus $q \geq 2$, and an “error” probability distribution $\chi : \mathbb{Z}_q \rightarrow \mathbb{R}^+$ on \mathbb{Z}_q .
- For a $s \in_R \mathbb{Z}_q^n$, let $A_{s,\chi}$ be a probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by
 - choosing a vector $a \in_R \mathbb{Z}_q^n$,
 - choosing $e \in_\chi \mathbb{Z}_q$
 - and outputting the pair $(a, \langle a, s \rangle + e \bmod q)$
- We say that an algorithm solves $\text{LWE}_{q,\chi}$ if,
 - for any $s \in \mathbb{Z}_q^n$,
 - given an **arbitrary** number of independent samples from $A_{s,\chi}$
 - it outputs s with high probability.

Definition

- Fix a size parameter $n \geq 1$, a modulus $q \geq 2$, and an “error” probability distribution $\chi : \mathbb{Z}_q \rightarrow \mathbb{R}^+$ on \mathbb{Z}_q .
- For a $s \in_R \mathbb{Z}_q^n$, let $A_{s,\chi}$ be a probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by
 - choosing a vector $a \in_R \mathbb{Z}_q^n$,
 - choosing $e \in_\chi \mathbb{Z}_q$
 - and outputting the pair $(a, \langle a, s \rangle + e \bmod q)$
- We say that an algorithm solves $\text{LWE}_{q,\chi}$ if,
 - for any $s \in \mathbb{Z}_q^n$,
 - given an **arbitrary** number of independent samples from $A_{s,\chi}$
 - it outputs s with high probability.

Definition

- Fix a size parameter $n \geq 1$, a modulus $q \geq 2$, and an “error” probability distribution $\chi : \mathbb{Z}_q \rightarrow \mathbb{R}^+$ on \mathbb{Z}_q .
- For a $s \in_R \mathbb{Z}_q^n$, let $A_{s,\chi}$ be a probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by
 - choosing a vector $a \in_R \mathbb{Z}_q^n$,
 - choosing $e \in_\chi \mathbb{Z}_q$
 - and outputting the pair $(a, \langle a, s \rangle + e \bmod q)$
- We say that an algorithm solves $\text{LWE}_{q,\chi}$ if,
 - for any $s \in \mathbb{Z}_q^n$,
 - given an **arbitrary** number of independent samples from $A_{s,\chi}$
 - it outputs s with high probability.

Definition

- Fix a size parameter $n \geq 1$, a modulus $q \geq 2$, and an “error” probability distribution $\chi : \mathbb{Z}_q \rightarrow \mathbb{R}^+$ on \mathbb{Z}_q .
- For a $s \in_R \mathbb{Z}_q^n$, let $A_{s,\chi}$ be a probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by
 - choosing a vector $a \in_R \mathbb{Z}_q^n$,
 - choosing $e \in_\chi \mathbb{Z}_q$
 - and outputting the pair $(a, \langle a, s \rangle + e \bmod q)$
- We say that an algorithm solves $\text{LWE}_{q,\chi}$ if,
 - for any $s \in \mathbb{Z}_q^n$,
 - given an **arbitrary** number of independent samples from $A_{s,\chi}$
 - it outputs s with high probability.

Definition

- Fix a size parameter $n \geq 1$, a modulus $q \geq 2$, and an “error” probability distribution $\chi : \mathbb{Z}_q \rightarrow \mathbb{R}^+$ on \mathbb{Z}_q .
- For a $s \in_R \mathbb{Z}_q^n$, let $A_{s,\chi}$ be a probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by
 - choosing a vector $a \in_R \mathbb{Z}_q^n$,
 - choosing $e \in_\chi \mathbb{Z}_q$
 - and outputting the pair $(a, \langle a, s \rangle + e \bmod q)$
- We say that an algorithm solves $\text{LWE}_{q,\chi}$ if,
 - for any $s \in \mathbb{Z}_q^n$,
 - given an **arbitrary** number of independent samples from $A_{s,\chi}$
 - it outputs s with high probability.

Definition

- Fix a size parameter $n \geq 1$, a modulus $q \geq 2$, and an “error” probability distribution $\chi : \mathbb{Z}_q \rightarrow \mathbb{R}^+$ on \mathbb{Z}_q .
- For a $s \in_R \mathbb{Z}_q^n$, let $A_{s,\chi}$ be a probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by
 - choosing a vector $a \in_R \mathbb{Z}_q^n$,
 - choosing $e \in_\chi \mathbb{Z}_q$
 - and outputting the pair $(a, \langle a, s \rangle + e \bmod q)$
- We say that an algorithm solves $\text{LWE}_{q,\chi}$ if,
 - for any $s \in \mathbb{Z}_q^n$,
 - given an **arbitrary** number of independent samples from $A_{s,\chi}$
 - it outputs s with high probability.

Example

- Say $n = 4$ and $q = 17$.
- Choose $s \in \mathbb{Z}_{17}^4$
- Output
 - $14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8 \pmod{17}$
 - $9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 \pmod{17}$
 - $13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 \pmod{17}$
 - \vdots

Example

- Say $n = 4$ and $q = 17$.
- Choose $s \in \mathbb{Z}_{17}^4$
- Output
 - $14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8 \pmod{17}$
 - $9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 \pmod{17}$
 - $13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 \pmod{17}$
 - \vdots

Example

- Say $n = 4$ and $q = 17$.
- Choose $\mathbf{s} \in \mathbb{Z}_{17}^4$
- Output
 - $14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8 \pmod{17}$
 - $9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 \pmod{17}$
 - $13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 \pmod{17}$
 - \vdots

Example

- Say $n = 4$ and $q = 17$.
- Choose $s \in \mathbb{Z}_{17}^4$
- Output
 - $14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8 \pmod{17}$
 - $9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 \pmod{17}$
 - $13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 \pmod{17}$
 - \vdots

Example

- Say $n = 4$ and $q = 17$.
- Choose $s \in \mathbb{Z}_{17}^4$
- Output
 - $14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8 \pmod{17}$
 - $9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 \pmod{17}$
 - $13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 \pmod{17}$
 - \vdots

Example

- Say $n = 4$ and $q = 17$.
- Choose $s \in \mathbb{Z}_{17}^4$
- Output
 - $14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8 \pmod{17}$
 - $9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 \pmod{17}$
 - $13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 \pmod{17}$
 - \vdots

Example

- Say $n = 4$ and $q = 17$.
- Choose $s \in \mathbb{Z}_{17}^4$
- Output
 - $14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8 \pmod{17}$
 - $9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 \pmod{17}$
 - $13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 \pmod{17}$
 - \vdots

Example

- Say $n = 4$ and $q = 17$.
- Choose $s \in \mathbb{Z}_{17}^4$
- Output
 - $14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8 \pmod{17}$
 - $9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 \pmod{17}$
 - $13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 \pmod{17}$
 - \vdots

Example

- Say $n = 4$ and $q = 17$.
- Choose $s \in \mathbb{Z}_{17}^4$
- Output
 - $14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8 \pmod{17}$
 - $9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 \pmod{17}$
 - $13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 \pmod{17}$
 - \vdots

LWE Parameters

- The error distribution is chosen to be
 - normal distribution rounded to the nearest integer (and reduced modulo q)
 - with standard deviation αq , where $\alpha > 0$ is taken to be $1/\text{poly}(n)$
- The modulus q is typically taken to be polynomial in n .
- The number of equations seems to be, for most purposes, independent.

LWE Parameters

- The error distribution is chosen to be
 - normal distribution rounded to the nearest integer (and reduced modulo q)
 - with standard deviation αq , where $\alpha > 0$ is taken to be $1/\text{poly}(n)$
- The modulus q is typically taken to be polynomial in n .
- The number of equations seems to be, for most purposes, independent.

LWE Parameters

- The error distribution is chosen to be
 - normal distribution rounded to the nearest integer (and reduced modulo q)
 - with standard deviation αq , where $\alpha > 0$ is taken to be $1/\text{poly}(n)$
- The modulus q is typically taken to be polynomial in n .
- The number of equations seems to be, for most purposes, independent.

LWE Parameters

- The error distribution is chosen to be
 - normal distribution rounded to the nearest integer (and reduced modulo q)
 - with standard deviation αq , where $\alpha > 0$ is taken to be $1/\text{poly}(n)$
- The modulus q is typically taken to be polynomial in n .
- The number of equations seems to be, for most purposes, independent.

LWE Parameters

- The error distribution is chosen to be
 - normal distribution rounded to the nearest integer (and reduced modulo q)
 - with standard deviation αq , where $\alpha > 0$ is taken to be $1/\text{poly}(n)$
- The modulus q is typically taken to be polynomial in n .
- The number of equations seems to be, for most purposes, independent.

LWE Parameters

- The error distribution is chosen to be
 - normal distribution rounded to the nearest integer (and reduced modulo q)
 - with standard deviation αq , where $\alpha > 0$ is taken to be $1/\text{poly}(n)$
- The modulus q is typically taken to be polynomial in n .
- The number of equations seems to be, for most purposes, independent.

Error Distribution

Solving LWE: Known Methods

- A naïve method: maximum likelihood algorithm
 - One can prove that after about $O(n)$ equations, the only assignment to s that approximately satisfies' the equations is the correct one.
 - This can be shown by a standard argument based on Chernoff's bound.
 - This leads to an algorithm that uses only $O(n)$ samples, and run in time $2^{O(n \log n)}$
 - As a corollary we obtain that LWE is **well-defined** in the sense that with high probability the solution s is unique (assuming the number of equations is $\Omega(n)$).

Solving LWE: Known Methods

- A naïve method: maximum likelihood algorithm
 - One can prove that after about $O(n)$ equations, the only assignment to s that approximately satisfies' the equations is the correct one.
 - This can be shown by a standard argument based on Chernoff's bound.
 - This leads to an algorithm that uses only $O(n)$ samples, and run in time $2^{O(n \log n)}$
 - As a corollary we obtain that LWE is **well-defined** in the sense that with high probability the solution s is unique (assuming the number of equations is $\Omega(n)$).

Solving LWE: Known Methods

- A naïve method: maximum likelihood algorithm
 - One can prove that after about $O(n)$ equations, the only assignment to s that approximately satisfies' the equations is the correct one.
 - This can be shown by a standard argument based on Chernoff's bound.
 - This leads to an algorithm that uses only $O(n)$ samples, and run in time $2^{O(n \log n)}$
 - As a corollary we obtain that LWE is **well-defined** in the sense that with high probability the solution s is unique (assuming the number of equations is $\Omega(n)$).

Solving LWE: Known Methods

- A naïve method: maximum likelihood algorithm
 - One can prove that after about $O(n)$ equations, the only assignment to s that approximately satisfies' the equations is the correct one.
 - This can be shown by a standard argument based on Chernoff's bound.
 - This leads to an algorithm that uses only $O(n)$ samples, and run in time $2^{O(n \log n)}$
 - As a corollary we obtain that LWE is **well-defined** in the sense that with high probability the solution s is unique (assuming the number of equations is $\Omega(n)$).

Solving LWE: Known Methods

- A naïve method: maximum likelihood algorithm
 - One can prove that after about $O(n)$ equations, the only assignment to s that approximately satisfies' the equations is the correct one.
 - This can be shown by a standard argument based on Chernoff's bound.
 - This leads to an algorithm that uses only $O(n)$ samples, and run in time $2^{O(n \log n)}$
 - As a corollary we obtain that LWE is **well-defined** in the sense that with high probability the solution s is unique (assuming the number of equations is $\Omega(n)$).

Solving LWE: Known Methods

- A naïve method: maximum likelihood algorithm
 - One can prove that after about $O(n)$ equations, the only assignment to s that approximately satisfies' the equations is the correct one.
 - This can be shown by a standard argument based on Chernoff's bound.
 - This leads to an algorithm that uses only $O(n)$ samples, and run in time $2^{O(n \log n)}$
 - As a corollary we obtain that LWE is **well-defined** in the sense that with high probability the solution s is unique (assuming the number of equations is $\Omega(n)$).

Solving LWE: Known Methods

- A even more naïve method
 - Keep asking for LWE samples until seeing $\text{poly}(n)$ equations of the form $s_1 \approx \dots$ (i.e., a pair (\mathbf{a}, b)) where $\mathbf{a} = (1, 0, \dots, 0)$, at which point we can recover the value of s_1 .
 - We then repeat this for all s_j .
 - The probability of seeing such an equation is $1/q^n$, leading to an algorithm requiring $2^{O(n \log n)}$ equations, and with a similar running time.
- The best known algorithm for the LWE follows from the work of Blum, Kalai, and Wasserman, and requires only $2^{O(n)}$ samples and time.

Solving LWE: Known Methods

- A even more naïve method
 - Keep asking for LWE samples until seeing $\text{poly}(n)$ equations of the form $s_1 \approx \dots$ (i.e., a pair (\mathbf{a}, b)) where $\mathbf{a} = (1, 0, \dots, 0)$, at which point we can recover the value of s_1 .
 - We then repeat this for all s_j .
 - The probability of seeing such an equation is $1/q^n$, leading to an algorithm requiring $2^{O(n \log n)}$ equations, and with a similar running time.
- The best known algorithm for the LWE follows from the work of Blum, Kalai, and Wasserman, and requires only $2^{O(n)}$ samples and time.

Solving LWE: Known Methods

- A even more naïve method
 - Keep asking for LWE samples until seeing $\text{poly}(n)$ equations of the form $s_1 \approx \dots$ (i.e., a pair (\mathbf{a}, b)) where $\mathbf{a} = (1, 0, \dots, 0)$, at which point we can recover the value of s_1 .
 - We then repeat this for all s_j .
 - The probability of seeing such an equation is $1/q^n$, leading to an algorithm requiring $2^{O(n \log n)}$ equations, and with a similar running time.
- The best known algorithm for the LWE follows from the work of Blum, Kalai, and Wasserman, and requires only $2^{O(n)}$ samples and time.

Solving LWE: Known Methods

- A even more naïve method
 - Keep asking for LWE samples until seeing $\text{poly}(n)$ equations of the form $s_1 \approx \dots$ (i.e., a pair (\mathbf{a}, b)) where $\mathbf{a} = (1, 0, \dots, 0)$, at which point we can recover the value of s_1 .
 - We then repeat this for all s_i .
 - The probability of seeing such an equation is $1/q^n$, leading to an algorithm requiring $2^{O(n \log n)}$ equations, and with a similar running time.
- The best known algorithm for the LWE follows from the work of Blum, Kalai, and Wasserman, and requires only $2^{O(n)}$ samples and time.

Solving LWE: Known Methods

- A even more naïve method
 - Keep asking for LWE samples until seeing $\text{poly}(n)$ equations of the form $s_1 \approx \dots$ (i.e., a pair (\mathbf{a}, b)) where $\mathbf{a} = (1, 0, \dots, 0)$, at which point we can recover the value of s_1 .
 - We then repeat this for all s_i .
 - The probability of seeing such an equation is $1/q^n$, leading to an algorithm requiring $2^{O(n \log n)}$ equations, and with a similar running time.
- The best known algorithm for the LWE follows from the work of Blum, Kalai, and Wasserman, and requires only $2^{O(n)}$ samples and time.

Solving LWE: Known Methods

- A even more naïve method
 - Keep asking for LWE samples until seeing $\text{poly}(n)$ equations of the form $s_1 \approx \dots$ (i.e., a pair (\mathbf{a}, b)) where $\mathbf{a} = (1, 0, \dots, 0)$, at which point we can recover the value of s_1 .
 - We then repeat this for all s_i .
 - The probability of seeing such an equation is $1/q^n$, leading to an algorithm requiring $2^{O(n \log n)}$ equations, and with a similar running time.
- The best known algorithm for the LWE follows from the work of Blum, Kalai, and Wasserman, and requires only $2^{O(n)}$ samples and time.

Outline of the talk

- 1 LWE problem
- 2 Hardness of LWE**
- 3 Cryptographic Applications

LWE Hardness: In Light of Lattice Problems

Lattices: Basic Definitions

- A lattice in \mathbb{R}^n is defined as the set of all integer combinations of n linearly independent vectors.
- This set of vectors is known as a **basis** of the lattice and is not unique
- The **dual** of a lattice Λ in \mathbb{R}^n , denoted Λ^* , is the lattice given by the set of all vectors $y \in \mathbb{R}^n$ such that $\langle x, y \rangle \in \mathbb{Z}$ for all vectors $x \in \Lambda$
- We let $\lambda_1(\Lambda)$ denote the length of the shortest nonzero vector in the lattice Λ .

Lattices: Basic Definitions

- A lattice in \mathbb{R}^n is defined as the set of all integer combinations of n linearly independent vectors.
- This set of vectors is known as a **basis** of the lattice and is not unique
- The **dual** of a lattice Λ in \mathbb{R}^n , denoted Λ^* , is the lattice given by the set of all vectors $y \in \mathbb{R}^n$ such that $\langle x, y \rangle \in \mathbb{Z}$ for all vectors $x \in \Lambda$
- We let $\lambda_1(\Lambda)$ denote the length of the shortest nonzero vector in the lattice Λ .

Lattices: Basic Definitions

- A lattice in \mathbb{R}^n is defined as the set of all integer combinations of n linearly independent vectors.
- This set of vectors is known as a **basis** of the lattice and is not unique
- The **dual** of a lattice Λ in \mathbb{R}^n , denoted Λ^* , is the lattice given by the set of all vectors $y \in \mathbb{R}^n$ such that $\langle x, y \rangle \in \mathbb{Z}$ for all vectors $x \in \Lambda$
- We let $\lambda_1(\Lambda)$ denote the length of the shortest nonzero vector in the lattice Λ .

Lattices: Basic Definitions

- A lattice in \mathbb{R}^n is defined as the set of all integer combinations of n linearly independent vectors.
- This set of vectors is known as a **basis** of the lattice and is not unique
- The **dual** of a lattice Λ in \mathbb{R}^n , denoted Λ^* , is the lattice given by the set of all vectors $y \in \mathbb{R}^n$ such that $\langle x, y \rangle \in \mathbb{Z}$ for all vectors $x \in \Lambda$
- We let $\lambda_1(\Lambda)$ denote the length of the shortest nonzero vector in the lattice Λ .

Lattices: Basic Definitions

- A lattice in \mathbb{R}^n is defined as the set of all integer combinations of n linearly independent vectors.
- This set of vectors is known as a **basis** of the lattice and is not unique
- The **dual** of a lattice Λ in \mathbb{R}^n , denoted Λ^* , is the lattice given by the set of all vectors $y \in \mathbb{R}^n$ such that $\langle x, y \rangle \in \mathbb{Z}$ for all vectors $x \in \Lambda$
- We let $\lambda_1(\Lambda)$ denote the length of the shortest nonzero vector in the lattice Λ .

Lattices: Basic Definitions

- A lattice in \mathbb{R}^n is defined as the set of all integer combinations of n linearly independent vectors.
- This set of vectors is known as a **basis** of the lattice and is not unique
- The **dual** of a lattice Λ in \mathbb{R}^n , denoted Λ^* , is the lattice given by the set of all vectors $y \in \mathbb{R}^n$ such that $\langle x, y \rangle \in \mathbb{Z}$ for all vectors $x \in \Lambda$
- We let $\lambda_1(\Lambda)$ denote the length of the shortest nonzero vector in the lattice Λ .

Lattice Problems

- **GapSVP** _{$\gamma(n)$}
 - An instance of GapSPV _{$\gamma(n)$} is given by an n -dimensional lattice Λ and a number $d > 0$. In YES instance, $\lambda_1(\Lambda) \leq d$ whereas in NO instances $\lambda_1(\Lambda) \geq \gamma(n) \times d$
- **SIVP** _{$\gamma(n)$}
 - An instance of SIVP _{$\gamma(n)$} is given by an n -dimensional lattice Λ . The goal is to output a set of n linearly independent lattice vectors of length at most $\gamma(n) \cdot \lambda_n(\Lambda)$
- **BDD**
 - For some distance parameter $d > 0$, we are given a lattice Λ and a point x within distance at most d of Λ , and asked to find the closest lattice vector to x .

Lattice Problems

- **GapSVP** _{$\gamma(n)$}
 - An instance of GapSPV _{$\gamma(n)$} is given by an n -dimensional lattice Λ and a number $d > 0$. In YES instance, $\lambda_1(\Lambda) \leq d$ whereas in NO instances $\lambda_1(\Lambda) \geq \gamma(n) \times d$
- **SIVP** _{$\gamma(n)$}
 - An instance of SIVP _{$\gamma(n)$} is given by an n -dimensional lattice Λ . The goal is to output a set of n linearly independent lattice vectors of length at most $\gamma(n) \cdot \lambda_n(\Lambda)$
- **BDD**
 - For some distance parameter $d > 0$, we are given a lattice Λ and a point x within distance at most d of Λ , and asked to find the closest lattice vector to x .

Lattice Problems

- **GapSVP** _{$\gamma(n)$}
 - An instance of GapSPV _{$\gamma(n)$} is given by an n -dimensional lattice Λ and a number $d > 0$. In YES instance, $\lambda_1(\Lambda) \leq d$ whereas in NO instances $\lambda_1(\Lambda) \geq \gamma(n) \times d$
- **SIVP** _{$\gamma(n)$}
 - An instance of SIVP _{$\gamma(n)$} is given by an n -dimensional lattice Λ . The goal is to output a set of n linearly independent lattice vectors of length at most $\gamma(n) \cdot \lambda_n(\Lambda)$
- **BDD**
 - For some distance parameter $d > 0$, we are given a lattice Λ and a point x within distance at most d of Λ , and asked to find the closest lattice vector to x .

Lattice Problems

- **GapSVP** _{$\gamma(n)$}
 - An instance of GapSPV _{$\gamma(n)$} is given by an n -dimensional lattice Λ and a number $d > 0$. In YES instance, $\lambda_1(\Lambda) \leq d$ whereas in NO instances $\lambda_1(\Lambda) \geq \gamma(n) \times d$
- **SIVP** _{$\gamma(n)$}
 - An instance of SIVP _{$\gamma(n)$} is given by an n -dimensional lattice Λ . The goal is to output a set of n linearly independent lattice vectors of length at most $\gamma(n) \cdot \lambda_n(\Lambda)$
- **BDD**
 - For some distance parameter $d > 0$, we are given a lattice Λ and a point x within distance at most d of Λ , and asked to find the closest lattice vector to x .

Lattice Problems

- **GapSVP** _{$\gamma(n)$}
 - An instance of GapSPV _{$\gamma(n)$} is given by an n -dimensional lattice Λ and a number $d > 0$. In YES instance, $\lambda_1(\Lambda) \leq d$ whereas in NO instances $\lambda_1(\Lambda) \geq \gamma(n) \times d$
- **SIVP** _{$\gamma(n)$}
 - An instance of SIVP _{$\gamma(n)$} is given by an n -dimensional lattice Λ . The goal is to output a set of n linearly independent lattice vectors of length at most $\gamma(n) \cdot \lambda_n(\Lambda)$
- **BDD**
 - For some distance parameter $d > 0$, we are given a lattice Λ and a point x within distance at most d of Λ , and asked to find the closest lattice vector to x .

Lattice Problems

- **GapSVP** _{$\gamma(n)$}
 - An instance of GapSPV _{$\gamma(n)$} is given by an n -dimensional lattice Λ and a number $d > 0$. In YES instance, $\lambda_1(\Lambda) \leq d$ whereas in NO instances $\lambda_1(\Lambda) \geq \gamma(n) \times d$
- **SIVP** _{$\gamma(n)$}
 - An instance of SIVP _{$\gamma(n)$} is given by an n -dimensional lattice Λ . The goal is to output a set of n linearly independent lattice vectors of length at most $\gamma(n) \cdot \lambda_n(\Lambda)$
- **BDD**
 - For some distance parameter $d > 0$, we are given a lattice Λ and a point x within distance at most d of Λ , and asked to find the closest lattice vector to x .

Lattice Problems

- **GapSVP** _{$\gamma(n)$}
 - An instance of GapSPV _{$\gamma(n)$} is given by an n -dimensional lattice Λ and a number $d > 0$. In YES instance, $\lambda_1(\Lambda) \leq d$ whereas in NO instances $\lambda_1(\Lambda) \geq \gamma(n) \times d$
- **SIVP** _{$\gamma(n)$}
 - An instance of SIVP _{$\gamma(n)$} is given by an n -dimensional lattice Λ . The goal is to output a set of n linearly independent lattice vectors of length at most $\gamma(n) \cdot \lambda_n(\Lambda)$
- **BDD**
 - For some distance parameter $d > 0$, we are given a lattice Λ and a point x within distance at most d of Λ , and asked to find the closest lattice vector to x .

Discrete Gaussian Distribution on Λ

- We will make good use of the discrete Gaussian distribution on Λ of width r , denoted $D_{\Lambda,r}$
- this distribution has support Λ and in which the probability of each $x \in \Lambda$ is proportional to $e^{-\pi\|x/r\|^2}$.
- **smoothing parameter** $\eta_\epsilon(\Lambda)$: Roughly speaking, it gives the smallest r starting from which $D_{\Lambda,r}$ behaves like a continuous Gaussian distribution.
- For instance, for $r \geq \eta_\epsilon(\Lambda)$, vectors chosen from $D_{\Lambda,r}$ have norm roughly $r\sqrt{n}$ with high probability.

Discrete Gaussian Distribution on Λ

- We will make good use of the discrete Gaussian distribution on Λ of width r , denoted $D_{\Lambda,r}$
- this distribution has support Λ and in which the probability of each $x \in \Lambda$ is proportional to $e^{-\pi\|x/r\|^2}$.
- **smoothing parameter** $\eta_\epsilon(\Lambda)$: Roughly speaking, it gives the smallest r starting from which $D_{\Lambda,r}$ behaves like a continuous Gaussian distribution.
- For instance, for $r \geq \eta_\epsilon(\Lambda)$, vectors chosen from $D_{\Lambda,r}$ have norm roughly $r\sqrt{n}$ with high probability.

Discrete Gaussian Distribution on Λ

- We will make good use of the discrete Gaussian distribution on Λ of width r , denoted $D_{\Lambda,r}$
- this distribution has support Λ and in which the probability of each $x \in \Lambda$ is proportional to $e^{-\pi\|x/r\|^2}$.
- **smoothing parameter** $\eta_\epsilon(\Lambda)$: Roughly speaking, it gives the smallest r starting from which $D_{\Lambda,r}$ behaves like a continuous Gaussian distribution.
- For instance, for $r \geq \eta_\epsilon(\Lambda)$, vectors chosen from $D_{\Lambda,r}$ have norm roughly $r\sqrt{n}$ with high probability.

Discrete Gaussian Distribution on Λ

- We will make good use of the discrete Gaussian distribution on Λ of width r , denoted $D_{\Lambda,r}$
- this distribution has support Λ and in which the probability of each $x \in \Lambda$ is proportional to $e^{-\pi\|x/r\|^2}$.
- **smoothing parameter** $\eta_\epsilon(\Lambda)$: Roughly speaking, it gives the smallest r starting from which $D_{\Lambda,r}$ behaves like a continuous Gaussian distribution.
- For instance, for $r \geq \eta_\epsilon(\Lambda)$, vectors chosen from $D_{\Lambda,r}$ have norm roughly $r\sqrt{n}$ with high probability.

Discrete Gaussian Distribution on Λ

- We will make good use of the discrete Gaussian distribution on Λ of width r , denoted $D_{\Lambda,r}$
- this distribution has support Λ and in which the probability of each $x \in \Lambda$ is proportional to $e^{-\pi\|x/r\|^2}$.
- **smoothing parameter** $\eta_\epsilon(\Lambda)$: Roughly speaking, it gives the smallest r starting from which $D_{\Lambda,r}$ behaves like a continuous Gaussian distribution.
- For instance, for $r \geq \eta_\epsilon(\Lambda)$, vectors chosen from $D_{\Lambda,r}$ have norm roughly $r\sqrt{n}$ with high probability.

Discrete Gaussian Distribution on Λ

- We will make good use of the discrete Gaussian distribution on Λ of width r , denoted $D_{\Lambda,r}$
- this distribution has support Λ and in which the probability of each $x \in \Lambda$ is proportional to $e^{-\pi\|x/r\|^2}$.
- **smoothing parameter** $\eta_\epsilon(\Lambda)$: Roughly speaking, it gives the smallest r starting from which $D_{\Lambda,r}$ behaves like a continuous Gaussian distribution.
- For instance, for $r \geq \eta_\epsilon(\Lambda)$, vectors chosen from $D_{\Lambda,r}$ have norm roughly $r\sqrt{n}$ with high probability.

Discret Gaussian Sampling Problem (DGS)

- **DGS:** Given an n -dimensional lattice Λ and a number $r \geq \sqrt{2n} \cdot \eta_\epsilon(\Lambda) / \alpha$, output a sample from $D_{\Lambda, r}$
- GapSVP $\xrightarrow{\text{QuantumReduction}}$ LWE
- DGS $\xrightarrow{\text{QuantumReduction}}$ LWE
- GapSVP \rightarrow DGS

Discret Gaussian Sampling Problem (DGS)

- **DGS:** Given an n -dimensional lattice Λ and a number $r \geq \sqrt{2n} \cdot \eta_\epsilon(\Lambda) / \alpha$, output a sample from $D_{\Lambda,r}$
- GapSVP $\xrightarrow{\text{QuantumReduction}}$ LWE
- DGS $\xrightarrow{\text{QuantumReduction}}$ LWE
- GapSVP \rightarrow DGS

Discret Gaussian Sampling Problem (DGS)

- **DGS:** Given an n -dimensional lattice Λ and a number $r \geq \sqrt{2n} \cdot \eta_\epsilon(\Lambda) / \alpha$, output a sample from $D_{\Lambda, r}$
- GapSVP $\xrightarrow{\text{QuantumReduction}}$ LWE
- DGS $\xrightarrow{\text{QuantumReduction}}$ LWE
- GapSVP \rightarrow DGS

Discret Gaussian Sampling Problem (DGS)

- **DGS:** Given an n -dimensional lattice Λ and a number $r \geq \sqrt{2n} \cdot \eta_\epsilon(\Lambda) / \alpha$, output a sample from $D_{\Lambda,r}$
- GapSVP $\xrightarrow{\text{QuantumReduction}}$ LWE
- DGS $\xrightarrow{\text{QuantumReduction}}$ LWE
- GapSVP \rightarrow DGS

Discret Gaussian Sampling Problem (DGS)

- **DGS:** Given an n -dimensional lattice Λ and a number $r \geq \sqrt{2n} \cdot \eta_\epsilon(\Lambda) / \alpha$, output a sample from $D_{\Lambda, r}$
- GapSVP $\xrightarrow{\text{QuantumReduction}}$ LWE
- DGS $\xrightarrow{\text{QuantumReduction}}$ LWE
- GapSVP \longrightarrow DGS

LWE Hardness: In Light of Lattice Problems

Theorem.

- Let $q \geq 2$ be an integer and α be a real number in $(0, 1)$.
- Assume we are given access to an oracle that solves the LWE problem with modulus q and error distribution α .
- Then, given as input any lattice Λ , a large enough polynomial number of samples from the discrete Gaussian distribution $D_{\Lambda^*, r}$ for some r ($r \geq q\sqrt{2n}/\lambda_1(\Lambda)$), and a point x within distance $\alpha q/\sqrt{2}r$ of Λ ,
- we can output the (unique) closest lattice point to x in polynomial time.

LWE Hardness: In Light of Lattice Problems

Theorem.

- Let $q \geq 2$ be an integer and α be a real number in $(0, 1)$.
- Assume we are given access to an oracle that solves the LWE problem with modulus q and error distribution α .
- Then, given as input any lattice Λ , a large enough polynomial number of samples from the discrete Gaussian distribution $D_{\Lambda^*, r}$ for some r ($r \geq q\sqrt{2n}/\lambda_1(\Lambda)$), and a point x within distance $\alpha q/\sqrt{2}r$ of Λ ,
- we can output the (unique) closest lattice point to x in polynomial time.

LWE Hardness: In Light of Lattice Problems

Theorem.

- Let $q \geq 2$ be an integer and α be a real number in $(0, 1)$.
- Assume we are given access to an oracle that solves the LWE problem with modulus q and error distribution α .
- Then, given as input any lattice Λ , a large enough polynomial number of samples from the discrete Gaussian distribution $D_{\Lambda^*, r}$ for some r ($r \geq q\sqrt{2n}/\lambda_1(\Lambda)$), and a point x within distance $\alpha q/\sqrt{2}r$ of Λ ,
- we can output the (unique) closest lattice point to x in polynomial time.

LWE Hardness: In Light of Lattice Problems

Theorem.

- Let $q \geq 2$ be an integer and α be a real number in $(0, 1)$.
- Assume we are given access to an oracle that solves the LWE problem with modulus q and error distribution α .
- Then, given as input any lattice Λ , a large enough polynomial number of samples from the discrete Gaussian distribution $D_{\Lambda^*, r}$ for some r ($r \geq q\sqrt{2n}/\lambda_1(\Lambda)$), and a point x within distance $\alpha q/\sqrt{2r}$ of Λ ,
- we can output the (unique) closest lattice point to x in polynomial time.

LWE Hardness: In Light of Lattice Problems

Theorem.

- Let $q \geq 2$ be an integer and α be a real number in $(0, 1)$.
- Assume we are given access to an oracle that solves the LWE problem with modulus q and error distribution α .
- Then, given as input any lattice Λ , a large enough polynomial number of samples from the discrete Gaussian distribution $D_{\Lambda^*, r}$ for some r ($r \geq q\sqrt{2n}/\lambda_1(\Lambda)$), and a point x within distance $\alpha q/\sqrt{2r}$ of Λ ,
- we can output the (unique) closest lattice point to x in polynomial time.

LWE Hardness: In Light of Lattice Problems

- In order to understand the significance of this Theorem, it is useful to contrast it with the following result:
 - Given as input a lattice Λ , a large enough polynomial number of samples from the discrete Gaussian distribution $D_{\Lambda^*, r}$ for some (not too small) r , and a point x within distance $O(\sqrt{\log n}/r)$ of Λ , we can output the (unique) closest lattice point to x in **polynomial time**.
- Theorem shows: using an LWE oracle, the decoding radius can be increased from $(\sqrt{\log n}/r)$ to $\alpha q/\sqrt{2}r$
- Indication for LWE hardness: as it allows us to solve a worst-case lattice problem (BDD given a hint in the form of samples from the discrete Gaussian distribution) that we do not know how to solve otherwise
- More specifically, for $\alpha q = \sqrt{n}$, the best algorithms we have to solve the problem require exponential time.

LWE Hardness: In Light of Lattice Problems

- In order to understand the significance of this Theorem, it is useful to contrast it with the following result:
 - Given as input a lattice Λ , a large enough polynomial number of samples from the discrete Gaussian distribution $D_{\Lambda^*, r}$ for some (not too small) r , and a point x within distance $O(\sqrt{\log n}/r)$ of Λ , we can output the (unique) closest lattice point to x in **polynomial time**.
- Theorem shows: using an LWE oracle, the decoding radius can be increased from $(\sqrt{\log n}/r)$ to $\alpha q/\sqrt{2}r$
- Indication for LWE hardness: as it allows us to solve a worst-case lattice problem (BDD given a hint in the form of samples from the discrete Gaussian distribution) that we do not know how to solve otherwise
- More specifically, for $\alpha q = \sqrt{n}$, the best algorithms we have to solve the problem require exponential time.

LWE Hardness: In Light of Lattice Problems

- In order to understand the significance of this Theorem, it is useful to contrast it with the following result:
 - Given as input a lattice Λ , a large enough polynomial number of samples from the discrete Gaussian distribution $D_{\Lambda^*, r}$ for some (not too small) r , and a point x within distance $O(\sqrt{\log n}/r)$ of Λ , we can output the (unique) closest lattice point to x in **polynomial time**.
- Theorem shows: using an LWE oracle, the decoding radius can be increased from $(\sqrt{\log n}/r)$ to $\alpha q/\sqrt{2}r$
- Indication for LWE hardness: as it allows us to solve a worst-case lattice problem (BDD given a hint in the form of samples from the discrete Gaussian distribution) that we do not know how to solve otherwise
- More specifically, for $\alpha q = \sqrt{n}$, the best algorithms we have to solve the problem require exponential time.

LWE Hardness: In Light of Lattice Problems

- In order to understand the significance of this Theorem, it is useful to contrast it with the following result:
 - Given as input a lattice Λ , a large enough polynomial number of samples from the discrete Gaussian distribution $D_{\Lambda^*, r}$ for some (not too small) r , and a point x within distance $O(\sqrt{\log n}/r)$ of Λ , we can output the (unique) closest lattice point to x in **polynomial time**.
- Theorem shows: using an LWE oracle, the decoding radius can be increased from $(\sqrt{\log n}/r)$ to $\alpha q/\sqrt{2}r$
- Indication for LWE hardness: as it allows us to solve a worst-case lattice problem (BDD given a hint in the form of samples from the discrete Gaussian distribution) that we do not know how to solve otherwise
- More specifically, for $\alpha q = \sqrt{n}$, the best algorithms we have to solve the problem require exponential time.

LWE Hardness: In Light of Lattice Problems

- In order to understand the significance of this Theorem, it is useful to contrast it with the following result:
 - Given as input a lattice Λ , a large enough polynomial number of samples from the discrete Gaussian distribution $D_{\Lambda^*, r}$ for some (not too small) r , and a point x within distance $O(\sqrt{\log n}/r)$ of Λ , we can output the (unique) closest lattice point to x in **polynomial time**.
- Theorem shows: using an LWE oracle, the decoding radius can be increased from $(\sqrt{\log n}/r)$ to $\alpha q/\sqrt{2}r$
- Indication for LWE hardness: as it allows us to solve a worst-case lattice problem (BDD given a hint in the form of samples from the discrete Gaussian distribution) that we do not know how to solve otherwise
- More specifically, for $\alpha q = \sqrt{n}$, the best algorithms we have to solve the problem require exponential time.

LWE Hardness: In Light of Lattice Problems

- In order to understand the significance of this Theorem, it is useful to contrast it with the following result:
 - Given as input a lattice Λ , a large enough polynomial number of samples from the discrete Gaussian distribution $D_{\Lambda^*, r}$ for some (not too small) r , and a point x within distance $O(\sqrt{\log n}/r)$ of Λ , we can output the (unique) closest lattice point to x in **polynomial time**.
- Theorem shows: using an LWE oracle, the decoding radius can be increased from $(\sqrt{\log n}/r)$ to $\alpha q/\sqrt{2}r$
- Indication for LWE hardness: as it allows us to solve a worst-case lattice problem (BDD given a hint in the form of samples from the discrete Gaussian distribution) that we do not know how to solve otherwise
- More specifically, for $\alpha q = \sqrt{n}$, the best algorithms we have to solve the problem require exponential time.

LWE Hardness: In Light of Lattice Problems

- We now relate this problem to more standard lattice problems.
- There is a polynomial time reduction from the standard lattice problem GapSVP (with a $\text{poly}(n)$ approximation factor) to BDD (to within distance $\lambda_1/\text{poly}(n)$)
- This is quite reassuring, as it tells us that as long as $\alpha q/r = \lambda_1(\Lambda)/\text{poly}(n)$, the LWE problem is as hard as the variant of the worst-case lattice problem GapSVP in which we are given samples from $D_{\Lambda^*,r}$

LWE Hardness: In Light of Lattice Problems

- We now relate this problem to more standard lattice problems.
- There is a polynomial time reduction from the standard lattice problem GapSVP (with a $\text{poly}(n)$ approximation factor) to BDD (to within distance $\lambda_1/\text{poly}(n)$)
- This is quite reassuring, as it tells us that as long as $\alpha q/r = \lambda_1(\Lambda)/\text{poly}(n)$, the LWE problem is as hard as the variant of the worst-case lattice problem GapSVP in which we are given samples from $D_{\Lambda^*,r}$

LWE Hardness: In Light of Lattice Problems

- We now relate this problem to more standard lattice problems.
- There is a polynomial time reduction from the standard lattice problem GapSVP (with a $\text{poly}(n)$ approximation factor) to BDD (to within distance $\lambda_1/\text{poly}(n)$)
- This is quite reassuring, as it tells us that as long as $\alpha q/r = \lambda_1(\Lambda)/\text{poly}(n)$, the LWE problem is as hard as the variant of the worst-case lattice problem GapSVP in which we are given samples from $D_{\Lambda^*,r}$

LWE Hardness: In Light of Lattice Problems

- We now relate this problem to more standard lattice problems.
- There is a polynomial time reduction from the standard lattice problem GapSVP (with a $\text{poly}(n)$ approximation factor) to BDD (to within distance $\lambda_1/\text{poly}(n)$)
- This is quite reassuring, as it tells us that as long as $\alpha q/r = \lambda_1(\Lambda)/\text{poly}(n)$, the LWE problem is as hard as the variant of the worst-case lattice problem GapSVP in which we are given samples from $D_{\Lambda^*,r}$

LWE Hardness: In Light of Lattice Problems

- It would be even nicer if we could replace the somewhat unusual assumption regarding the discrete Gaussian samples with a more familiar one.
- This can be done using certain sampling which roughly speaking, is able to efficiently produce such samples given a basis of Λ^* all of whose vectors are of length at most r
- This leads to a hardness result for LWE based on the assumption that GapSVP is hard even given an unusually good basis for it.
- Alternatively, using the LLL algorithm we can efficiently produce a basis of Λ^* whose vectors are of length at most $2^n/\lambda_1(\Lambda)$
- this implies that LWE for **exponential moduli** $q = 2^{O(n)}$ is as hard as the standard worst-case lattice problem GapSVP.

LWE Hardness: In Light of Lattice Problems

- It would be even nicer if we could replace the somewhat unusual assumption regarding the discrete Gaussian samples with a more familiar one.
- This can be done using certain sampling which roughly speaking, is able to efficiently produce such samples given a basis of Λ^* all of whose vectors are of length at most r
- This leads to a hardness result for LWE based on the assumption that GapSVP is hard even given an unusually good basis for it.
- Alternatively, using the LLL algorithm we can efficiently produce a basis of Λ^* whose vectors are of length at most $2^n/\lambda_1(\Lambda)$
- this implies that LWE for **exponential moduli** $q = 2^{O(n)}$ is as hard as the standard worst-case lattice problem GapSVP.

LWE Hardness: In Light of Lattice Problems

- It would be even nicer if we could replace the somewhat unusual assumption regarding the discrete Gaussian samples with a more familiar one.
- This can be done using certain sampling which roughly speaking, is able to efficiently produce such samples given a basis of Λ^* all of whose vectors are of length at most r
- This leads to a hardness result for LWE based on the assumption that GapSVP is hard even given an unusually good basis for it.
- Alternatively, using the LLL algorithm we can efficiently produce a basis of Λ^* whose vectors are of length at most $2^n/\lambda_1(\Lambda)$
- this implies that LWE for **exponential moduli** $q = 2^{O(n)}$ is as hard as the standard worst-case lattice problem GapSVP.

LWE Hardness: In Light of Lattice Problems

- It would be even nicer if we could replace the somewhat unusual assumption regarding the discrete Gaussian samples with a more familiar one.
- This can be done using certain sampling which roughly speaking, is able to efficiently produce such samples given a basis of Λ^* all of whose vectors are of length at most r
- This leads to a hardness result for LWE based on the assumption that GapSVP is hard even given an unusually good basis for it.
- Alternatively, using the LLL algorithm we can efficiently produce a basis of Λ^* whose vectors are of length at most $2^n/\lambda_1(\Lambda)$
- this implies that LWE for **exponential moduli** $q = 2^{O(n)}$ is as hard as the standard worst-case lattice problem GapSVP.

LWE Hardness: In Light of Lattice Problems

- It would be even nicer if we could replace the somewhat unusual assumption regarding the discrete Gaussian samples with a more familiar one.
- This can be done using certain sampling which roughly speaking, is able to efficiently produce such samples given a basis of Λ^* all of whose vectors are of length at most r
- This leads to a hardness result for LWE based on the assumption that GapSVP is hard even given an unusually good basis for it.
- Alternatively, using the LLL algorithm we can efficiently produce a basis of Λ^* whose vectors are of length at most $2^n/\lambda_1(\Lambda)$
- this implies that LWE for **exponential moduli** $q = 2^{O(n)}$ is as hard as the standard worst-case lattice problem GapSVP.

LWE Hardness: In Light of Lattice Problems

- It would be even nicer if we could replace the somewhat unusual assumption regarding the discrete Gaussian samples with a more familiar one.
- This can be done using certain sampling which roughly speaking, is able to efficiently produce such samples given a basis of Λ^* all of whose vectors are of length at most r
- This leads to a hardness result for LWE based on the assumption that GapSVP is hard even given an unusually good basis for it.
- Alternatively, using the LLL algorithm we can efficiently produce a basis of Λ^* whose vectors are of length at most $2^n/\lambda_1(\Lambda)$
- this implies that LWE for **exponential moduli** $q = 2^{O(n)}$ is as hard as the standard worst-case lattice problem GapSVP.

LWE Hardness: In Light of Lattice Problems

Proof Outline

- We demonstrate the main idea of the proof with the lattice $\Lambda = \mathbb{Z}^n$
- The use of \mathbb{Z}^n is just in order to clarify the main ideas (BDD is trivial on \mathbb{Z}^n !!).
- Given a point x close to some unknown lattice vector $v \in \mathbb{Z}^n$.
- We will show below how to generate samples from the LWE distribution with secret $s = v \pmod{q}$

LWE Hardness: In Light of Lattice Problems

Proof Outline

- We demonstrate the main idea of the proof with the lattice $\Lambda = \mathbb{Z}^n$
- The use of \mathbb{Z}^n is just in order to clarify the main ideas (BDD is trivial on \mathbb{Z}^n !!).
- Given a point x close to some unknown lattice vector $v \in \mathbb{Z}^n$.
- We will show below how to generate samples from the LWE distribution with secret $s = v \pmod{q}$

LWE Hardness: In Light of Lattice Problems

Proof Outline

- We demonstrate the main idea of the proof with the lattice $\Lambda = \mathbb{Z}^n$
- The use of \mathbb{Z}^n is just in order to clarify the main ideas (BDD is trivial on \mathbb{Z}^n !!).
- Given a point x close to some unknown lattice vector $v \in \mathbb{Z}^n$.
- We will show below how to generate samples from the LWE distribution with secret $s = v \pmod{q}$

LWE Hardness: In Light of Lattice Problems

Proof Outline

- We demonstrate the main idea of the proof with the lattice $\Lambda = \mathbb{Z}^n$
- The use of \mathbb{Z}^n is just in order to clarify the main ideas (BDD is trivial on \mathbb{Z}^n !!).
- Given a point x close to some unknown lattice vector $v \in \mathbb{Z}^n$.
- We will show below how to generate samples from the LWE distribution with secret $s = v \pmod{q}$

LWE Hardness: In Light of Lattice Problems

Proof Outline

- We demonstrate the main idea of the proof with the lattice $\Lambda = \mathbb{Z}^n$
- The use of \mathbb{Z}^n is just in order to clarify the main ideas (BDD is trivial on \mathbb{Z}^n !!).
- Given a point x close to some unknown lattice vector $v \in \mathbb{Z}^n$.
- We will show below how to generate samples from the LWE distribution with secret $s = v \pmod{q}$

Proof Outline

- Using the LWE oracle, we can recover $s \pmod{q}$ i.e., the least significant digits of v in the base q
- Note that the vector $(x - s)/q$ is close to the lattice vector $(v - s)/q \in \mathbb{Z}^n$
- run the same process on $(x - s)/q$ to recover the second digits of v in base q , and so on..
- Thus, the core of the proof, is therefore, in producing LWE samples with secret s .

Proof Outline

- Using the LWE oracle, we can recover $s \pmod{q}$ i.e., the least significant digits of v in the base q
- Note that the vector $(x - s)/q$ is close to the lattice vector $(v - s)/q \in \mathbb{Z}^n$
- run the same process on $(x - s)/q$ to recover the second digits of v in base q , and so on..
- Thus, the core of the proof, is therefore, in producing LWE samples with secret s .

Proof Outline

- Using the LWE oracle, we can recover $s \pmod{q}$ i.e., the least significant digits of v in the base q
- Note that the vector $(x - s)/q$ is close to the lattice vector $(v - s)/q \in \mathbb{Z}^n$
- run the same process on $(x - s)/q$ to recover the second digits of v in base q , and so on..
- Thus, the core of the proof, is therefore, in producing LWE samples with secret s .

Proof Outline

- Using the LWE oracle, we can recover $s \pmod{q}$ i.e., the least significant digits of v in the base q
- Note that the vector $(x - s)/q$ is close to the lattice vector $(v - s)/q \in \mathbb{Z}^n$
- run the same process on $(x - s)/q$ to recover the second digits of v in base q , and so on..
- Thus, the core of the proof, is therefore, in producing LWE samples with secret s .

Proof Outline

- Take a sample y from $D_{\mathbb{Z}^n, r}$ (using the samples given to us as input)
- Output the pair

$$(a = y \bmod q, b = \lfloor \langle y, x \rangle \rfloor \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}$$

- Since r is not too small, the distribution of a is essentially uniform over \mathbb{Z}_q^n

Proof Outline

- Take a sample y from $D_{\mathbb{Z}^n, r}$ (using the samples given to us as input)
- Output the pair

$$(a = y \bmod q, b = \lfloor \langle y, x \rangle \rfloor \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}$$

- Since r is not too small, the distribution of a is essentially uniform over \mathbb{Z}_q^n

Proof Outline

- Take a sample y from $D_{\mathbb{Z}^n, r}$ (using the samples given to us as input)
- Output the pair

$$(a = y \bmod q, b = \lfloor \langle y, x \rangle \rfloor \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}$$

- Since r is not too small, the distribution of a is essentially uniform over \mathbb{Z}_q^n

Proof Outline

- Take a sample y from $D_{\mathbb{Z}^n, r}$ (using the samples given to us as input)
- Output the pair

$$(a = y \bmod q, b = \lfloor \langle y, x \rangle \rfloor \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}$$

- Since r is not too small, the distribution of a is essentially uniform over \mathbb{Z}_q^n

Proof Outline

- We now analyze the distribution of the second coordinate
- We have $x = v + e$ for some vector e of norm at most $\alpha q / \sqrt{2}r$
- Thus $b = \lfloor \langle y, x \rangle \rfloor \bmod q = \lfloor \langle y, v \rangle + \langle e, y \rangle \rfloor \bmod q$
- Thus we obtain an error term in the second coordinate of the form $\lfloor e, y \rfloor$
- $|\langle e, y \rangle| \leq \|e\| \|y\| \leq \alpha q / \sqrt{2}r \times r \approx \alpha q$.
- Being the inner product of a fixed vector with a discrete Gaussian vector, this error term is essentially normally distributed with standard deviation at most roughly αq , as required.

Proof Outline

- We now analyze the distribution of the second coordinate
- We have $x = v + e$ for some vector e of norm at most $\alpha q / \sqrt{2}r$
- Thus $b = \lfloor \langle y, x \rangle \rfloor \bmod q = \lfloor \langle y, v \rangle + \langle e, y \rangle \rfloor \bmod q$
- Thus we obtain an error term in the second coordinate of the form $\lfloor e, y \rfloor$
- $|\langle e, y \rangle| \leq \|e\| \|y\| \leq \alpha q / \sqrt{2}r \times r \approx \alpha q$.
- Being the inner product of a fixed vector with a discrete Gaussian vector, this error term is essentially normally distributed with standard deviation at most roughly αq , as required.

Proof Outline

- We now analyze the distribution of the second coordinate
- We have $x = v + e$ for some vector e of norm at most $\alpha q / \sqrt{2}r$
- Thus $b = \lfloor \langle y, x \rangle \rfloor \bmod q = \lfloor \langle y, v \rangle + \langle e, y \rangle \rfloor \bmod q$
- Thus we obtain an error term in the second coordinate of the form $\lfloor e, y \rfloor$
- $|\langle e, y \rangle| \leq \|e\| \|y\| \leq \alpha q / \sqrt{2}r \times r \approx \alpha q$.
- Being the inner product of a fixed vector with a discrete Gaussian vector, this error term is essentially normally distributed with standard deviation at most roughly αq , as required.

Proof Outline

- We now analyze the distribution of the second coordinate
- We have $x = v + e$ for some vector e of norm at most $\alpha q / \sqrt{2}r$
- Thus $b = \lfloor \langle y, x \rangle \rfloor \bmod q = \lfloor \langle y, v \rangle + \langle e, y \rangle \rfloor \bmod q$
- Thus we obtain an error term in the second coordinate of the form $\lfloor e, y \rfloor$
- $|\langle e, y \rangle| \leq \|e\| \|y\| \leq \alpha q / \sqrt{2}r \times r \approx \alpha q$.
- Being the inner product of a fixed vector with a discrete Gaussian vector, this error term is essentially normally distributed with standard deviation at most roughly αq , as required.

Proof Outline

- We now analyze the distribution of the second coordinate
- We have $x = v + e$ for some vector e of norm at most $\alpha q / \sqrt{2}r$
- Thus $b = \lfloor \langle y, x \rangle \rfloor \bmod q = \lfloor \langle y, v \rangle + \langle e, y \rangle \rfloor \bmod q$
- Thus we obtain an error term in the second coordinate of the form $\lfloor e, y \rfloor$
- $|\langle e, y \rangle| \leq \|e\| \|y\| \leq \alpha q / \sqrt{2}r \times r \approx \alpha q$.
- Being the inner product of a fixed vector with a discrete Gaussian vector, this error term is essentially normally distributed with standard deviation at most roughly αq , as required.

Proof Outline

- We now analyze the distribution of the second coordinate
- We have $x = v + e$ for some vector e of norm at most $\alpha q / \sqrt{2}r$
- Thus $b = \lfloor \langle y, x \rangle \rfloor \bmod q = \lfloor \langle y, v \rangle + \langle e, y \rangle \rfloor \bmod q$
- Thus we obtain an error term in the second coordinate of the form $\lfloor e, y \rfloor$
- $|\langle e, y \rangle| \leq \|e\| \|y\| \leq \alpha q / \sqrt{2}r \times r \approx \alpha q$.
- Being the inner product of a fixed vector with a discrete Gaussian vector, this error term is essentially normally distributed with standard deviation at most roughly αq , as required.

Variants of the LWE

- the LWE problem can be reduced to many, apparently easier, problems
- these reductions are one of the main reason the LWE problem finds so many applications in cryptography
- search to decision reduction, showing that it suffices to distinguish LWE samples from entirely uniform samples
- worst-case to average-case reduction, showing that it suffices to solve this distinguishing task for a uniform secret $s \in \mathbb{Z}_q^n$

Variants of the LWE

- the LWE problem can be reduced to many, apparently easier, problems
- these reductions are one of the main reason the LWE problem finds so many applications in cryptography
- search to decision reduction, showing that it suffices to distinguish LWE samples from entirely uniform samples
- worst-case to average-case reduction, showing that it suffices to solve this distinguishing task for a uniform secret $s \in \mathbb{Z}_q^n$

Variants of the LWE

- the LWE problem can be reduced to many, apparently easier, problems
- these reductions are one of the main reason the LWE problem finds so many applications in cryptography
- search to decision reduction, showing that it suffices to distinguish LWE samples from entirely uniform samples
- worst-case to average-case reduction, showing that it suffices to solve this distinguishing task for a uniform secret $s \in \mathbb{Z}_q^n$

Variants of the LWE

- the LWE problem can be reduced to many, apparently easier, problems
- these reductions are one of the main reason the LWE problem finds so many applications in cryptography
- search to decision reduction, showing that it suffices to distinguish LWE samples from entirely uniform samples
- worst-case to average-case reduction, showing that it suffices to solve this distinguishing task for a uniform secret $s \in \mathbb{Z}_q^n$

Variants of the LWE

- the LWE problem can be reduced to many, apparently easier, problems
- these reductions are one of the main reason the LWE problem finds so many applications in cryptography
- search to decision reduction, showing that it suffices to distinguish LWE samples from entirely uniform samples
- worst-case to average-case reduction, showing that it suffices to solve this distinguishing task for a uniform secret $s \in \mathbb{Z}_q^n$

Search to Decision

- Assume we have access to an oracle W that
 - for all s accepts with high probability on inputs from $\mathcal{A}_{s,\chi}$
 - and rejects with high probability on inputs from \mathcal{A}_U
- then, there exists an efficient algorithm W' that, given samples from $\mathcal{A}_{s,\chi}$ for some unknown s , outputs s with high probability
- Proof Outline:
 - For each $k \in \mathbb{Z}_q$, given a pair (a, b) , output $(a + (r, 0, \dots, 0), b + r \cdot k)$, where $r \in_R \mathbb{Z}_q$
 - If $k = s_1$, the transformation takes $\mathcal{A}_{s,\chi}$ to itself
 - If $k \neq s_1$, it takes to \mathcal{A}_U (requires q to be prime)
 - there are q possibilities for s_1 , $q < \text{poly}(n)$

Search to Decision

- Assume we have access to an oracle W that
 - for all s accepts with high probability on inputs from $\mathcal{A}_{s,\chi}$
 - and rejects with high probability on inputs from \mathcal{A}_U
- then, there exists an efficient algorithm W' that, given samples from $\mathcal{A}_{s,\chi}$ for some unknown s , outputs s with high probability
- Proof Outline:
 - For each $k \in \mathbb{Z}_q$, given a pair (a, b) , output $(a + (r, 0, \dots, 0), b + r \cdot k)$, where $r \in_R \mathbb{Z}_q$
 - If $k = s_1$, the transformation takes $\mathcal{A}_{s,\chi}$ to itself
 - If $k \neq s_1$, it takes to \mathcal{A}_U (requires q to be prime)
 - there are q possibilities for s_1 , $q < \text{poly}(n)$

Search to Decision

- Assume we have access to an oracle W that
 - for all s accepts with high probability on inputs from $\mathcal{A}_{s,\chi}$
 - and rejects with high probability on inputs from \mathcal{A}_U
- then, there exists an efficient algorithm W' that, given samples from $\mathcal{A}_{s,\chi}$ for some unknown s , outputs s with high probability
- Proof Outline:
 - For each $k \in \mathbb{Z}_q$, given a pair (a, b) , output $(a + (r, 0, \dots, 0), b + r \cdot k)$, where $r \in_R \mathbb{Z}_q$
 - If $k = s_1$, the transformation takes $\mathcal{A}_{s,\chi}$ to itself
 - If $k \neq s_1$, it takes to \mathcal{A}_U (requires q to be prime)
 - there are q possibilities for s_1 , $q < \text{poly}(n)$

Search to Decision

- Assume we have access to an oracle W that
 - for all s accepts with high probability on inputs from $\mathcal{A}_{s,\chi}$
 - and rejects with high probability on inputs from \mathcal{A}_U
- then, there exists an efficient algorithm W' that, given samples from $\mathcal{A}_{s,\chi}$ for some unknown s , outputs s with high probability
- Proof Outline:
 - For each $k \in \mathbb{Z}_q$, given a pair (a, b) , output $(a + (r, 0, \dots, 0), b + r \cdot k)$, where $r \in_R \mathbb{Z}_q$
 - If $k = s_1$, the transformation takes $\mathcal{A}_{s,\chi}$ to itself
 - If $k \neq s_1$, it takes to \mathcal{A}_U (requires q to be prime)
 - there are q possibilities for s_1 , $q < \text{poly}(n)$

Search to Decision

- Assume we have access to an oracle W that
 - for all s accepts with high probability on inputs from $\mathcal{A}_{s,\chi}$
 - and rejects with high probability on inputs from \mathcal{A}_U
- then, there exists an efficient algorithm W' that, given samples from $\mathcal{A}_{s,\chi}$ for some unknown s , outputs s with high probability
- Proof Outline:
 - For each $k \in \mathbb{Z}_q$, given a pair (a, b) , output $(a + (r, 0, \dots, 0), b + r \cdot k)$, where $r \in_R \mathbb{Z}_q$
 - If $k = s_1$, the transformation takes $\mathcal{A}_{s,\chi}$ to itself
 - If $k \neq s_1$, it takes to \mathcal{A}_U (requires q to be prime)
 - there are q possibilities for s_1 , $q < \text{poly}(n)$

Search to Decision

- Assume we have access to an oracle W that
 - for all s accepts with high probability on inputs from $\mathcal{A}_{s,\chi}$
 - and rejects with high probability on inputs from \mathcal{A}_U
- then, there exists an efficient algorithm W' that, given samples from $\mathcal{A}_{s,\chi}$ for some unknown s , outputs s with high probability
- Proof Outline:
 - For each $k \in \mathbb{Z}_q$, given a pair (a, b) , output $(a + (r, 0, \dots, 0), b + r \cdot k)$, where $r \in_R \mathbb{Z}_q$
 - If $k = s_1$, the transformation takes $\mathcal{A}_{s,\chi}$ to itself
 - If $k \neq s_1$, it takes to \mathcal{A}_U (requires q to be prime)
 - there are q possibilities for s_1 , $q < \text{poly}(n)$

Search to Decision

- Assume we have access to an oracle W that
 - for all s accepts with high probability on inputs from $\mathcal{A}_{s,\chi}$
 - and rejects with high probability on inputs from \mathcal{A}_U
- then, there exists an efficient algorithm W' that, given samples from $\mathcal{A}_{s,\chi}$ for some unknown s , outputs s with high probability
- Proof Outline:
 - For each $k \in \mathbb{Z}_q$, given a pair (a, b) , output $(a + (r, 0, \dots, 0), b + r \cdot k)$, where $r \in_R \mathbb{Z}_q$
 - If $k = s_1$, the transformation takes $\mathcal{A}_{s,\chi}$ to itself
 - If $k \neq s_1$, it takes to \mathcal{A}_U (requires q to be prime)
 - there are q possibilities for s_1 , $q < \text{poly}(n)$

Search to Decision

- Assume we have access to an oracle W that
 - for all s accepts with high probability on inputs from $\mathcal{A}_{s,\chi}$
 - and rejects with high probability on inputs from \mathcal{A}_U
- then, there exists an efficient algorithm W' that, given samples from $\mathcal{A}_{s,\chi}$ for some unknown s , outputs s with high probability
- Proof Outline:
 - For each $k \in \mathbb{Z}_q$, given a pair (a, b) , output $(a + (r, 0, \dots, 0), b + r \cdot k)$, where $r \in_R \mathbb{Z}_q$
 - If $k = s_1$, the transformation takes $\mathcal{A}_{s,\chi}$ to itself
 - If $k \neq s_1$, it takes to \mathcal{A}_U (requires q to be prime)
 - there are q possibilities for s_1 , $q < \text{poly}(n)$

Search to Decision

- Assume we have access to an oracle W that
 - for all s accepts with high probability on inputs from $\mathcal{A}_{s,\chi}$
 - and rejects with high probability on inputs from \mathcal{A}_U
- then, there exists an efficient algorithm W' that, given samples from $\mathcal{A}_{s,\chi}$ for some unknown s , outputs s with high probability
- Proof Outline:
 - For each $k \in \mathbb{Z}_q$, given a pair (a, b) , output $(a + (r, 0, \dots, 0), b + r \cdot k)$, where $r \in_R \mathbb{Z}_q$
 - If $k = s_1$, the transformation takes $\mathcal{A}_{s,\chi}$ to itself
 - If $k \neq s_1$, it takes to \mathcal{A}_U (requires q to be prime)
 - there are q possibilities for s_1 , $q < \text{poly}(n)$

Search to Decision

- Assume we have access to an oracle W that
 - for all s accepts with high probability on inputs from $\mathcal{A}_{s,\chi}$
 - and rejects with high probability on inputs from \mathcal{A}_U
- then, there exists an efficient algorithm W' that, given samples from $\mathcal{A}_{s,\chi}$ for some unknown s , outputs s with high probability
- Proof Outline:
 - For each $k \in \mathbb{Z}_q$, given a pair (a, b) , output $(a + (r, 0, \dots, 0), b + r \cdot k)$, where $r \in_R \mathbb{Z}_q$
 - If $k = s_1$, the transformation takes $\mathcal{A}_{s,\chi}$ to itself
 - If $k \neq s_1$, it takes to \mathcal{A}_U (requires q to be prime)
 - there are q possibilities for s_1 , $q < \text{poly}(n)$

Search to Decision

- Assume we have access to an oracle W that
 - for all s accepts with high probability on inputs from $\mathcal{A}_{s,\chi}$
 - and rejects with high probability on inputs from \mathcal{A}_U
- then, there exists an efficient algorithm W' that, given samples from $\mathcal{A}_{s,\chi}$ for some unknown s , outputs s with high probability
- Proof Outline:
 - For each $k \in \mathbb{Z}_q$, given a pair (a, b) , output $(a + (r, 0, \dots, 0), b + r \cdot k)$, where $r \in_R \mathbb{Z}_q$
 - If $k = s_1$, the transformation takes $\mathcal{A}_{s,\chi}$ to itself
 - If $k \neq s_1$, it takes to \mathcal{A}_U (requires q to be prime)
 - there are q possibilities for s_1 , $q < \text{poly}(n)$

Worst-case to Average-case

- Assume we have access to a distinguisher W that distinguishes $\mathcal{A}_{s,\chi}$ from \mathcal{A}_U for a non-negligible fraction of all possible s
- then there exists an efficient algorithm W' that distinguishes for all s
- Proof Outline: For any $t \in \mathbb{Z}_q^n$ consider
 $f_t : \mathbb{Z}_q^n \times \mathbb{Z}_q \rightarrow \mathbb{Z}_q^n \times \mathbb{Z}_q$

$$f_t(a, b) = (a, b + \langle a, t \rangle)$$

- this transforms $\mathcal{A}_{s,\chi}$ to $\mathcal{A}_{s+t,\chi}$; \mathcal{A}_U to itself

Worst-case to Average-case

- Assume we have access to a distinguisher W that distinguishes $\mathcal{A}_{s,\chi}$ from \mathcal{A}_U for a non-negligible fraction of all possible s
- then there exists an efficient algorithm W' that distinguishes for all s
- Proof Outline: For any $t \in \mathbb{Z}_q^n$ consider
 $f_t : \mathbb{Z}_q^n \times \mathbb{Z}_q \rightarrow \mathbb{Z}_q^n \times \mathbb{Z}_q$

$$f_t(a, b) = (a, b + \langle a, t \rangle)$$

- this transforms $\mathcal{A}_{s,\chi}$ to $\mathcal{A}_{s+t,\chi}$; \mathcal{A}_U to itself

Worst-case to Average-case

- Assume we have access to a distinguisher W that distinguishes $\mathcal{A}_{s,\chi}$ from \mathcal{A}_U for a non-negligible fraction of all possible s
- then there exists an efficient algorithm W' that distinguishes for all s
- Proof Outline: For any $t \in \mathbb{Z}_q^n$ consider
 $f_t : \mathbb{Z}_q^n \times \mathbb{Z}_q \rightarrow \mathbb{Z}_q^n \times \mathbb{Z}_q$

$$f_t(a, b) = (a, b + \langle a, t \rangle)$$

- this transforms $\mathcal{A}_{s,\chi}$ to $\mathcal{A}_{s+t,\chi}$; \mathcal{A}_U to itself

Worst-case to Average-case

- Assume we have access to a distinguisher W that distinguishes $\mathcal{A}_{s,\chi}$ from \mathcal{A}_U for a non-negligible fraction of all possible s
- then there exists an efficient algorithm W' that distinguishes for all s
- Proof Outline: For any $t \in \mathbb{Z}_q^n$ consider
 $f_t : \mathbb{Z}_q^n \times \mathbb{Z}_q \rightarrow \mathbb{Z}_q^n \times \mathbb{Z}_q$

$$f_t(\mathbf{a}, b) = (\mathbf{a}, b + \langle \mathbf{a}, t \rangle)$$

- this transforms $\mathcal{A}_{s,\chi}$ to $\mathcal{A}_{s+t,\chi}$; \mathcal{A}_U to itself

Worst-case to Average-case

- Assume we have access to a distinguisher W that distinguishes $\mathcal{A}_{s,\chi}$ from \mathcal{A}_U for a non-negligible fraction of all possible s
- then there exists an efficient algorithm W' that distinguishes for all s
- Proof Outline: For any $t \in \mathbb{Z}_q^n$ consider
 $f_t : \mathbb{Z}_q^n \times \mathbb{Z}_q \rightarrow \mathbb{Z}_q^n \times \mathbb{Z}_q$

$$f_t(a, b) = (a, b + \langle a, t \rangle)$$

- this transforms $\mathcal{A}_{s,\chi}$ to $\mathcal{A}_{s+t,\chi}$; \mathcal{A}_U to itself

Worst-case to Average-case

- Assume we have access to a distinguisher W that distinguishes $\mathcal{A}_{s,\chi}$ from \mathcal{A}_U for a non-negligible fraction of all possible s
- then there exists an efficient algorithm W' that distinguishes for all s
- Proof Outline: For any $t \in \mathbb{Z}_q^n$ consider
 $f_t : \mathbb{Z}_q^n \times \mathbb{Z}_q \rightarrow \mathbb{Z}_q^n \times \mathbb{Z}_q$

$$f_t(a, b) = (a, b + \langle a, t \rangle)$$

- this transforms $\mathcal{A}_{s,\chi}$ to $\mathcal{A}_{s+t,\chi}$; \mathcal{A}_U to itself

Outline of the talk

- 1 LWE problem
- 2 Hardness of LWE
- 3 Cryptographic Applications**

LWE based Public Key Encryption

System Parameter: Integers n (the security parameter), m (number of equations), q modulus, and a real $\alpha > 0$ (noise parameter)

- **Secret Key:** is a vector $s \in_R \mathbb{Z}_q^n$
- **Public Key:** consists of m samples $(a_i, b_i)_{i=1}^m$ from the LWE distribution with secret s , modulus q , and error parameter α .
- **Encryption:** To encrypt each bit of the message, do the following.
 - Choose a string $t = (t_1, \dots, t_m) \in_R \{0, 1\}^m$.
 - The encryption is $(\sum_{i=1}^m t_i \cdot a_i, \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 0
 - else $(\sum_{i=1}^m t_i \cdot a_i, \lfloor \frac{q}{2} \rfloor + \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 1.
- **Decryption:** of a pair (a, b) is 0 if $b - \langle a, s \rangle$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ modulo q , and 1 otherwise

LWE based Public Key Encryption

System Parameter: Integers n (the security parameter), m (number of equations), q modulus, and a real $\alpha > 0$ (noise parameter)

- **Secret Key:** is a vector $s \in_R \mathbb{Z}_q^n$
- **Public Key:** consists of m samples $(a_i, b_i)_{i=1}^m$ from the LWE distribution with secret s , modulus q , and error parameter α .
- **Encryption:** To encrypt each bit of the message, do the following.
 - Choose a string $t = (t_1, \dots, t_m) \in_R \{0, 1\}^m$.
 - The encryption is $(\sum_{i=1}^m t_i \cdot a_i, \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 0
 - else $(\sum_{i=1}^m t_i \cdot a_i, \lfloor \frac{q}{2} \rfloor + \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 1.
- **Decryption:** of a pair (a, b) is 0 if $b - \langle a, s \rangle$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ modulo q , and 1 otherwise

LWE based Public Key Encryption

System Parameter: Integers n (the security parameter), m (number of equations), q modulus, and a real $\alpha > 0$ (noise parameter)

- **Secret Key:** is a vector $s \in_R \mathbb{Z}_q^n$
- **Public Key:** consists of m samples $(a_i, b_i)_{i=1}^m$ from the LWE distribution with secret s , modulus q , and error parameter α .
- **Encryption:** To encrypt each bit of the message, do the following.
 - Choose a string $t = (t_1, \dots, t_m) \in_R \{0, 1\}^m$.
 - The encryption is $(\sum_{i=1}^m t_i \cdot a_i, \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 0
 - else $(\sum_{i=1}^m t_i \cdot a_i, \lfloor \frac{q}{2} \rfloor + \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 1.
- **Decryption:** of a pair (a, b) is 0 if $b - \langle a, s \rangle$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ modulo q , and 1 otherwise

LWE based Public Key Encryption

System Parameter: Integers n (the security parameter), m (number of equations), q modulus, and a real $\alpha > 0$ (noise parameter)

- **Secret Key:** is a vector $s \in_R \mathbb{Z}_q^n$
- **Public Key:** consists of m samples $(a_i, b_i)_{i=1}^m$ from the LWE distribution with secret s , modulus q , and error parameter α .
- **Encryption:** To encrypt each bit of the message, do the following.
 - Choose a string $t = (t_1, \dots, t_m) \in_R \{0, 1\}^m$.
 - The encryption is $(\sum_{i=1}^m t_i \cdot a_i, \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 0
 - else $(\sum_{i=1}^m t_i \cdot a_i, \lfloor \frac{q}{2} \rfloor + \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 1.
- **Decryption:** of a pair (a, b) is 0 if $b - \langle a, s \rangle$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ modulo q , and 1 otherwise

LWE based Public Key Encryption

System Parameter: Integers n (the security parameter), m (number of equations), q modulus, and a real $\alpha > 0$ (noise parameter)

- **Secret Key:** is a vector $s \in_R \mathbb{Z}_q^n$
- **Public Key:** consists of m samples $(a_i, b_i)_{i=1}^m$ from the LWE distribution with secret s , modulus q , and error parameter α .
- **Encryption:** To encrypt each bit of the message, do the following.
 - Choose a string $t = (t_1, \dots, t_m) \in_R \{0, 1\}^m$.
 - The encryption is $(\sum_{i=1}^m t_i \cdot a_i, \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 0
 - else $(\sum_{i=1}^m t_i \cdot a_i, \lfloor \frac{q}{2} \rfloor + \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 1.
- **Decryption:** of a pair (a, b) is 0 if $b - \langle a, s \rangle$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ modulo q , and 1 otherwise

LWE based Public Key Encryption

System Parameter: Integers n (the security parameter), m (number of equations), q modulus, and a real $\alpha > 0$ (noise parameter)

- **Secret Key:** is a vector $s \in_R \mathbb{Z}_q^n$
- **Public Key:** consists of m samples $(a_i, b_i)_{i=1}^m$ from the LWE distribution with secret s , modulus q , and error parameter α .
- **Encryption:** To encrypt each bit of the message, do the following.
 - Choose a string $t = (t_1, \dots, t_m) \in_R \{0, 1\}^m$.
 - The encryption is $(\sum_{i=1}^m t_i \cdot a_i, \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 0
 - else $(\sum_{i=1}^m t_i \cdot a_i, \lfloor \frac{q}{2} \rfloor + \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 1.
- **Decryption:** of a pair (a, b) is 0 if $b - \langle a, s \rangle$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ modulo q , and 1 otherwise

LWE based Public Key Encryption

System Parameter: Integers n (the security parameter), m (number of equations), q modulus, and a real $\alpha > 0$ (noise parameter)

- **Secret Key:** is a vector $s \in_R \mathbb{Z}_q^n$
- **Public Key:** consists of m samples $(a_i, b_i)_{i=1}^m$ from the LWE distribution with secret s , modulus q , and error parameter α .
- **Encryption:** To encrypt each bit of the message, do the following.
 - Choose a string $t = (t_1, \dots, t_m) \in_R \{0, 1\}^m$.
 - The encryption is $(\sum_{i=1}^m t_i \cdot a_i, \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 0
 - else $(\sum_{i=1}^m t_i \cdot a_i, \lfloor \frac{q}{2} \rfloor + \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 1.
- **Decryption:** of a pair (a, b) is 0 if $b - \langle a, s \rangle$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ modulo q , and 1 otherwise

LWE based Public Key Encryption

System Parameter: Integers n (the security parameter), m (number of equations), q modulus, and a real $\alpha > 0$ (noise parameter)

- **Secret Key:** is a vector $s \in_R \mathbb{Z}_q^n$
- **Public Key:** consists of m samples $(a_i, b_i)_{i=1}^m$ from the LWE distribution with secret s , modulus q , and error parameter α .
- **Encryption:** To encrypt each bit of the message, do the following.
 - Choose a string $t = (t_1, \dots, t_m) \in_R \{0, 1\}^m$.
 - The encryption is $(\sum_{i=1}^m t_i \cdot a_i, \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 0
 - else $(\sum_{i=1}^m t_i \cdot a_i, \lfloor \frac{q}{2} \rfloor + \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 1.
- **Decryption:** of a pair (a, b) is 0 if $b - \langle a, s \rangle$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ modulo q , and 1 otherwise

LWE based Public Key Encryption

System Parameter: Integers n (the security parameter), m (number of equations), q modulus, and a real $\alpha > 0$ (noise parameter)

- **Secret Key:** is a vector $s \in_R \mathbb{Z}_q^n$
- **Public Key:** consists of m samples $(a_i, b_i)_{i=1}^m$ from the LWE distribution with secret s , modulus q , and error parameter α .
- **Encryption:** To encrypt each bit of the message, do the following.
 - Choose a string $t = (t_1, \dots, t_m) \in_R \{0, 1\}^m$.
 - The encryption is $(\sum_{i=1}^m t_i \cdot a_i, \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 0
 - else $(\sum_{i=1}^m t_i \cdot a_i, \lfloor \frac{q}{2} \rfloor + \sum_{i=1}^m t_i \cdot b_i)$ if the bit is 1.
- **Decryption:** of a pair (a, b) is 0 if $b - \langle a, s \rangle$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ modulo q , and 1 otherwise

Correctness

- One possible choice of parameters that guarantees both correctness and security is the following.
- Choose q to be a prime between n^2 and $2n^2$,
 $m = 1.1 \cdot n \log q$, and $\alpha = 1/\sqrt{n} \log^2 n$
- Note that a decryption error occurs if the sum of the error terms is greater than $q/4$
- Since we are summing at most m normal error terms, each with standard deviation αq , the standard deviation of the sum is at most $\sqrt{m} \alpha q < q/\log n$
- A standard calculation shows that the probability that such a normal variable is greater than $q/4$ is negligible.

Correctness

- One possible choice of parameters that guarantees both correctness and security is the following.
- Choose q to be a prime between n^2 and $2n^2$,
 $m = 1.1 \cdot n \log q$, and $\alpha = 1/\sqrt{n} \log^2 n$
- Note that a decryption error occurs if the sum of the error terms is greater than $q/4$
- Since we are summing at most m normal error terms, each with standard deviation αq , the standard deviation of the sum is at most $\sqrt{m} \alpha q < q/\log n$
- A standard calculation shows that the probability that such a normal variable is greater than $q/4$ is negligible.

Correctness

- One possible choice of parameters that guarantees both correctness and security is the following.
- Choose q to be a prime between n^2 and $2n^2$,
 $m = 1.1 \cdot n \log q$, and $\alpha = 1/\sqrt{n \log^2 n}$
- Note that a decryption error occurs if the sum of the error terms is greater than $q/4$
- Since we are summing at most m normal error terms, each with standard deviation αq , the standard deviation of the sum is at most $\sqrt{m} \alpha q < q/\log n$
- A standard calculation shows that the probability that such a normal variable is greater than $q/4$ is negligible.

Correctness

- One possible choice of parameters that guarantees both correctness and security is the following.
- Choose q to be a prime between n^2 and $2n^2$,
 $m = 1.1 \cdot n \log q$, and $\alpha = 1/\sqrt{n} \log^2 n$
- Note that a decryption error occurs if the sum of the error terms is greater than $q/4$
- Since we are summing at most m normal error terms, each with standard deviation αq , the standard deviation of the sum is at most $\sqrt{m} \alpha q < q/\log n$
- A standard calculation shows that the probability that such a normal variable is greater than $q/4$ is negligible.

Correctness

- One possible choice of parameters that guarantees both correctness and security is the following.
- Choose q to be a prime between n^2 and $2n^2$,
 $m = 1.1 \cdot n \log q$, and $\alpha = 1/\sqrt{n} \log^2 n$
- Note that a decryption error occurs if the sum of the error terms is greater than $q/4$
- Since we are summing at most m normal error terms, each with standard deviation αq , the standard deviation of the sum is at most $\sqrt{m} \alpha q < q/\log n$
- A standard calculation shows that the probability that such a normal variable is greater than $q/4$ is negligible.

Correctness

- One possible choice of parameters that guarantees both correctness and security is the following.
- Choose q to be a prime between n^2 and $2n^2$,
 $m = 1.1 \cdot n \log q$, and $\alpha = 1/\sqrt{n \log^2 n}$
- Note that a decryption error occurs if the sum of the error terms is greater than $q/4$
- Since we are summing at most m normal error terms, each with standard deviation αq , the standard deviation of the sum is at most $\sqrt{m} \alpha q < q/\log n$
- A standard calculation shows that the probability that such a normal variable is greater than $q/4$ is negligible.

Thank You

- Reference
 - Oded Regev. The Learning with Errors Problem. Invited survey in CCC 2010

Making NTRUencrypt as secure as worst-case problems over ideal lattices

Damien Stehlé and Ron Steinfeld

CNRS – ENS de Lyon
Macquarie University

Kolkata, January 2012

The NTRU cryptographic functions

NTRUEncrypt: A public-key encryption scheme.

- 1996: Proposed by Hoffstein, Pipher & Silverman.
- 1997: Improved lattice attacks by Coppersmith & Shamir.
- 1998: Revised by Hoffstein et al.

⇒ The design has proven fairly robust over time.

The NTRU cryptographic functions

NTRUEncrypt: A public-key encryption scheme.

- 1996: Proposed by Hoffstein, Pipher & Silverman.
- 1997: Improved lattice attacks by Coppersmith & Shamir.
- 1998: Revised by Hoffstein et al.

⇒ The design has proven fairly robust over time.

Why studying NTRUencrypt?

- Standardized & commercialized.
- Super-fast (comparison to 1024-bit RSA, based on an NTRU brochure):
 - Encryption ~ 10 times faster.
 - Decryption ~ 100 times faster.
 - Asymptotically: $\tilde{O}(\lambda)$ versus $\tilde{O}(\lambda^6)$, for security 2^λ .
- Interesting security features:
 - Does not rely on the hardness of Int-Fac or DLog.
 - Seems to resist practical attacks.
 - Seems to resist quantum attacks.

Why studying NTRUencrypt?

- Standardized & commercialized.
- Super-fast (comparison to 1024-bit RSA, based on an NTRU brochure):
 - Encryption ~ 10 times faster.
 - Decryption ~ 100 times faster.
 - Asymptotically: $\tilde{O}(\lambda)$ versus $\tilde{O}(\lambda^6)$, for security 2^λ .
- Interesting security features:
 - Does not rely on the hardness of Int-Fac or DLog.
 - Seems to resist practical attacks.
 - Seems to resist quantum attacks.

Our main result

An IND-CPA variant of NTRUencrypt

It is possible to modify NTRUencrypt so that:

- Encryption and decryption of λ bits still cost $\tilde{O}(\lambda)$.
- Any semantic attack running in time $\left\{ \begin{array}{l} \text{Poly}(n) \\ 2^{\alpha(n)} \end{array} \right\}$ leads to a $\left\{ \begin{array}{l} \text{Poly}(n) \\ 2^{\alpha(n)} \end{array} \right\}$ quantum algorithm for $\text{Poly}(n)$ -Ideal-SVP.

Our main result

An IND-CPA variant of NTRUencrypt

It is possible to modify NTRUencrypt so that:

- Encryption and decryption of λ bits still cost $\tilde{O}(\lambda)$.
- Any semantic attack running in time $\left\{ \begin{array}{l} \text{Poly}(n) \\ 2^{o(n)} \end{array} \right\}$ leads to a $\left\{ \begin{array}{l} \text{Poly}(n) \\ 2^{o(n)} \end{array} \right\}$ quantum algorithm for $\text{Poly}(n)$ -Ideal-SVP.

Our main result

An IND-CPA variant of NTRUEncrypt

It is possible to modify NTRUEncrypt so that:

- Encryption and decryption of λ bits still cost $\tilde{O}(\lambda)$.
- Any semantic attack running in time $\left\{ \begin{matrix} \text{Poly}(n) \\ 2^{o(n)} \end{matrix} \right\}$ leads to a $\left\{ \begin{matrix} \text{Poly}(n) \\ 2^{o(n)} \end{matrix} \right\}$ quantum algorithm for $\text{Poly}(n)$ -Ideal-SVP.
- Similar result for NTRUSign, in the random oracle model and with a non-quantum security proof.
- Relies on results from [LyPeRe'10, GePeVa'08].

Outline of the talk

- 1- **Regular** NTRUEncrypt.
- 2- The R-LWE problem.
- 3- The modified NTRUEncrypt.

Polynomial Rings: Generalizing \mathbb{Z}

Take $\Phi \in \mathbb{Z}[x]$ monic of degree n .

$$R^\Phi := \left[\mathbb{Z}[x]/(\Phi), +, \times \right].$$

Interesting Φ 's:

- $\Phi = x^n - 1 \rightarrow R^-$, $\Phi = x^n + 1 \rightarrow R^+$.
- $x^n + 1$ irreducible if n is a power of 2.
- In this case, R^Φ is the ring of integers of the cyclotomic number field:

$$\mathbb{Q}[e^{i\pi/n}] \simeq \mathbb{Q}[x]/(\Phi).$$

Polynomial Rings: Generalizing \mathbb{Z}

Take $\Phi \in \mathbb{Z}[x]$ monic of degree n .

$$R^\Phi := \left[\mathbb{Z}[x]/(\Phi), +, \times \right].$$

Interesting Φ 's:

- $\Phi = x^n - 1 \rightarrow R^-$, $\Phi = x^n + 1 \rightarrow R^+$.
- $x^n + 1$ irreducible if n is a power of 2.
- In this case, R^Φ is the ring of integers of the cyclotomic number field:

$$\mathbb{Q}[e^{i\pi/n}] \simeq \mathbb{Q}[x]/(\Phi).$$

Polynomial Rings: Generalizing $\mathbb{Z}/q\mathbb{Z}$

Let $q \geq 2$ and $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$.

$$R_q^\Phi := \left[\mathbb{Z}_q[x]/(\Phi), +, \times \right] = R^\Phi/(q) = \mathbb{Z}[x]/(\Phi, q).$$

- Arithmetic in R_q^Φ costs $\tilde{O}(n \log q)$.
- R_q^- and R_q^+ defined similarly.
- If $\Phi = x^n \pm 1$ has n distinct linear factors modulo prime q , then R_q^Φ comes with a natural FFT.

Polynomial Rings: Generalizing $\mathbb{Z}/q\mathbb{Z}$

Let $q \geq 2$ and $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$.

$$R_q^\Phi := \left[\mathbb{Z}_q[x]/(\Phi), +, \times \right] = R^\Phi/(q) = \mathbb{Z}[x]/(\Phi, q).$$

- Arithmetic in R_q^Φ costs $\tilde{O}(n \log q)$.
- R_q^- and R_q^+ defined similarly.
- If $\Phi = x^n \pm 1$ has n distinct linear factors modulo prime q , then R_q^Φ comes with a natural FFT.

If $f \in R^\Phi$ has coefficients in $(-q/2, q/2)$, then $(f \bmod q)$ is f .

Polynomial Rings: Generalizing $\mathbb{Z}/q\mathbb{Z}$

Let $q \geq 2$ and $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$.

$$R_q^\Phi := \left[\mathbb{Z}_q[x]/(\Phi), +, \times \right] = R^\Phi/(q) = \mathbb{Z}[x]/(\Phi, q).$$

- Arithmetic in R_q^Φ costs $\tilde{O}(n \log q)$.
- R_q^- and R_q^+ defined similarly.
- If $\Phi = x^n \pm 1$ has n distinct linear factors modulo prime q , then R_q^Φ comes with a natural FFT.

The key to decryption correctness

If $f \in R^\Phi$ has coefficients in $(-q/2, q/2)$, then $(f \bmod q)$ is f .

Polynomial Rings: Generalizing $\mathbb{Z}/q\mathbb{Z}$

Let $q \geq 2$ and $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$.

$$R_q^\Phi := \left[\mathbb{Z}_q[x]/(\Phi), +, \times \right] = R^\Phi/(q) = \mathbb{Z}[x]/(\Phi, q).$$

- Arithmetic in R_q^Φ costs $\tilde{O}(n \log q)$.
- R_q^- and R_q^+ defined similarly.
- If $\Phi = x^n \pm 1$ has n distinct linear factors modulo prime q , then R_q^Φ comes with a natural FFT.

The key to decryption correctness

If $f \in R^\Phi$ has coefficients in $(-q/2, q/2)$, then $(f \bmod q)$ is f .

Description of NTRUencrypt, Part I

Parameters: n prime, $q \approx n$ a power of 2.

E.g.: $(n, q) = (503, 256)$.

- **Secret key sk :** $f, g \in R^-$ such that:
 - f is invertible mod q and mod 3.
 - The coeffs of f and g are in $\{-1, 0, 1\}$.
- **Public key pk :** $h = g/f \bmod q$.

Security intuition

Given $h \in R_q$, finding $g, f \in R$ small s.t. $h = g/f [q]$ is hard.

Description of NTRUencrypt, Part I

Parameters: n prime, $q \approx n$ a power of 2.

E.g.: $(n, q) = (503, 256)$.

- **Secret key** sk : $f, g \in R^-$ such that:
 - f is invertible mod q and mod 3.
 - The coeffs of f and g are in $\{-1, 0, 1\}$.
- **Public key** pk : $h = g/f \text{ mod } q$.

Security intuition

Given $h \in R_q$, finding $g, f \in R$ small s.t. $h = g/f [q]$ is hard.

Description of NTRUencrypt, Part I

Parameters: n prime, $q \approx n$ a power of 2.

E.g.: $(n, q) = (503, 256)$.

- **Secret key** sk : $f, g \in R^-$ such that:
 - f is invertible mod q and mod 3.
 - The coeffs of f and g are in $\{-1, 0, 1\}$.
- **Public key** pk : $h = g/f \text{ mod } q$.

Security intuition

Given $h \in R_q$, finding $g, f \in R$ small s.t. $h = g/f [q]$ is hard.

Description of NTRUencrypt, Part I

Parameters: n prime, $q \approx n$ a power of 2.

E.g.: $(n, q) = (503, 256)$.

- **Secret key** sk : $f, g \in R^-$ such that:
 - f is invertible mod q and mod 3.
 - The coeffs of f and g are in $\{-1, 0, 1\}$.
- **Public key** pk : $h = g/f \text{ mod } q$.

Security intuition

Given $h \in R_q$, finding $g, f \in R$ small s.t. $h = g/f [q]$ is hard.

Description of NTRUencrypt, Part II

- sk : $f, g \in R$ small with f invertible mod q and mod 3.
- pk : $h = g/f \text{ mod } q$.

Description of NTRUencrypt, Part II

- sk : $f, g \in R$ small with f invertible mod q and mod 3.
- pk : $h = g/f \bmod q$.

Encryption of $M \in \{0, 1\}[x]/(x^n - 1)$:

- Sample $s \in R_q^-$ with coeffs in $\{-1, 0, 1\}$,
- Return $C := 3hs + M \bmod q$.

Description of NTRUencrypt, Part II

- sk : $f, g \in R$ small with f invertible mod q and mod 3.
- pk : $h = g/f \text{ mod } q$.

Encryption of $M \in \{0, 1\}[x]/(x^n - 1)$:

- Sample $s \in R_q^-$ with coeffs in $\{-1, 0, 1\}$,
- Return $C := 3hs + M \text{ mod } q$.

Decryption of $C \in R_q^-$:

- $f \times C = 3gs + fM \text{ mod } q$.
- g, M, f, s small \Rightarrow equality holds over R^- .
- $(f \times C \text{ mod } q) \text{ mod } 3 = fM \text{ mod } 3$.
- Multiply by the inverse of $f \text{ mod } 3$.

Security intuition

Given $C \in R_q$, finding $M, s \in R$ small s.t. $C = 3hs + M \text{ [} q \text{]}$ is hard.

Description of NTRUencrypt, Part II

- sk : $f, g \in R$ small with f invertible mod q and mod 3.
- pk : $h = g/f \text{ mod } q$.

Encryption of $M \in \{0, 1\}[x]/(x^n - 1)$:

- Sample $s \in R_q^-$ with coeffs in $\{-1, 0, 1\}$,
- Return $C := 3hs + M \text{ mod } q$.

Decryption of $C \in R_q^-$:

- $f \times C = 3gs + fM \text{ mod } q$.
- g, M, f, s small \Rightarrow equality holds over R^- .
- $(f \times C \text{ mod } q) \text{ mod } 3 = fM \text{ mod } 3$.
- Multiply by the inverse of $f \text{ mod } 3$.

Security intuition

Given $C \in R_q$, finding $M, s \in R$ small s.t. $C = 3hs + M [q]$ is hard.

Outline of the talk

- 1- Regular NTRUEncrypt.
- 2- **The R-LWE problem.**
- 3- The modified NTRUEncrypt.

Ideals in R^Φ

- $I \subseteq R^\Phi$ is an **ideal** if:

$$\forall a, b \in I, \forall r \in R^\Phi : a + b \cdot r \in I.$$

- Let's identify polynomials to vectors via their coefficients:

$$\begin{aligned} R^\Phi &\rightarrow \mathbb{Z}^n \\ \sum_{i < n} f_i x^i &\mapsto (f_0, \dots, f_{n-1})^t \end{aligned}$$

- Ideal I is mapped to an integer **lattice**.
- A **Φ -ideal lattice** is a lattice corresponding to an ideal of R^Φ .

Ideals in R^Φ

- $I \subseteq R^\Phi$ is an **ideal** if:

$$\forall a, b \in I, \forall r \in R^\Phi : a + b \cdot r \in I.$$

- Let's identify polynomials to vectors via their coefficients:

$$\begin{aligned} R^\Phi &\rightarrow \mathbb{Z}^n \\ \sum_{i < n} f_i x^i &\mapsto (f_0, \dots, f_{n-1})^t \end{aligned}$$

- Ideal I is mapped to an integer **lattice**.
- A **Φ -ideal lattice** is a lattice corresponding to an ideal of R^Φ .

Ideals in R^Φ

- $I \subseteq R^\Phi$ is an **ideal** if:

$$\forall a, b \in I, \forall r \in R^\Phi : a + b \cdot r \in I.$$

- Let's identify polynomials to vectors via their coefficients:

$$\begin{aligned} R^\Phi &\rightarrow \mathbb{Z}^n \\ \sum_{i < n} f_i x^i &\mapsto (f_0, \dots, f_{n-1})^t \end{aligned}$$

- Ideal I is mapped to an integer **lattice**.
- A **Φ -ideal lattice** is a lattice corresponding to an ideal of R^Φ .

(Integral) lattices and the Shortest Vector Problem

Lattice $\equiv \{ \sum_{i \leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z} \}$,
for some lin. independent \mathbf{b}_i 's.

Minimum: $\lambda = \min(\|\mathbf{b}\| : \mathbf{b} \in L \setminus \mathbf{0})$.

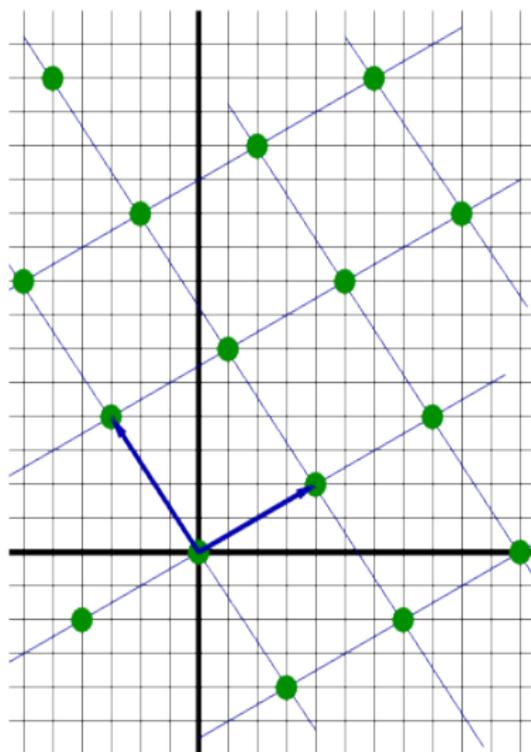
γ -SVP (computational variant)

Find $\mathbf{b} \in L$ with: $0 < \|\mathbf{b}\| \leq \gamma \cdot \lambda(L)$.

No known sub-exp. algo. for $\gamma = \text{Poly}(n)$.

γ -Ideal-SVP:

- γ -SVP restricted to Φ -ideal lattices.
- Does not seem easier than SVP.
- Standard in algebraic number theory.



(Integral) lattices and the Shortest Vector Problem

Lattice $\equiv \{\sum_{i \leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$,
for some lin. independent \mathbf{b}_i 's.

Minimum: $\lambda = \min(\|\mathbf{b}\| : \mathbf{b} \in L \setminus \mathbf{0})$.

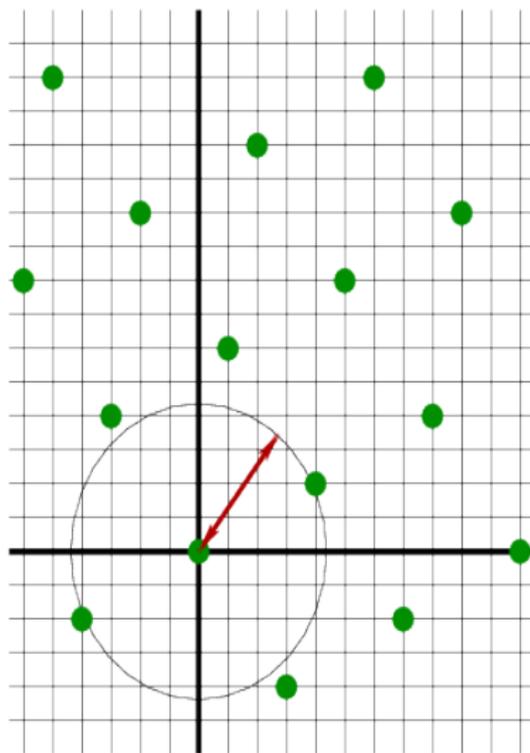
γ -SVP (computational variant)

Find $\mathbf{b} \in L$ with: $0 < \|\mathbf{b}\| \leq \gamma \cdot \lambda(L)$.

No known sub-exp. algo. for $\gamma = \text{Poly}(n)$.

γ -Ideal-SVP:

- γ -SVP restricted to Φ -ideal lattices.
- Does not seem easier than SVP.
- Standard in algebraic number theory.



(Integral) lattices and the Shortest Vector Problem

Lattice $\equiv \{\sum_{i \leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$,
for some lin. independent \mathbf{b}_i 's.

Minimum: $\lambda = \min(\|\mathbf{b}\| : \mathbf{b} \in L \setminus \mathbf{0})$.

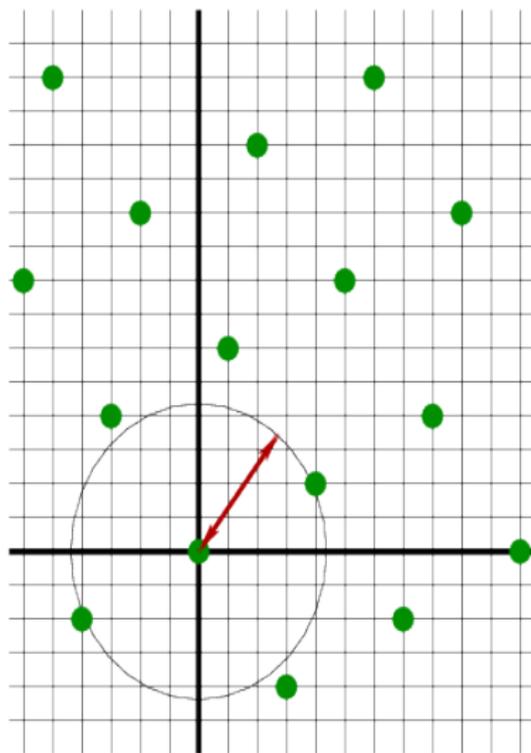
γ -SVP (computational variant)

Find $\mathbf{b} \in L$ with: $0 < \|\mathbf{b}\| \leq \gamma \cdot \lambda(L)$.

No known sub-exp. algo. for $\gamma = \text{Poly}(n)$.

γ -Ideal-SVP:

- γ -SVP restricted to Φ -ideal lattices.
- Does not seem easier than SVP.
- Standard in algebraic number theory.



The R-LWE problem [LyPeRe'10]

- Variant of Learning With Errors [Regev'05], over ring R_q^+ .
- Let ψ be a distribution over R_q^+ . We define D_ψ as the distribution obtained with the following experiment:

Sample $a \leftarrow U(R_q^+)$, $s \leftarrow \psi$, $e \leftarrow \psi$,
Return $(a, as + e) \in R_q^+ \times R_q^+$.

R-LWE $_{q,\psi}$ (Decisional variant with one sample)

Tell whether a given (a, b) is sampled from D_ψ or $U(R_q^+ \times R_q^+)$.

The R-LWE problem [LyPeRe'10]

- Variant of Learning With Errors [Regev'05], over ring R_q^+ .
- Let ψ be a distribution over R_q^+ . We define D_ψ as the distribution obtained with the following experiment:

Sample $a \leftarrow U(R_q^+)$, $s \leftarrow \psi$, $e \leftarrow \psi$,
Return $(a, as + e) \in R_q^+ \times R_q^+$.

R-LWE $_{q,\psi}$ (Decisional variant with one sample)

Tell whether a given (a, b) is sampled from D_ψ or $U(R_q^+ \times R_q^+)$.

R-LWE is hard [LyPeRe'10]

R-LWE $_{q,\psi}$ (Decisional variant with one sample)

Tell whether a given (a, b) is sampled from D_ψ or $U(R_q^+ \times R_q^+)$.

R-LWE is no easier than $\text{Poly}(n)$ -Ideal-SVP

Take $q = \text{Poly}(n)$ with $q = 1 \pmod{2n}$. There exists ψ s.t. solving R-LWE $_{q,\psi}$ with non-negligible advantage is computationally infeasible, assuming the quantum hardness of $\text{Poly}(n)$ -Ideal-SVP.

- Sampling from ψ can be done in time $\tilde{O}(n)$.
- Samples from ψ are small with very high probability: their Euclidean norms are $\leq q^\alpha$ for arbitrary $\alpha \in (0, 1)$.

R-LWE is hard [LyPeRe'10]

R-LWE $_{q,\psi}$ (Decisional variant with one sample)

Tell whether a given (a, b) is sampled from D_ψ or $U(R_q^+ \times R_q^+)$.

R-LWE is no easier than $\mathcal{Poly}(n)$ -Ideal-SVP

Take $q = \mathcal{Poly}(n)$ with $q = 1 \pmod{2n}$. There exists ψ s.t. solving R-LWE $_{q,\psi}$ with non-negligible advantage is computationally infeasible, assuming the quantum hardness of $\mathcal{Poly}(n)$ -Ideal-SVP.

- Sampling from ψ can be done in time $\tilde{O}(n)$.
- Samples from ψ are small with very high probability: their Euclidean norms are $\leq q^\alpha$ for arbitrary $\alpha \in (0, 1)$.

Outline of the talk

- 1- Regular NTRUEncrypt.
- 2- The R-LWE problem.
- 3- **The modified** NTRUEncrypt.

Some intuition

NTRUencrypt:

- pk: $h = g/f \in R_q^-$ with f, g small.
- Enc: $M \mapsto 3hs + M \pmod q$, where s is small.
- IND-CPA: we would like $(h, 3hs)$ to be pseudo-random.
- It's not! Divide RHS by h and check for smallness.

R-LWE with one sample:

- $(a, as + e) \approx^c U(R_q^+ \times R_q^+)$, where $a \leftarrow U(R_q^+)$, $s, e \leftarrow \psi$.
- Let's change rings and replace " (h, hs) " by " $(a, as + e)$ "!

Some intuition

NTRUencrypt:

- pk: $h = g/f \in R_q^-$ with f, g small.
- Enc: $M \mapsto 3hs + M \pmod q$, where s is small.
- IND-CPA: we would like $(h, 3hs)$ to be pseudo-random.
- It's not! Divide RHS by h and check for smallness.

R-LWE with one sample:

- $(a, as + e) \approx^c U(R_q^+ \times R_q^+)$, where $a \leftarrow U(R_q^+)$, $s, e \leftarrow \psi$.
- Let's change rings and replace " (h, hs) " by " $(a, as + e)$ "!

Some intuition

NTRUencrypt:

- pk: $h = g/f \in R_q^-$ with f, g small.
- Enc: $M \mapsto 3hs + M \bmod q$, where s is small.
- IND-CPA: we would like $(h, 3hs)$ to be pseudo-random.
- It's not! Divide RHS by h and check for smallness.

R-LWE with one sample:

- $(a, as + e) \approx^c U(R_q^+ \times R_q^+)$, where $a \leftarrow U(R_q^+)$, $s, e \leftarrow \psi$.
- Let's change rings and replace " (h, hs) " by " $(a, as + e)$ "!

Some intuition

NTRUEncrypt:

- pk: $h = g/f \in R_q^-$ with f, g small.
- Enc: $M \mapsto 3hs + M \pmod q$, where s is small.
- IND-CPA: we would like $(h, 3hs)$ to be pseudo-random.
- It's not! Divide RHS by h and check for smallness.

R-LWE with one sample:

- $(a, as + e) \approx^c U(R_q^+ \times R_q^+)$, where $a \leftarrow U(R_q^+)$, $s, e \leftarrow \psi$.
- Let's change rings and replace " (h, hs) " by " $(a, as + e)$ "!

Is it that simple?

- Enc: $M \mapsto 3hs + M \pmod q$, where s is small.
- R-LWE: $(a, as + e)$, where $a \leftarrow U(R_q^+)$, $s, e \leftarrow \psi$.
- Changing rings and replacing “ (h, hs) ” by “ $(a, as + e)$ ”?

Good news:

- s, e are small \Rightarrow decryption still works.
- q prime \Rightarrow multiplying by $p = 3$ preserves pseudo-randomness.
- Everything remains (asymptotically) efficient.

The catch:

- Relying on R-LWE requires h uniform in R_q^+ .

Is it that simple?

- Enc: $M \mapsto 3hs + M \pmod q$, where s is small.
- R-LWE: $(a, as + e)$, where $a \leftarrow U(R_q^+)$, $s, e \leftarrow \psi$.
- Changing rings and replacing “ (h, hs) ” by “ $(a, as + e)$ ”?

Good news:

- s, e are small \Rightarrow decryption still works.
- q prime \Rightarrow multiplying by $p = 3$ preserves pseudo-randomness.
- Everything remains (asymptotically) efficient.

The catch:

- Relying on R-LWE requires h uniform in R_q^+ .

Is it that simple?

- Enc: $M \mapsto 3hs + M \bmod q$, where s is small.
- R-LWE: $(a, as + e)$, where $a \leftarrow U(R_q^+)$, $s, e \leftarrow \psi$.
- Changing rings and replacing “ (h, hs) ” by “ $(a, as + e)$ ”?

Good news:

- s, e are small \Rightarrow decryption still works.
- q prime \Rightarrow multiplying by $p = 3$ preserves pseudo-randomness.
- Everything remains (asymptotically) efficient.

The catch:

- Relying on R-LWE requires h uniform in R_q^+ .

The modified scheme

Parameters: n prime, $q \approx n$ a power of 2.

Key generation:

- sk: $f, g \in R^-$ with:
 - f invertible mod q and 3.
 - Coeffs of f and g in $\{-1, 0, 1\}$.
- pk: $h = g/f \bmod q$.

Encryption of $M \in \{0, 1\}[x]/(x^n - 1)$:

- $C := 3hs + M \bmod q$, with coeffs of s in $\{-1, 0, 1\}$.

Decryption of $C \in R_q^-$:

- $f \times C \bmod q = 3gs + fM$.
- $(f \times C \bmod q) \bmod 3 = fM \bmod 3$.
- Multiply by the inverse of $f \bmod 3$.

The modified scheme

Parameters: n a power of 2, $q = \text{Poly}(n)$ prime s.t. $q \equiv 1 \pmod{2n}$.

Key generation:

- sk: $f, g \in R^+$ with:
 - f invertible mod q and 2.
 - Coeffs of f and g of magnitude $\approx \sqrt{q}$.
- pk: $h = g/f \pmod{q}$.

Encryption of $M \in \{0, 1\}[x]/(x^n + 1)$:

- $C := 2(hs + e) + M \pmod{q}$, with $s, e \leftarrow \psi$.

Decryption of $C \in R_q^+$:

- $f \times C \pmod{q} = 2(gs + fe) + fM$.
- $(f \times C \pmod{q}) \pmod{2} = fM \pmod{2}$.
- Multiply by the inverse of $f \pmod{2}$.

Making $h = g/f$ statistically close to uniform

- We want h uniform while having f and g with small coeffs.
- If we want a chance, we need the magnitudes to be $\geq \sqrt{q}$.

The distribution D_σ^\times used for f and g

- 1 Sample f from the discrete Gaussian $D_{\mathbb{Z}^n, \sigma}$, using [GePeVa'08]:

$$\forall x \in \mathbb{Z}^n, \quad D_{\mathbb{Z}^n, \sigma}[x] \sim \exp\left(-\pi \frac{\|x\|^2}{\sigma^2}\right).$$

- 2 If f is not invertible in R_q^+ , restart.

- It's a discrete Gaussian with a non-lattice support.
- We also want f invertible mod 2: handled by tweaking D_σ^\times .

Making $h = g/f$ statistically close to uniform

- We want h uniform while having f and g with small coeffs.
- If we want a chance, we need the magnitudes to be $\geq \sqrt{q}$.

The distribution D_{σ}^{\times} used for f and g

- 1 Sample f from the discrete Gaussian $D_{\mathbb{Z}^n, \sigma}$, using [GePeVa'08]:

$$\forall x \in \mathbb{Z}^n, \quad D_{\mathbb{Z}^n, \sigma}[x] \sim \exp\left(-\pi \frac{\|x\|^2}{\sigma^2}\right).$$

- 2 If f is not invertible in R_q^+ , restart.

- It's a discrete Gaussian with a non-lattice support.
- We also want f invertible mod 2: handled by tweaking D_{σ}^{\times} .

Making $h = g/f$ statistically close to uniform

- We want h uniform while having f and g with small coeffs.
- If we want a chance, we need the magnitudes to be $\geq \sqrt{q}$.

The distribution D_{σ}^{\times} used for f and g

- 1 Sample f from the discrete Gaussian $D_{\mathbb{Z}^n, \sigma}$, using [GePeVa'08]:

$$\forall x \in \mathbb{Z}^n, \quad D_{\mathbb{Z}^n, \sigma}[x] \sim \exp\left(-\pi \frac{\|x\|^2}{\sigma^2}\right).$$

- 2 If f is not invertible in R_q^+ , restart.

- It's a discrete Gaussian with a non-lattice support.
- We also want f invertible mod 2: handled by tweaking D_{σ}^{\times} .

Making $h = g/f$ statistically close to uniform

Our main technical contribution

If $\sigma = \tilde{\Omega}(n \cdot q^{\frac{1}{2} + \varepsilon})$ with $\varepsilon > 0$, then:

$$\Delta \left[\frac{D_\sigma^\times}{D_\sigma^\times} \bmod q, U(R_q^\times) \right] \leq q^{-\Omega(\varepsilon \cdot n)}.$$

- If $f \leftrightarrow D_\sigma^\times$, then $\|f\| \leq \sigma\sqrt{n}$, with overwhelming probability.
- We don't get uniformity in R_q but only in R_q^\times .
- R-LWE is still hard if h is restricted to $U(R_q^\times)$.

Proof: see proceedings of Eurocrypt'11.

Making $h = g/f$ statistically close to uniform

Our main technical contribution

If $\sigma = \tilde{\Omega}(n \cdot q^{\frac{1}{2} + \varepsilon})$ with $\varepsilon > 0$, then:

$$\Delta \left[\frac{D_\sigma^\times}{D_\sigma^\times} \bmod q, U(R_q^\times) \right] \leq q^{-\Omega(\varepsilon \cdot n)}.$$

- If $f \leftrightarrow D_\sigma^\times$, then $\|f\| \leq \sigma\sqrt{n}$, with overwhelming probability.
- We don't get uniformity in R_q but only in R_q^\times .
- R-LWE is still hard if h is restricted to $U(R_q^\times)$.

Proof: see proceedings of Eurocrypt'11.

Making $h = g/f$ statistically close to uniform

Our main technical contribution

If $\sigma = \tilde{\Omega}(n \cdot q^{\frac{1}{2} + \varepsilon})$ with $\varepsilon > 0$, then:

$$\Delta \left[\frac{D_\sigma^\times}{D_\sigma^\times} \bmod q, U(R_q^\times) \right] \leq q^{-\Omega(\varepsilon \cdot n)}.$$

- If $f \leftrightarrow D_\sigma^\times$, then $\|f\| \leq \sigma\sqrt{n}$, with overwhelming probability.
- We don't get uniformity in R_q but only in R_q^\times .
- R-LWE is still hard if h is restricted to $U(R_q^\times)$.

Proof: see proceedings of Eurocrypt'11.

Making $h = g/f$ statistically close to uniform

Our main technical contribution

If $\sigma = \tilde{\Omega}(n \cdot q^{\frac{1}{2} + \varepsilon})$ with $\varepsilon > 0$, then:

$$\Delta \left[\frac{D_\sigma^\times}{D_\sigma^\times} \bmod q, U(R_q^\times) \right] \leq q^{-\Omega(\varepsilon \cdot n)}.$$

- If $f \leftarrow D_\sigma^\times$, then $\|f\| \leq \sigma\sqrt{n}$, with overwhelming probability.
- We don't get uniformity in R_q but only in R_q^\times .
- R-LWE is still hard if h is restricted to $U(R_q^\times)$.

Proof: see proceedings of Eurocrypt'11.

Outline of the talk

- 1- Polynomial rings, NTRU and R-LWE.
- 2- Modifying NTRUencrypt to make it IND-CPA.
- 3- Modifying NTRUSign.

What's the interest of this result?

What we prove:

- There is a variant of NTRUencrypt that is provably IND-CPA under the assumption that $\mathcal{P}oly(n)$ -Ideal-SVP is hard to solve.
- It's **asymptotically** as efficient as the original NTRUencrypt.

What's the interest of this result?

What we prove:

- There is a variant of NTRUEncrypt that is provably IND-CPA under the assumption that $\mathcal{P}oly(n)$ -Ideal-SVP is hard to solve.
- It's **asymptotically** as efficient as the original NTRUEncrypt.

It does not mean we should blindly move to the provable variant:
It is most likely less practical.

What's the interest of this result?

What we prove:

- There is a variant of NTRUEncrypt that is provably IND-CPA under the assumption that $\mathcal{P}oly(n)$ -Ideal-SVP is hard to solve.
- It's **asymptotically** as efficient as the original NTRUEncrypt.

It does not mean we should blindly move to the provable variant: It is most likely less practical.

What it means:

- The general design of NTRUEncrypt is sound.
- It hints that we could
 - replace hs by $hs + e$, to thwart trivial CP attacks.
 - take less small coeffs for f, g, s, e , to improve security.

Work in progress and open problems

- ✓ A provably secure variant of NTRUSign.
- ✓ A provably IND-CCA2 variant of NTRUEncrypt.

- What about practice?
 - Which optimisations do not lower security?
 - What are the limits of the best known practical attacks? How do we extrapolate these limits to reach given security levels?
- Is $\mathcal{P}oly(n)$ -Ideal-SVP really so hard?
- Can we prove secure variants that are closer to the original design? E.g.:

Prove that $g/f \simeq^c U(R_q)$ for very small $f, g \in R_q$.

Work in progress and open problems

- ✓ A provably secure variant of NTRUSign.
- ✓ A provably IND-CCA2 variant of NTRUEncrypt.
- What about practice?
 - Which optimisations do not lower security?
 - What are the limits of the best known practical attacks? How do we extrapolate these limits to reach given security levels?
- Is $\mathcal{P}oly(n)$ -Ideal-SVP really so hard?
- Can we prove secure variants that are closer to the original design? E.g.:

Prove that $g/f \simeq^c U(R_q)$ for very small $f, g \in R_q$.

Work in progress and open problems

- ✓ A provably secure variant of NTRUSign.
- ✓ A provably IND-CCA2 variant of NTRUencrypt.

- What about practice?
 - Which optimisations do not lower security?
 - What are the limits of the best known practical attacks? How do we extrapolate these limits to reach given security levels?
- Is $\mathcal{P}oly(n)$ -Ideal-SVP really so hard?
- Can we prove secure variants that are closer to the original design? E.g.:

Prove that $g/f \simeq^c U(R_q)$ for very small $f, g \in R_q$.

Work in progress and open problems

- ✓ A provably secure variant of NTRUSign.
- ✓ A provably IND-CCA2 variant of NTRUencrypt.

- What about practice?
 - Which optimisations do not lower security?
 - What are the limits of the best known practical attacks? How do we extrapolate these limits to reach given security levels?
- Is $\mathcal{P}oly(n)$ -Ideal-SVP really so hard?
- Can we prove secure variants that are closer to the original design? E.g.:

Prove that $g/f \simeq^c U(R_q)$ for very small $f, g \in R_q$.

Quantifying the security of lattice-based cryptosystems in practice

Joop van de Pol

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB
United Kingdom.

January 12, 2012

Outline

Introduction

Lattice problems

Reduction algorithms

Predicting lattice reduction

Estimating security

Better key sizes

Better security estimates

Introduction

Lattice-based cryptography (1996-1997):

- ▶ Ajtai-Dwork
- ▶ Goldreich-Goldwasser-Halevi
- ▶ NTRU

Introduction

Lattice-based cryptography (1996-1997):

- ▶ Ajtai-Dwork (broken in 1998)
- ▶ Goldreich-Goldwasser-Halevi (broken in 1999)
- ▶ NTRU

Introduction

Lattice-based cryptography (1996-1997):

- ▶ Ajtai-Dwork (broken in 1998)
- ▶ Goldreich-Goldwasser-Halevi (broken in 1999)
- ▶ NTRU

Why were they broken?

Introduction

Lattice-based cryptography (1996-1997):

- ▶ Ajtai-Dwork (broken in 1998)
- ▶ Goldreich-Goldwasser-Halevi (broken in 1999)
- ▶ NTRU

Why were they broken?

- ▶ Low lattice dimension to maintain efficiency
- ▶ Attacks surprisingly strong in low dimensions

Introduction

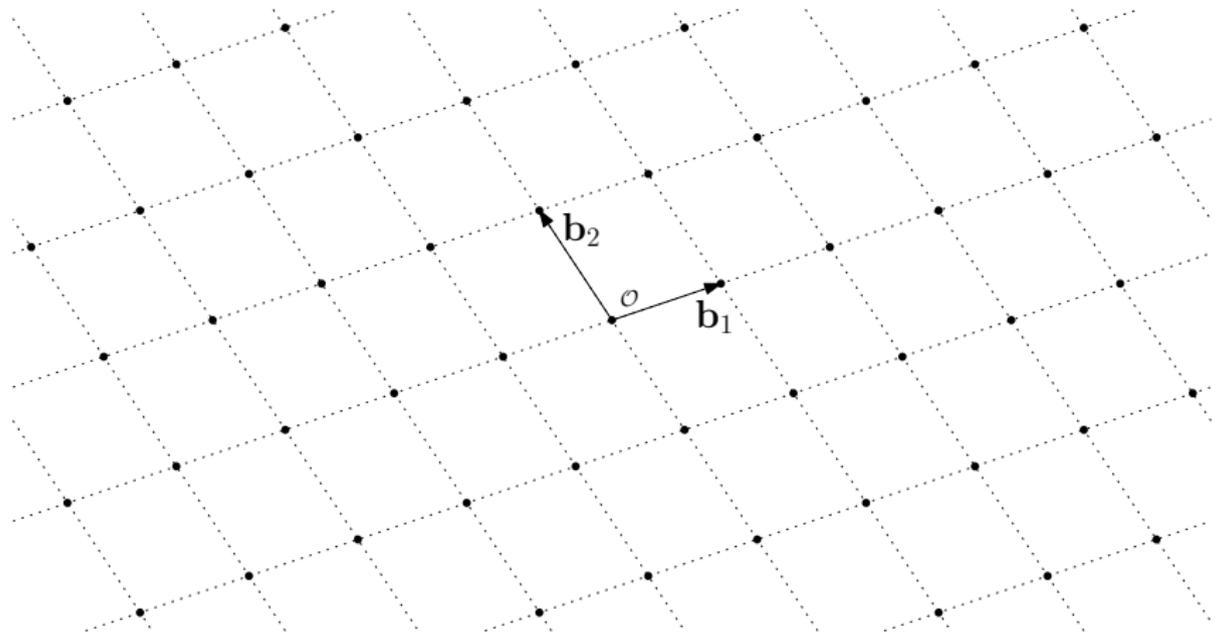
Lattice-based cryptography (1996-1997):

- ▶ Ajtai-Dwork (broken in 1998)
- ▶ Goldreich-Goldwasser-Halevi (broken in 1999)
- ▶ NTRU

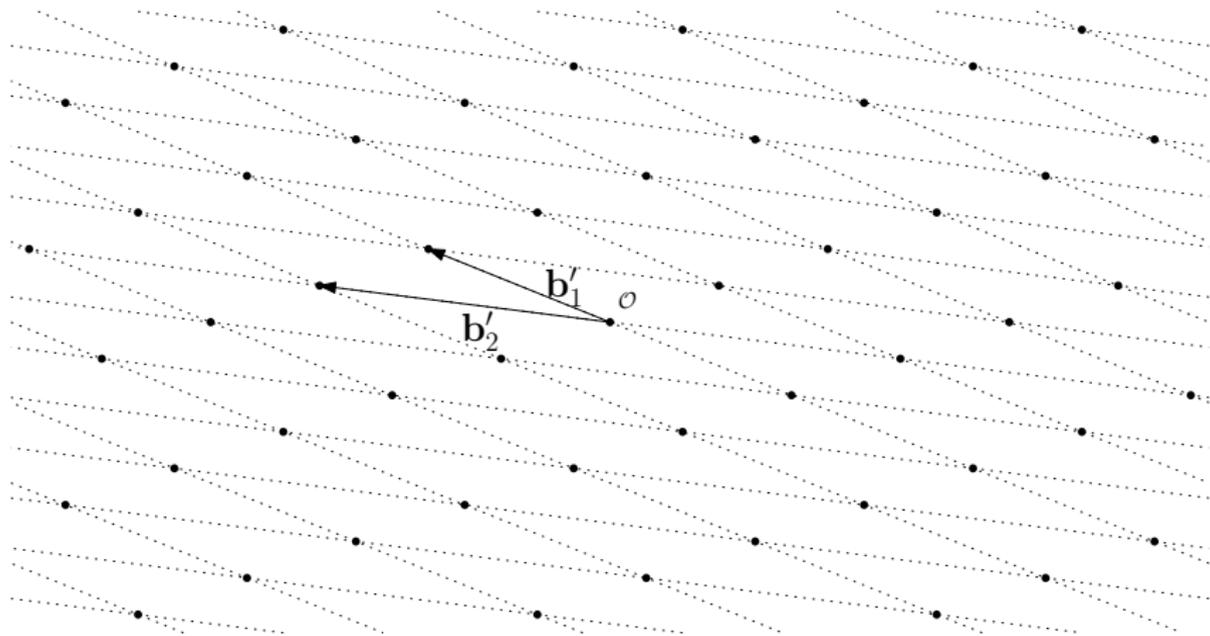
Why were they broken?

- ▶ Low lattice dimension to maintain efficiency
- ▶ Attacks surprisingly strong in low dimensions

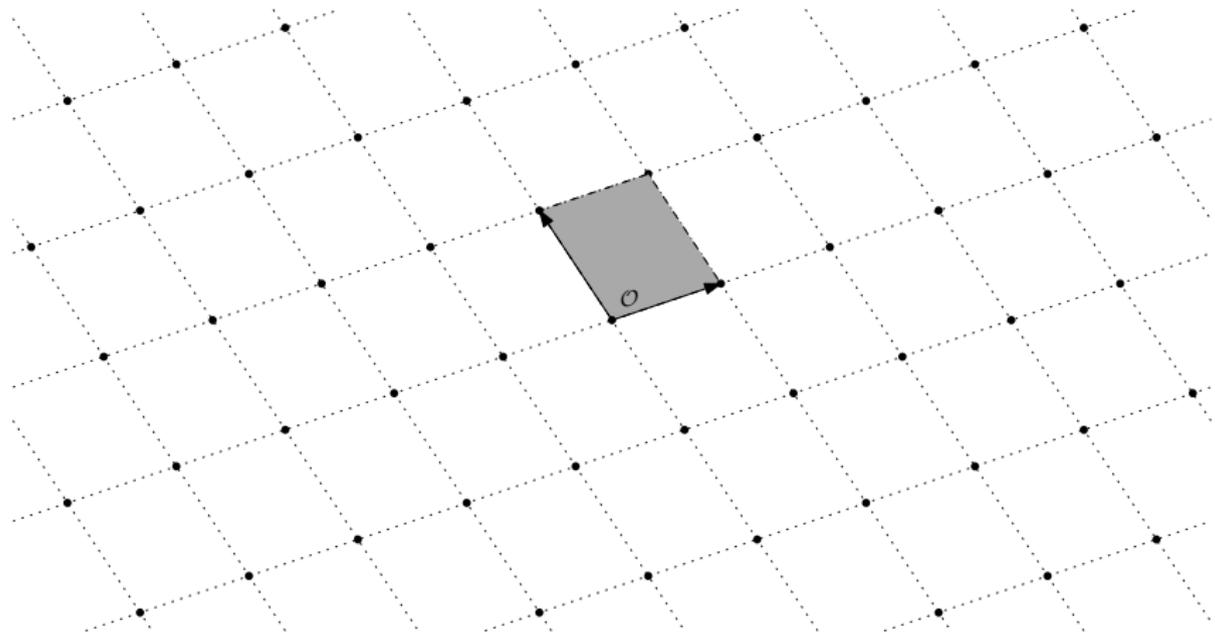
This presentation: How strong are these attacks?



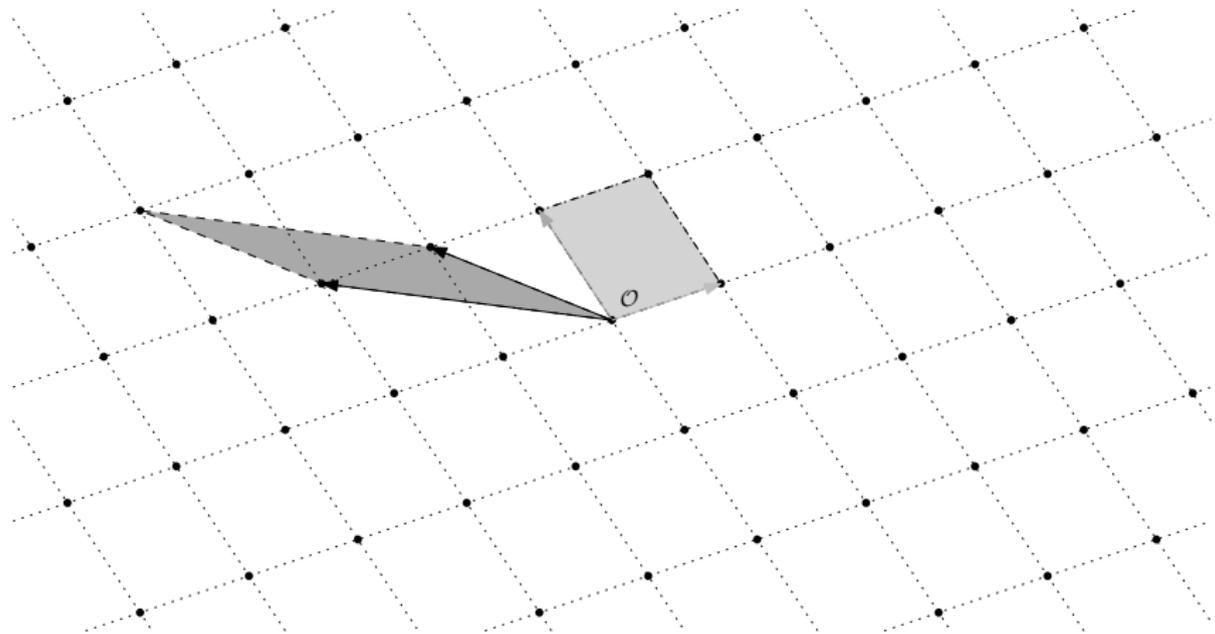
Lattices are described by a basis $\mathbf{b}_1, \dots, \mathbf{b}_n$.



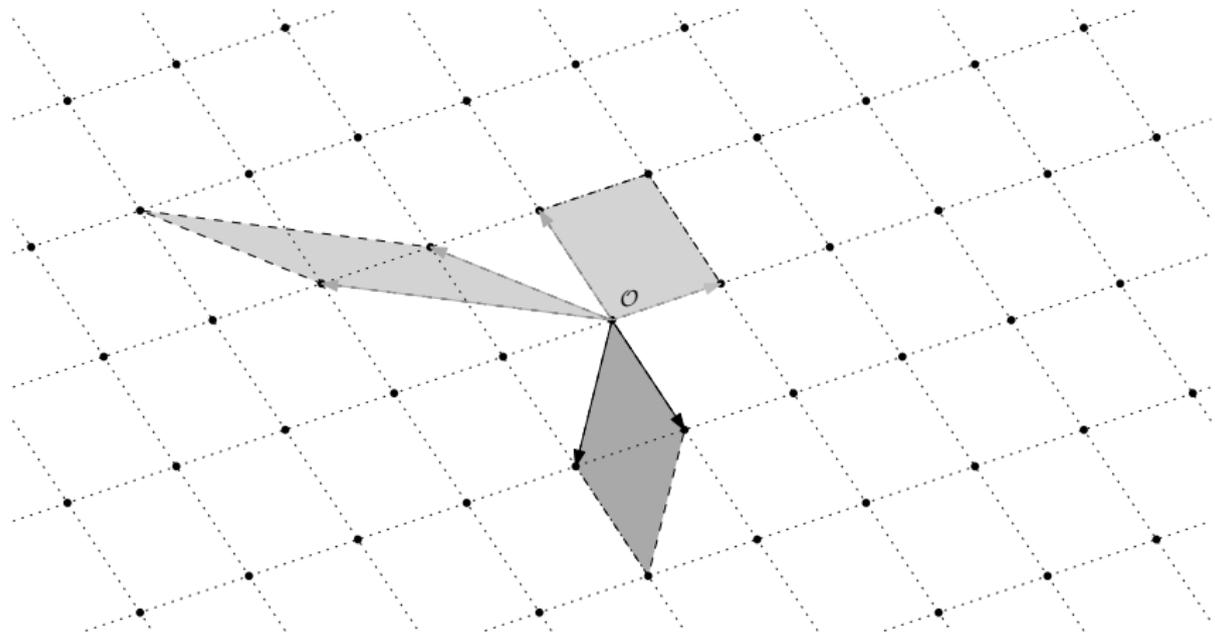
Lattices are described by a basis $\mathbf{b}_1, \dots, \mathbf{b}_n$. This basis is not unique.



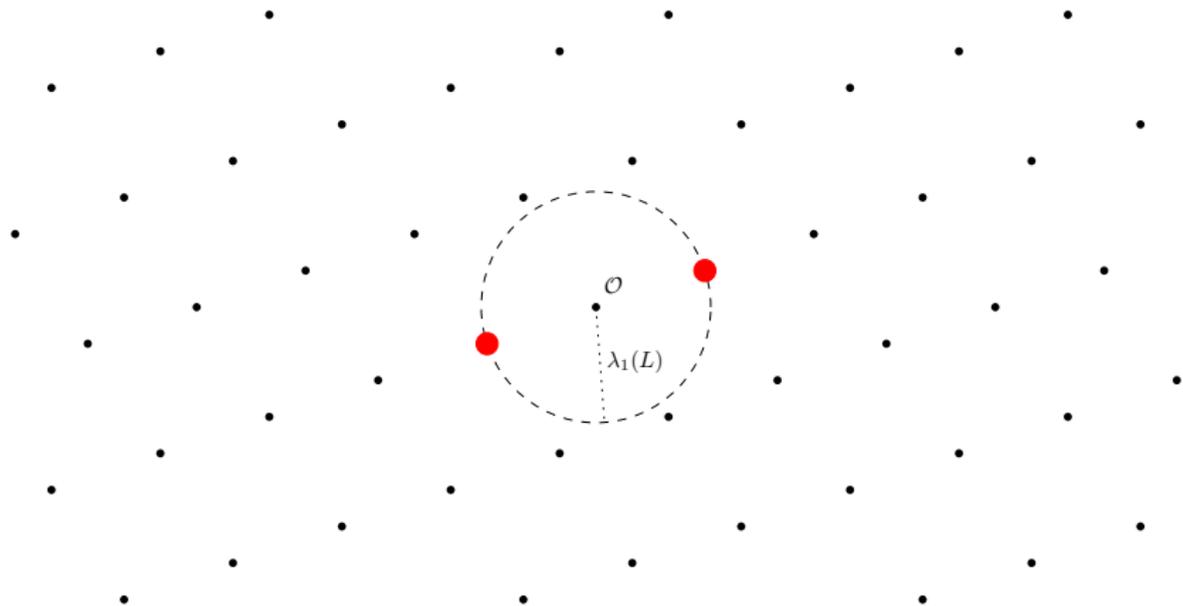
Lattices are described by a basis $\mathbf{b}_1, \dots, \mathbf{b}_n$. This basis is not unique. The volume $\text{vol}(L)$ is independent of the chosen basis.



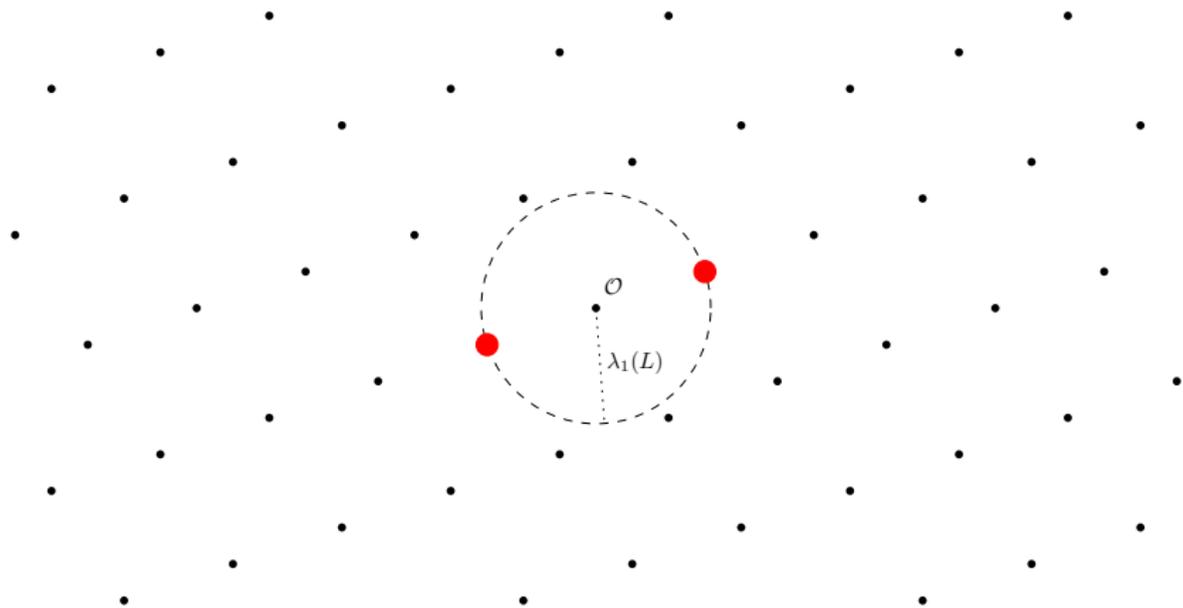
Lattices are described by a basis $\mathbf{b}_1, \dots, \mathbf{b}_n$. This basis is not unique. The volume $\text{vol}(L)$ is independent of the chosen basis.



Lattices are described by a basis $\mathbf{b}_1, \dots, \mathbf{b}_n$. This basis is not unique. The volume $\text{vol}(L)$ is independent of the chosen basis.



Lattices are described by a basis $\mathbf{b}_1, \dots, \mathbf{b}_n$. This basis is not unique. The volume $\text{vol}(L)$ is independent of the chosen basis. $\lambda_1(L)$ denotes the length of a shortest non-zero lattice vector.



Lattices are described by a basis $\mathbf{b}_1, \dots, \mathbf{b}_n$. This basis is not unique. The volume $\text{vol}(L)$ is independent of the chosen basis. $\lambda_1(L)$ denotes the length of a shortest non-zero lattice vector.

Hermite: $\lambda_1(L) \leq \sqrt{\gamma_n} \cdot \text{vol}(L)^{1/n}$

Lattice problems

SVP-type

CVP-type

Lattice problems

SVP-type:

- ▶ Shortest Vector Problem (SVP)
- ▶ Unique Shortest Vector Problem (uSVP)
- ▶ Small Integer Solutions (SIS)

CVP-type

Lattice problems

SVP-type:

- ▶ Shortest Vector Problem (SVP)
- ▶ Unique Shortest Vector Problem (uSVP)
- ▶ Small Integer Solutions (SIS)

CVP-type:

- ▶ Closest Vector Problem (CVP)
- ▶ Bounded Distance Decoding (BDD)
- ▶ Learning With Errors (LWE)

Lattice problems

SVP-type:

- ▶ Shortest Vector Problem (SVP)
- ▶ Unique Shortest Vector Problem (uSVP)
- ▶ Small Integer Solutions (SIS)

CVP-type:

- ▶ Closest Vector Problem (CVP)
- ▶ Bounded Distance Decoding (BDD)
- ▶ Learning With Errors (LWE)

In cryptanalysis:

- ▶ CVP-type solved using Babai's methods.
- ▶ CVP-type \rightarrow SVP-type (heuristically).
- ▶ SVP-type solved using basis reduction.

Lattice problems

SVP-type:

- ▶ Shortest Vector Problem (SVP)
- ▶ Unique Shortest Vector Problem (uSVP)
- ▶ Small Integer Solutions (SIS)

CVP-type:

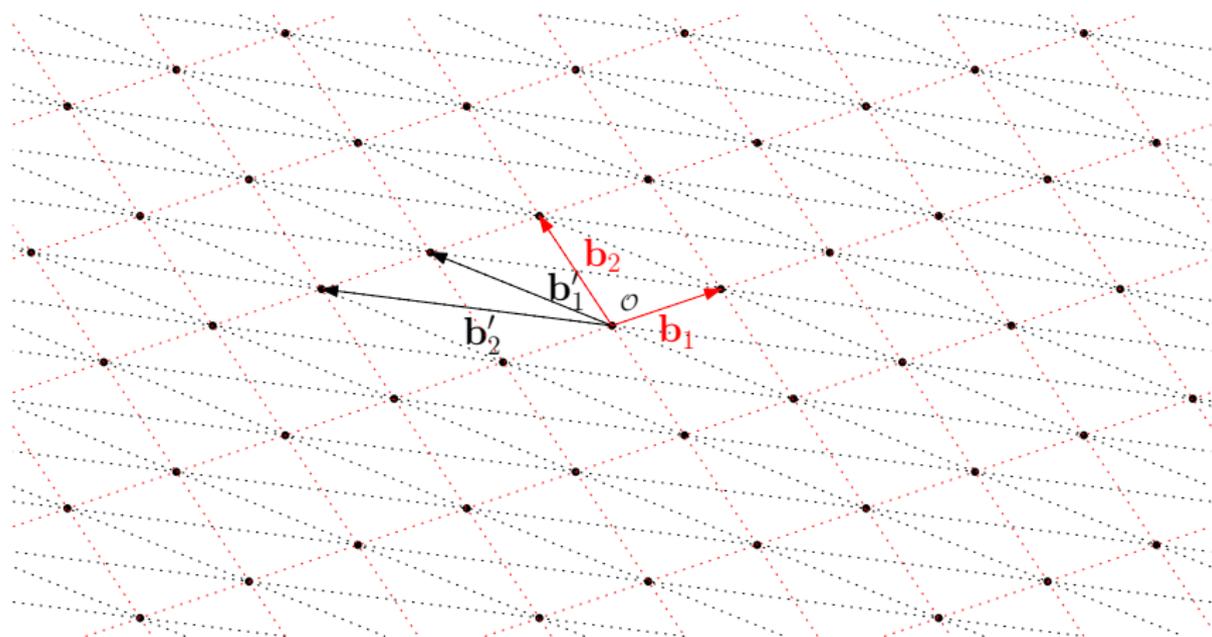
- ▶ Closest Vector Problem (CVP)
- ▶ Bounded Distance Decoding (BDD)
- ▶ Learning With Errors (LWE)

In cryptanalysis:

- ▶ CVP-type solved using Babai's methods.
- ▶ CVP-type \rightarrow SVP-type (heuristically).
- ▶ SVP-type solved using basis reduction.

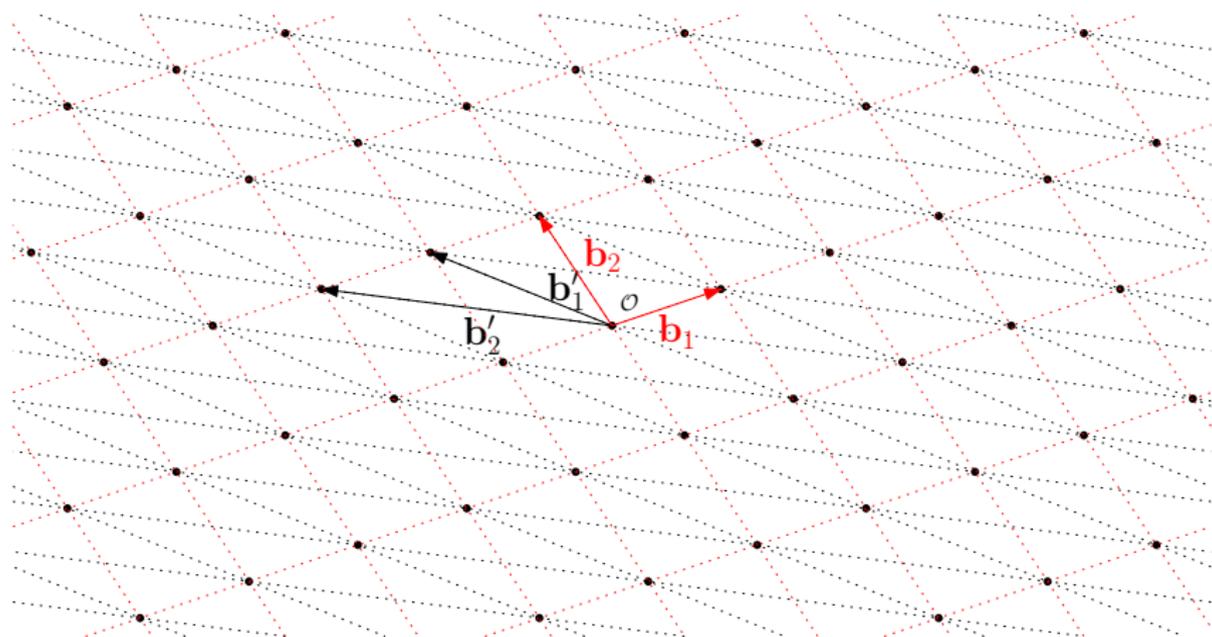
For practical analysis: Hermite Shortest Vector Problem (HSVP).

Basis reduction



A lattice has good and bad bases.

Basis reduction



A lattice has good and bad bases. Goal of basis reduction: go from bad to good basis.

Algorithms

The LLL algorithm, named for its creators Lenstra, Lenstra and Lovász:

- ▶ First basis reduction algorithm to run in polynomial time.
- ▶ Works by swapping two basis vectors in each stage and performing reduction.
- ▶ Finds a basis containing a short vector \mathbf{u} such that $\|\mathbf{u}\| \leq (4/3)^{(n-1)/4} \text{vol}(L)^{1/n}$.

Algorithms

The LLL algorithm, named for its creators Lenstra, Lenstra and Lovász:

- ▶ First basis reduction algorithm to run in polynomial time.
- ▶ Works by swapping two basis vectors in each stage and performing reduction.
- ▶ Finds a basis containing a short vector \mathbf{u} such that $\|\mathbf{u}\| \leq (4/3)^{(n-1)/4} \text{vol}(L)^{1/n}$.

Full enumeration of short vectors:

- ▶ Constructs a tree that contains all lattice vectors of norm $\leq R$ and searches this tree for a shortest non-zero vector.
- ▶ Finds an exact shortest vector, but runs in exponential time.

Algorithms

The Block-Korkine-Zolotarev algorithm (BKZ) combines these:

- ▶ Works on a block of β basis vectors at a time.
- ▶ Performs full enumeration on a lattice constructed with these β vectors.
- ▶ Finds a basis containing a short vector \mathbf{u} such that $\|\mathbf{u}\| \leq \sqrt{\gamma}^{\beta^{1+(n-1)/(\beta-1)}} \text{vol}(L)^{1/n}$.
- ▶ No good upper bound for the running time complexity.
- ▶ Outperforms other algorithms in practice.

Algorithms

The Block-Korkine-Zolotarev algorithm (BKZ) combines these:

- ▶ Works on a block of β basis vectors at a time.
- ▶ Performs full enumeration on a lattice constructed with these β vectors.
- ▶ Finds a basis containing a short vector \mathbf{u} such that $\|\mathbf{u}\| \leq \sqrt{\gamma}^{\beta^{1+(n-1)/(\beta-1)}} \text{vol}(L)^{1/n}$.
- ▶ No good upper bound for the running time complexity.
- ▶ Outperforms other algorithms in practice.

Theory only gives us upper bounds on the lengths of short vectors.

Predicting lattice reduction

Work by Gama and Nguyen (Eurocrypt 2008). Goal: perform experiments with reduction algorithms on random lattices to analyze their behavior. Their setup:

- ▶ Test the performance of LLL and BKZ on HSVP, aSVP and uSVP.
- ▶ Random lattices due to Goldstein and Mayer.
- ▶ Random bases heuristically chosen.

Predicting lattice reduction

Work by Gama and Nguyen (Eurocrypt 2008). Goal: perform experiments with reduction algorithms on random lattices to analyze their behavior. Their setup:

- ▶ Test the performance of LLL and BKZ on HSVP, aSVP and uSVP.
- ▶ Random lattices due to Goldstein and Mayer.
- ▶ Random bases heuristically chosen.

Observations:

- ▶ Hermite factor roughly behaves like $\gamma = \delta^n$.
- ▶ Base δ much smaller than expected from theoretical upper bounds
- ▶ BKZ impractical for block sizes $\beta \geq 25$.
- ▶ No worst case lattices, just bases.

Predicting lattice reduction

Parameter δ : 'root Hermite factor'. $\|\mathbf{u}\| = \delta^n \text{vol}(L)^{1/n}$. Theoretically:

- ▶ LLL: $\|\mathbf{u}\|/\text{vol}(L)^{1/n} \leq (4/3)^{(n-1)/4} \approx 1.07^n$.
- ▶ BKZ-20: $\|\mathbf{u}\|/\text{vol}(L)^{1/n} \leq \sqrt{\gamma_{20}}^{1+(n-1)/(19)} \approx 1.034^n$.

In practice, however, $\delta \approx 1.02$ for LLL and $\delta \approx 1.01$ for BKZ-20.

Predicting lattice reduction

Parameter δ : 'root Hermite factor'. $\|\mathbf{u}\| = \delta^n \text{vol}(L)^{1/n}$. Theoretically:

- ▶ LLL: $\|\mathbf{u}\|/\text{vol}(L)^{1/n} \leq (4/3)^{(n-1)/4} \approx 1.07^n$.
- ▶ BKZ-20: $\|\mathbf{u}\|/\text{vol}(L)^{1/n} \leq \sqrt{\gamma_{20}}^{1+(n-1)/(19)} \approx 1.034^n$.

In practice, however, $\delta \approx 1.02$ for LLL and $\delta \approx 1.01$ for BKZ-20.

Conclusions:

- ▶ For dimension $n \leq 450$, lattice reduction 'easy', since $\delta^n \lesssim n$.
- ▶ Hermite factor δ^n not yet reachable for $\delta = 1.005$.

Predicting lattice reduction

Parameter δ : 'root Hermite factor'. $\|\mathbf{u}\| = \delta^n \text{vol}(L)^{1/n}$. Theoretically:

- ▶ LLL: $\|\mathbf{u}\|/\text{vol}(L)^{1/n} \leq (4/3)^{(n-1)/4} \approx 1.07^n$.
- ▶ BKZ-20: $\|\mathbf{u}\|/\text{vol}(L)^{1/n} \leq \sqrt{\gamma_{20}}^{1+(n-1)/(19)} \approx 1.034^n$.

In practice, however, $\delta \approx 1.02$ for LLL and $\delta \approx 1.01$ for BKZ-20.

Conclusions:

- ▶ For dimension $n \leq 450$, lattice reduction 'easy', since $\delta^n \lesssim n$.
- ▶ Hermite factor δ^n not yet reachable for $\delta = 1.005$.

Remarks on results:

- ▶ Not aimed at cryptography.
- ▶ Does not give relation between δ and effort.
- ▶ Experimental results for lattices of dimension ≤ 200 .
- ▶ Not much information on the influence of dimension.

Estimating security

Work by Rückert and Schneider. Goal: adapt the approach of Gama and Nguyen to cryptographic applications and estimate their security in bits. Assumptions:

- ▶ Attacker model due to Lenstra and Verheul.
- ▶ Best way to attack systems is basis reduction (LWE \rightarrow SIS \rightarrow HSVP).
- ▶ Parameter δ may vary: it depends on the capabilities of the attacker.

Estimating security

Work by Rückert and Schneider. Goal: adapt the approach of Gama and Nguyen to cryptographic applications and estimate their security in bits. Assumptions:

- ▶ Attacker model due to Lenstra and Verheul.
- ▶ Best way to attack systems is basis reduction (LWE \rightarrow SIS \rightarrow HSVP).
- ▶ Parameter δ may vary: it depends on the capabilities of the attacker.

Conclusions:

- ▶ Parameter δ has more influence on the hardness than the dimension.
- ▶ Effort can be written as function of δ .
- ▶ Use model to decide if the attacker can reach δ .

Estimating security

Year	2010	2020	2030	2040	2050
Bit security	75	82	88	95	102
Hacker	1.01177	1.00965	1.00808	1.00702	1.00621
Lenstra	1.00919	1.00785	1.00678	1.00602	1.00541
Int. Agency	1.00799	1.00695	1.00610	1.00548	1.00497

Year	2060	2070	2080	2090	2100
Bit security	108	115	122	128	135
Hacker	1.00552	1.00501	1.00458	1.00419	1.00389
Lenstra	1.00488	1.00447	1.00413	1.00381	1.00356
Int. Agency	1.00452	1.00417	1.00387	1.00359	1.00336

Table: Values of δ predicted to be infeasible to break for the attackers.

Estimating security

Year	2010	2020	2030	2040	2050
Bit security	75	82	88	95	102
Hacker	1.01177	1.00965	1.00808	1.00702	1.00621
Lenstra	1.00919	1.00785	1.00678	1.00602	1.00541
Int. Agency	1.00799	1.00695	1.00610	1.00548	1.00497

Year	2060	2070	2080	2090	2100
Bit security	108	115	122	128	135
Hacker	1.00552	1.00501	1.00458	1.00419	1.00389
Lenstra	1.00488	1.00447	1.00413	1.00381	1.00356
Int. Agency	1.00452	1.00417	1.00387	1.00359	1.00336

Table: Values of δ predicted to be infeasible to break for the attackers.

Remarks on results:

- ▶ Experimental results for lattices of dimension ≤ 300
- ▶ Dimension discarded too easily.
- ▶ Relation to bit-security tenuous.

Better key sizes

Work by Lindner and Peikert (CT-RSA 2011). They propose several improvements to LWE-based encryption schemes and perform experiments to analyze their security:

- ▶ New LWE-based encryption scheme.
- ▶ New attack against LWE-based schemes.
- ▶ Perform experiments using both new and old attacks.

Better key sizes

Work by Lindner and Peikert (CT-RSA 2011). They propose several improvements to LWE-based encryption schemes and perform experiments to analyze their security:

- ▶ New LWE-based encryption scheme.
- ▶ New attack against LWE-based schemes.
- ▶ Perform experiments using both new and old attacks.

Results:

- ▶ Their new attack is faster than the old one.
- ▶ A basis of lesser quality (higher δ) suffices.
- ▶ They elect not to give symmetric bit security estimates.

Better key sizes

Parameters			Distinguish		Decode	
n	q	s	δ	$\lg(t)$	δ	$\lg(t)$
128	2053	6.77	1.0065	83	1.0089	32
192	4093	8.87	1.0045	168	1.0067	78
256	4093	8.35	1.0034	258	1.0052	132
320	4093	8.00	1.0027	353	1.0042	189

Table: Values of δ needed for and estimated running times of the attacks

Better key sizes

Parameters			Distinguish		Decode	
n	q	s	δ	$\lg(t)$	δ	$\lg(t)$
128	2053	6.77	1.0065	83	1.0089	32
192	4093	8.87	1.0045	168	1.0067	78
256	4093	8.35	1.0034	258	1.0052	132
320	4093	8.00	1.0027	353	1.0042	189

Table: Values of δ needed for and estimated running times of the attacks

Remarks on results:

- ▶ Only applicable to LWE-based encryption schemes.
- ▶ Experimental results for lattices of dimension ≈ 200 .
- ▶ Running time for decoding attack is estimation based on probabilities.

Better security estimates: BKZ 2.0

Work by Chen and Nguyen (Asiacrypt 2011). They made several improvements to BKZ and performed new experiments to update the results of 2008:

- ▶ Abort after a fixed number of rounds.
- ▶ Several improvements to enumeration.

Better security estimates: BKZ 2.0

Work by Chen and Nguyen (Asiacrypt 2011). They made several improvements to BKZ and performed new experiments to update the results of 2008:

- ▶ Abort after a fixed number of rounds.
- ▶ Several improvements to enumeration.

Results:

- ▶ Simulation to predict the behavior for higher block sizes.
- ▶ Upper and lower bounds on the cost of enumeration.
- ▶ Broke several lattice challenges ($1.0095 \leq \delta \leq 1.0099$).
- ▶ Revised security estimates.

Better security estimates

Hermite factor	1.01^n	1.009^n	1.008^n	1.007^n	1.006^n	1.005^n
Blocksize	85	106	133	168	216	286

Table: Simulated block sizes needed to achieve Hermite factors

Better security estimates

Hermite factor	1.01^n	1.009^n	1.008^n	1.007^n	1.006^n	1.005^n
Blocksize	85	106	133	168	216	286

Table: Simulated block sizes needed to achieve Hermite factors

Blocksize	100	120	140	160	180	200	250
lg(# nodes)	41.4	53.1	66.8	84.7	105.8	129.4	204.1

Table: Upper bound of the cost of enumeration

Better security estimates

Hermite factor	1.01^n	1.009^n	1.008^n	1.007^n	1.006^n	1.005^n
Blocksize	85	106	133	168	216	286

Table: Simulated block sizes needed to achieve Hermite factors

Blocksize	100	120	140	160	180	200	250
lg(# nodes)	41.4	53.1	66.8	84.7	105.8	129.4	204.1

Table: Upper bound of the cost of enumeration

Remarks on results:

- ▶ Bounds on cost of enumeration not tight.
- ▶ Dimension is a “second-order term”.
- ▶ No timings reported for challenges (dimension up to 800).

Summary

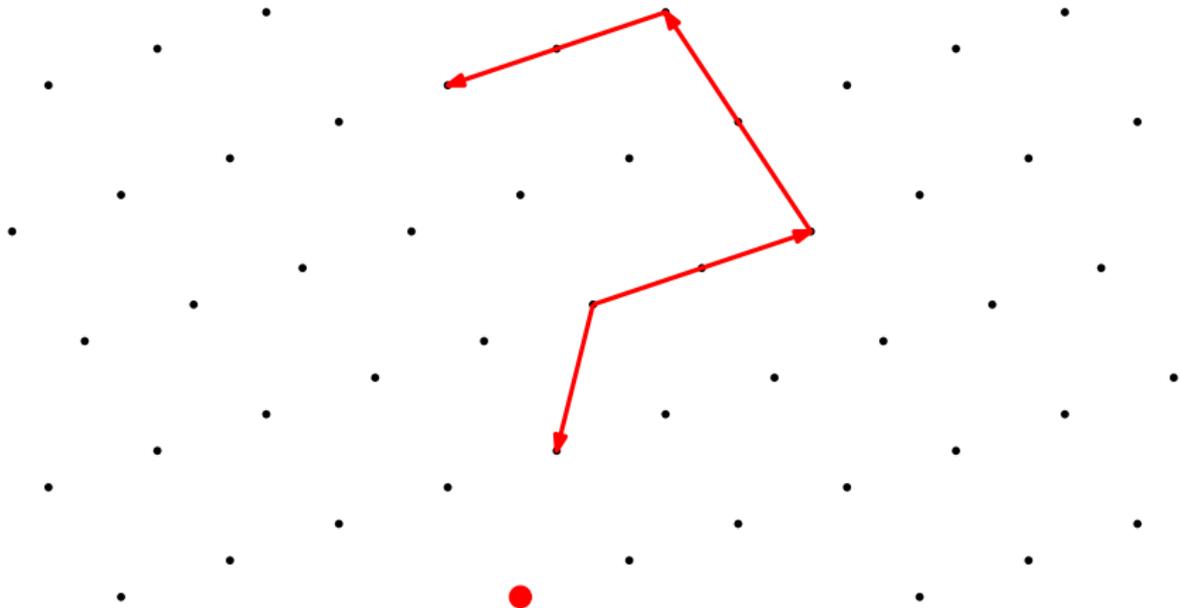
- ▶ Basis reduction algorithms are our main attack tools.
- ▶ We need to understand how they work in practice.
- ▶ We have some experimental results of this.
- ▶ Not everything explained by theory yet.

Summary

- ▶ Basis reduction algorithms are our main attack tools.
- ▶ We need to understand how they work in practice.
- ▶ We have some experimental results of this.
- ▶ Not everything explained by theory yet.

Future research:

- ▶ Dimension does not seem to matter so far; why?
- ▶ What about ideal lattices? Is the security the same?
- ▶ What about other methods for enumeration (sieving)?
- ▶ Improve basis reduction algorithms.



A Generic Variant of NIST's KAS2 Key Agreement Protocol

Sanjit Chatterjee

(Joint work with Alfred Menezes and Berkant Ustaoglu)

Indian Institute of Science

KAS2 Key Agreement Protocol

- ▶ NIST's [SP 800-56B](#) [2009] standardizes several RSA-based key establishment schemes.
- ▶ *KAS2-bilateral-confirmation* ([KAS2](#)) is a three-pass protocol that offers key confirmation.
- ▶ SP 800-56B describes three other variants of KAS2 and also a two-pass protocol KAS1.
- ▶ KAS2-bilateral-confirmation protocol offers the most security attributes of the different KAS2 variants.
 - ▶ Most likely to be deployed in applications that wish to be compliant with SP 800-56B.
 - ▶ We focus on this particular version of KAS2.

Our Work

- ▶ A generic three-pass key agreement protocol based on trapdoor one-way function family.
- ▶ A security model for the generic protocol.
- ▶ Specific instantiations:
 1. RSA setting: yields the **KAS2** protocol.
 2. Discrete log setting: yields a new protocol **DH2**.
 3. Hybrid setting: combines RSA and dlog setting to get a new a protocol called **KAS2-DH2**.
- ▶ Reductionist security argument in the RSA and discrete log setting.

A Trapdoor One-way Function Family

- ▶ Let $f : Z \rightarrow Z$ is from a family of trapdoor one-way functions.
 1. f is bijective.
 2. \exists an efficient algorithm that outputs $(X, f(X))$ with $X \in_R Z$.
 3. Given $f(X)$ for $X \in_R Z$, it is **infeasible** to determine X .
 4. Given a trapdoor T_f , one can **efficiently** compute X given $f(X)$ for $X \in_R Z$.

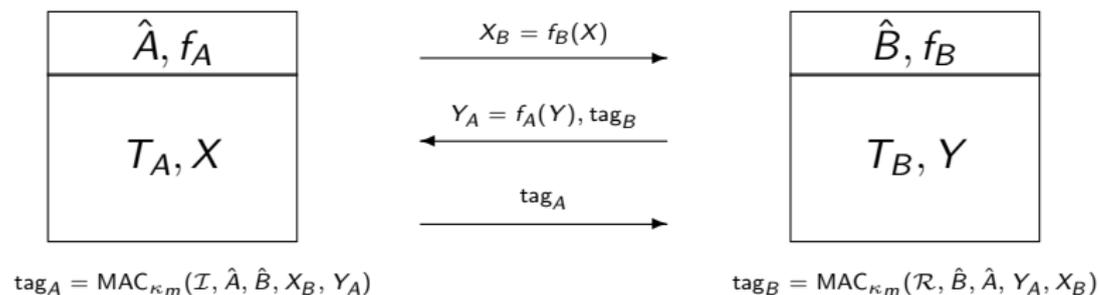
A Trapdoor One-way Function Family

- ▶ Let $f : Z \rightarrow Z$ is from a family of trapdoor one-way functions.
 1. f is bijective.
 2. \exists an efficient algorithm that outputs $(X, f(X))$ with $X \in_R Z$.
 3. Given $f(X)$ for $X \in_R Z$, it is **infeasible** to determine X .
 4. Given a trapdoor T_f , one can **efficiently** compute X given $f(X)$ for $X \in_R Z$.
- ▶ $f_{N,e} : \mathbb{Z}_N \rightarrow \mathbb{Z}_N$ defined as $f_{N,e}(m) = m^e \bmod N$.
 - ▶ (N, e) is an RSA public key.
 - ▶ The trapdoor is the RSA private key d .

A Trapdoor One-way Function Family

- ▶ Let $f : Z \rightarrow Z$ is from a family of trapdoor one-way functions.
 1. f is bijective.
 2. \exists an efficient algorithm that outputs $(X, f(X))$ with $X \in_R Z$.
 3. Given $f(X)$ for $X \in_R Z$, it is **infeasible** to determine X .
 4. Given a trapdoor T_f , one can **efficiently** compute X given $f(X)$ for $X \in_R Z$.
- ▶ $f_{N,e} : \mathbb{Z}_N \rightarrow \mathbb{Z}_N$ defined as $f_{N,e}(m) = m^e \bmod N$.
 - ▶ (N, e) is an RSA public key.
 - ▶ The trapdoor is the RSA private key d .
- ▶ $\mathbb{G} = \langle g \rangle$: cyclic group of prime order q .
 - ▶ Let $a \in_R \mathbb{Z}_q$, $A = g^a$.
 - ▶ $f_A : \mathbb{G} \rightarrow \mathbb{G}$ defined as $f(g^x) = A^x$ is a trapdoor one-way function with trapdoor a .
 - ▶ *Diffie-Hellman division (DHD) problem*: given $g, A^x, A \in \mathbb{G}$, determine g^x .

A Generic Protocol



$$(\kappa_m, \kappa) = H(X, Y, \hat{A}, \hat{B}, X_B, Y_A)$$

- ▶ \hat{A} 's static public key is a trapdoor function $f_A : Z_A \rightarrow Z_A$, and the corresponding trapdoor data T_A is her static private key.
- ▶ \hat{B} 's static public key is the trapdoor function $f_B : Z_B \rightarrow Z_B$ and the corresponding trapdoor data T_B is his static private key.
- ▶ MAC is a secure message authentication code algorithm.

Security Model

- ▶ Static private key of a party is used as a **trapdoor** to extract the other party's ephemeral private key.
- ▶ Session key is the **hash** of individual ephemeral private keys (and some public information).
- ▶ We follow the eCK model but take into consideration above features of the protocol.
 - ▶ Definition of fresh session is more **restrictive** compared to the eCK model.
 - ▶ The model incorporates resistance to **KCI** attacks (not covered in CK model).
 - ▶ Also covers **half-forward secrecy** – security of a session key is preserved even if adversary (\mathcal{M}) learns the static key of one of the parties.

Matching Sessions

- ▶ Let $s = (\hat{A}, \hat{B}, role, *, *)$, where $role \in \{\mathcal{I}, \mathcal{R}\}$, \hat{A} is the *owner* and \hat{B} is the *peer* of session s .
- ▶ Let s be a session with complete session identifier $(\hat{A}, \hat{B}, role_A, f_B(X), f_A(Y))$ where $role_A \in \{\mathcal{I}, \mathcal{R}\}$.
- ▶ A session s^* with session identifier $(\hat{C}, \hat{D}, role_C, f_D(U), f_C(V))$, where $role_C \in \{\mathcal{I}, \mathcal{R}\}$, is *matching* to s if
 1. $\hat{A} = \hat{D}$ and $\hat{B} = \hat{C}$,
 2. $role_A \neq role_C$,
 3. $f_B(X) = f_C(V)$ and $f_A(Y) = f_D(U)$.
- ▶ A session s with incomplete session identifier $(\hat{A}, \hat{B}, \mathcal{I}, f_B(X))$ is matching to any session $s = (\hat{C}, \hat{D}, \mathcal{R}, f_D(U), f_C(V))$ with $\hat{A} = \hat{D}$, $\hat{B} = \hat{C}$ and $f_B(X) = f_C(V)$; s^* is also matching to s .

Adversary

- ▶ The adversary \mathcal{M} controls *all* communications but does not have immediate access to a party's private information.
- ▶ To capture possible leakage of private information \mathcal{M} is allowed to make the following queries:
 1. *StaticKeyReveal*(\hat{A})
 2. *EphemeralKeyReveal*(s)
 3. *SessionKeyReveal*(s)
 4. *EstablishParty*(\hat{A}, A)
 5. *Expire*(s)
- ▶ Parties established by \mathcal{M} using *EstablishParty* are called *corrupted*, parties not corrupted are *honest*.

Fresh Session

- ▶ s : id of a completed session, owned by \hat{A} with peer \hat{B} , both honest.
- ▶ s^* : id of the matching session of s (if exists).
- ▶ s is *fresh* if none of the following conditions hold:
 1. \mathcal{M} issued $SessionKeyReveal(s)$ or $SessionKeyReveal(s^*)$ (if s^* exists).
 2. s^* exists and \mathcal{M} issued one of the following:
 - 2.1 Both $StaticKeyReveal(\hat{A})$ and $EphemeralKeyReveal(s)$.
 - 2.2 Both $StaticKeyReveal(\hat{B})$ and $EphemeralKeyReveal(s^*)$.
 - 2.3 Both $StaticKeyReveal(\hat{A})$ and $StaticKeyReveal(\hat{B})$.
 - 2.4 Both $EphemeralKeyReveal(s)$ and $EphemeralKeyReveal(s^*)$.
 3. s^* does not exist and \mathcal{M} issued one of the following:
 - 3.1 $EphemeralKeyReveal(s)$.
 - 3.2 $StaticKeyReveal(\hat{B})$ before $Expire(s)$.

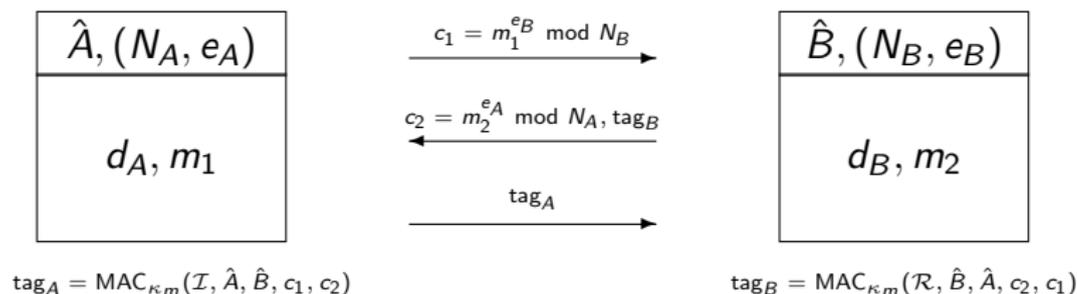
Security of Key Agreement

- ▶ \mathcal{M} is allowed to make a special query $Test(s)$ to a **fresh** session s .
 - ▶ \mathcal{M} gets with equal probability either the session key held by s or a random key.
 - ▶ \mathcal{M} wins if it can guess correctly whether the key is random or not.
 - ▶ \mathcal{M} can continue interacting with the parties after issuing the $Test$ query, but the test session must remain fresh throughout \mathcal{M} 's experiment.

Security of Key Agreement

- ▶ \mathcal{M} is allowed to make a special query $Test(s)$ to a **fresh** session s .
 - ▶ \mathcal{M} gets with equal probability either the session key held by s or a random key.
 - ▶ \mathcal{M} wins if it can guess correctly whether the key is random or not.
 - ▶ \mathcal{M} can continue interacting with the parties after issuing the $Test$ query, but the test session must remain fresh throughout \mathcal{M} 's experiment.
- ▶ A key agreement protocol is **secure**:
 1. If two honest parties complete matching sessions then, except with negligible probability, they both compute the same session key.
 2. No polynomially bounded adversary \mathcal{M} can distinguish the session key of a fresh session from a randomly chosen session key with probability greater than $\frac{1}{2}$ plus a negligible fraction.

KAS2 Protocol



$$(\kappa_m, \kappa) = H(m_1, m_2, \hat{A}, \hat{B}, c_1, c_2)$$

- ▶ In SP 800-56B, H also takes input an integer keydatalen, a bit string AlgorithmID, and two optional strings SuppPubInfo and SuppPrivInfo.
- ▶ (c_1, c_2) are included in SuppPubInfo to simplify the security reduction.
- ▶ keydatalen, AlgorithmID and SuppPrivInfo are omitted as they are not relevant in security analysis.

Security of KAS2

- ▶ *RSA problem*: Determine $m \in [2, N - 2]$ such that $c \equiv m^e \pmod{N}$ given an RSA public key (N, e) and an integer $c \in_R [2, N - 2]$.
- ▶ *RSA assumption*: No polynomially-bounded algorithm can solve the RSA problem with non-negligible probability of success.
- ▶ *Security statement*: KAS2 protocol is secure assuming:
 1. RSA assumption holds;
 2. MAC scheme is secure
 3. H is a random oracle.

Security Argument

- ▶ H is a random function so \mathcal{M} has only two strategies to win with probability significantly greater than $\frac{1}{2}$:

Security Argument

- ▶ H is a random function so \mathcal{M} has only two strategies to win with probability significantly greater than $\frac{1}{2}$:
- ▶ **Strategy 1:** Induce two non-matching sessions to establish the same session key, set one as the test session, and issue a *SessionKeyReveal* query to the other.
 - ▶ But non-matching completed sessions produce **different** session keys except with negligible probability of H collisions!

Security Argument

- ▶ H is a random function so \mathcal{M} has only two strategies to win with probability significantly greater than $\frac{1}{2}$:
- ▶ **Strategy 1:** Induce two non-matching sessions to establish the same session key, set one as the test session, and issue a *SessionKeyReveal* query to the other.
 - ▶ But non-matching completed sessions produce **different** session keys except with negligible probability of H collisions!
- ▶ **Strategy 2:** Query oracle H with $(c_1^{d_B} \bmod N_B, c_2^{d_A} \bmod N_A, \hat{A}, \hat{B}, c_1, c_2)$ where test session is $(\hat{A}, \hat{B}, \mathcal{I}, c_1, c_2)$ or $(\hat{B}, \hat{A}, \mathcal{R}, c_2, c_1)$.
 - ▶ Construct \mathcal{S} that takes input an RSA challenge (N_V, e_V, c_V) , has access to a MAC oracle with unknown key κ_m and produces either a solution to the RSA challenge or a MAC forgery.

Intuitive idea

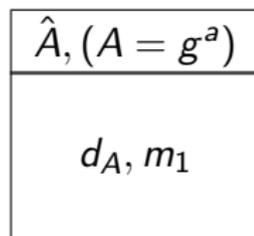
- ▶ s^t : test session; s^m : the matching session (if exists).
- ▶ Break-up \mathcal{M} 's success into two complementary events.
 1. E_1 : s^m exists and \mathcal{M} issues neither $StaticKeyReveal(\hat{A})$ nor $EphemeralKeyReveal(s^m)$.
 2. E_2 : either s^m does not exist, or s^m exists and \mathcal{M} issues $StaticKeyReveal(\hat{A})$ or $EphemeralKeyReveal(s^m)$.
- ▶ E_1 : \mathcal{S} sets the static public key of \hat{A} as (N_V, e_V) and the ephemeral public key of s^m as c_V .
- ▶ E_2 : \mathcal{S} sets the static public key of \hat{B} as (N_V, e_V) , the ephemeral public key of s^t as c_V and use the MAC oracle for the test session.
- ▶ Requires some ingenuity in programming the hash function for a proper simulation.

Discrete Log Setting

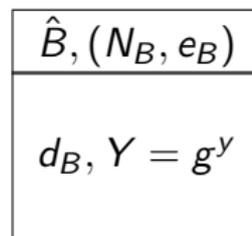
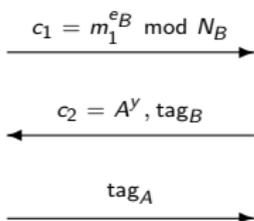
- ▶ Our generic protocol can be specialized to the discrete log setting to yield a new protocol called **DH2**.
 - ▶ Security is based on the Gap-DH assumption.
- ▶ In DH2, parties can use **different** groups (e.g., different elliptic curves).

The hybrid protocol

- ▶ The generic protocol also has a **hybrid** implementation.
 - ▶ One party can use an RSA key pair.
 - ▶ The other party can use a discrete log key pair.
 - ▶ Security is based on both RSA and Gap-DH assumptions.



$$\text{tag}_A = \text{MAC}_{\kappa_m}(\mathcal{I}, \hat{A}, \hat{B}, c_1, c_2)$$



$$\text{tag}_B = \text{MAC}_{\kappa_m}(\mathcal{R}, \hat{B}, \hat{A}, c_2, c_1)$$

$$(\kappa_m, \kappa) = H(m_1, Y, \hat{A}, \hat{B}, c_1, c_2)$$

Thank you for your attention!

Relaxing IND-CCA: Indistinguishability Against Chosen Ciphertext Verification Attack

Sumit Kumar Pandey

Indian Statistical Institute
Kolkata

January 14, 2012

Outline

- 1 Definitions
 - Encryption Scheme
 - IND-CPA
 - IND-CCA
 - IND-CCVA
- 2 Bleichenbacher's attack on PKCS#1
- 3 ElGamal Encryption Scheme
- 4 Cramer-Shoup light version
- 5 ElGamal-ElGamal Encryption Scheme
- 6 Generic Construction

Definition: Encryption Scheme

- **KG(1^λ)**: A probabilistic polynomial time algorithm which takes security parameter 1^λ as input and outputs a public-private key pair (PK, SK) .
- **ENC(m, PK)**: A probabilistic polynomial time algorithm which takes a message m and public key PK as input and returns ciphertext \mathcal{C} .
- **DEC(\mathcal{C}, SK, PK)**: A deterministic polynomial time algorithm which takes ciphertext \mathcal{C} , secret key SK and public key PK as input and returns a message m if \mathcal{C} is a valid ciphertext else \perp .

Definition: Encryption Scheme

- **KG(1^λ)**: A probabilistic polynomial time algorithm which takes security parameter 1^λ as input and outputs a public-private key pair (PK, SK) .
- **ENC(m, PK)**: A probabilistic polynomial time algorithm which takes a message m and public key PK as input and returns ciphertext \mathcal{C} .
- **DEC(\mathcal{C}, SK, PK)**: A deterministic polynomial time algorithm which takes ciphertext \mathcal{C} , secret key SK and public key PK as input and returns a message m if \mathcal{C} is a valid ciphertext else \perp .

For consistency, it is required that for all $(PK, SK) \leftarrow \text{KG}(1^\lambda)$ and all messages m , $m = \text{DEC}(\text{ENC}(m, PK), SK, PK)$.

Definition: IND-CPA

An encryption scheme S_{ENC} is said to be **IND-CPA (indistinguishable against chosen plaintext attack)** secure if no probabilistic polynomial time algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game:

Definition: IND-CPA

An encryption scheme S_{ENC} is said to be **IND-CPA (indistinguishable against chosen plaintext attack)** secure if no probabilistic polynomial time algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game:

Game $_{S_{ENC}, \mathcal{A}}^{IND-CPA}$

- $(PK, SK) \leftarrow \text{KG}(1^\lambda)$
- $(m_0, m_1, st) \leftarrow \mathcal{A}_1(PK)$
- $b \xleftarrow{R} \{0, 1\}$
- $y \leftarrow \text{ENC}(m_b, PK)$
- $b' \leftarrow \mathcal{A}_2(y, PK, st)$

The advantage of \mathcal{A} is defined as $\text{Adv}(\mathcal{A}) = |\Pr(b = b') - \frac{1}{2}|$

Definition: IND-CCA

An encryption scheme S_{ENC} is said to be **IND-CCA** (**indistinguishable against chosen ciphertext attack**) secure if no probabilistic polynomial time algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game:

Definition: IND-CCA

An encryption scheme S_{ENC} is said to be **IND-CCA** (**indistinguishable against chosen ciphertext attack**) secure if no probabilistic polynomial time algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game:

- *DecryptionOracle*(\mathcal{O}): Given a ciphertext \mathcal{C} , except the challenge ciphertext, the oracle returns $m \leftarrow \text{DEC}(\mathcal{C}, SK, PK)$.

Game $_{S_{ENC}, \mathcal{A}}^{\text{IND-CCA}}$

- $(PK, SK) \leftarrow \text{KG}(1^\lambda)$
- $(m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK)$
- $b \xleftarrow{R} \{0, 1\}$
- $y \leftarrow \text{ENC}(m_b, PK)$
- $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(y, PK, st)$

The advantage of \mathcal{A} is defined as $\text{Adv}(\mathcal{A}) = |\Pr(b = b') - \frac{1}{2}|$

Definition: IND-CCVA

An encryption scheme S_{ENC} is said to be **IND-CCVA (indistinguishable against chosen ciphertext verification attack)** secure if no probabilistic polynomial time algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game:

Definition: IND-CCVA

An encryption scheme S_{ENC} is said to be **IND-CCVA (indistinguishable against chosen ciphertext verification attack)** secure if no probabilistic polynomial time algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game:

- *ChosenCiphertextVerificationOracle*(\mathcal{O}): Given a ciphertext \mathcal{C} , the oracle returns 1 if \mathcal{C} is valid else returns 0.

Game $_{S_{ENC}, \mathcal{A}}^{IND-CCVA}$

- $(PK, SK) \leftarrow \text{KG}(1^\lambda)$
- $(m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK)$
- $b \xleftarrow{R} \{0, 1\}$
- $y \leftarrow \text{ENC}(m_b, PK)$
- $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(y, PK, st)$

The advantage of \mathcal{A} is defined as $\text{Adv}(\mathcal{A}) = |\Pr(b = b') - \frac{1}{2}|$

Trivial Conclusions

- 1 IND-CCVA secure encryption schemes are IND-CPA secure also.

IND-CCVA \rightarrow IND-CPA

- 2 IND-CCA secure encryption schemes are IND-CCVA secure also.

IND-CCA \rightarrow IND-CCVA

Does CCVA make sense?

PKCS#1

- **KG(1^λ):** Choose primes p, q ($4k$ bit each) and compute $n = pq$ (n is k byte number). Choose e, d , such that $ed \equiv 1 \pmod{\phi(n)}$. The public key, PK , is (n, e) and the secret key, SK , is (p, q, d) .
- **ENC(m, PK):** A data block D , consisting of $|D|$ bytes, is encrypted as follows:
 - First, a padding string PS , consisting of $k - 3 - |D|$ nonzero bytes, is generated pseudo-randomly (the byte length of PS is atleast 8).
 - Now, the encryption block $EB = 00||02||PS||00||D$ is formed, is converted into an integer x , and is encrypted with RSA, giving the ciphertext $c = x^e \pmod{n}$.

PKCS#1

- **DEC**(c, SK, PK) A Ciphertext c is decrypted as follows:
 - Compute $x' = c^d \pmod{n}$.
 - Converts x' into an encryption block EB' .
 - Check, if the encryption block is PKCS *conforming* (An encryption block EB consisting of k bytes, $EB = EB_1 || \dots || EB_k$, is called PKCS conforming, if it satisfies the following conditions: $EB_1 = 00$, $EB_2 = 02$, EB_3 through EB_{10} are nonzero and at least one of the bytes EB_{11} through EB_k is 00).
 - If the encryption block is PKCS conforming, then output the data block; otherwise an error sign.

Bleichenbacher's Attack on PKCS#1

Bleichenbacher's attack assumes that the adversary has access to an oracle that, for every ciphertext, returns whether the corresponding plaintext is PKCS conforming. If the plaintext is not PKCS conforming, the oracle outputs an error sign. Given just these error signs, because of specific properties of PKCS #1, Bleichenbacher showed how a very clever program can decrypt a target ciphertext (the oracle answer will reveal the first two bytes of the corresponding plaintext of the chosen ciphertext).

Bleichenbacher's Attack on PKCS#1

Bleichenbacher's attack assumes that the adversary has access to an oracle that, for every ciphertext, returns whether the corresponding plaintext is PKCS conforming. If the plaintext is not PKCS conforming, the oracle outputs an error sign. Given just these error signs, because of specific properties of PKCS #1, Bleichenbacher showed how a very clever program can decrypt a target ciphertext (the oracle answer will reveal the first two bytes of the corresponding plaintext of the chosen ciphertext).

D. Bleichenbacher. *Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1*. In Proc. Crypto'98, pages 1-12, 1998.

- CCVA makes sense.

- CCVA makes sense.

Questions

- 1 Does there exist any encryption scheme which is IND-CCVA secure but not IND-CCA secure?
- 2 Does there exist any encryption scheme which is IND-CPA secure but not IND-CCVA secure?

A glance over some existing schemes

ElGamal Encryption Scheme

- **KG(1^λ):** The key generation algorithm runs as follows.
 - Choose a group G of prime order p , where $2^{\lambda-1} < p < 2^\lambda$
 - Choose $g \xleftarrow{\mathcal{R}} G$ and $x \xleftarrow{\mathcal{R}} \mathbb{Z}_p$.
 - Compute $c = g^x$.
 - The public key, PK , for this scheme is tuple (G, g, c) , with corresponding secret key, SK , is x .
 - message space = G .
 - ciphertext space = $G \times G$
- **ENC(m, PK):** To encrypt a message $m \in G$, the encryption algorithm runs as follows.
 - Choose $r \xleftarrow{\mathcal{R}} \mathbb{Z}_p$.
 - Compute $u = g^r, e = mc^r$.
 - The ciphertext, \mathcal{C} , is (u, e) .
- **DEC(\mathcal{C}, SK, PK):** Decryption works in the following way: given the ciphertext (u, e) and secret key (x) ,
 - Compute $m = eu^{-x}$

Security of ElGamal Encryption Scheme

- ElGamal is IND-CPA secure if DDH assumption holds in G .

Security of ElGamal Encryption Scheme

- ElGamal is IND-CPA secure if DDH assumption holds in G .

Definition

Let \mathcal{D} be an algorithm that takes triples of group elements as input and outputs a bit. The DDH-advantage of \mathcal{D} is defined as

$$|\Pr[\mathcal{D}(g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{D}(g^x, g^y, g^z) = 1]|$$

Then DDH assumption for G assumes that for any efficient algorithm \mathcal{D} , it's DDH-advantage is negligible.

Security of ElGamal Encryption Scheme

- ElGamal is IND-CPA secure if DDH assumption holds in G .

Definition

Let \mathcal{D} be an algorithm that takes triples of group elements as input and outputs a bit. The DDH-advantage of \mathcal{D} is defined as

$$|\Pr[\mathcal{D}(g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{D}(g^x, g^y, g^z) = 1]|$$

Then DDH assumption for G assumes that for any efficient algorithm \mathcal{D} , it's DDH-advantage is negligible.

- ElGamal is not IND-CCA secure.

Security of ElGamal Encryption Scheme

- ElGamal is IND-CPA secure if DDH assumption holds in G .

Definition

Let \mathcal{D} be an algorithm that takes triples of group elements as input and outputs a bit. The DDH-advantage of \mathcal{D} is defined as

$$|\Pr[\mathcal{D}(g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{D}(g^x, g^y, g^z) = 1]|$$

Then DDH assumption for G assumes that for any efficient algorithm \mathcal{D} , it's DDH-advantage is negligible.

- ElGamal is not IND-CCA secure.
- ElGamal is IND-CCVA secure if DDH assumption holds in G .

Cramer-Shoup's Light Version

- **KG(1^λ)**: The key generation algorithm runs as follows.
 - Choose a group G of prime order p , where $2^{\lambda-1} < p < 2^\lambda$
 - Choose $g_1, g_2 \stackrel{\mathcal{R}}{\leftarrow} G$ and $x_1, x_2, z \in \mathbb{Z}_p$.
 - Compute $c = g_1^{x_1} g_2^{x_2}$ and $h = g_1^z$.
 - The public key, PK , for this scheme is tuple (g_1, g_2, c, h) , with corresponding secret key, SK , is (x_1, x_2, z) .
 - message space = G .
 - ciphertext space = $G \times G \times G \times G$.

Cramer-Shoup's Light Version

- **ENC**(m, PK): To encrypt a message $m \in G$, the encryption algorithm runs as follows.
 - Choose $r \xleftarrow{\mathcal{R}} \mathbb{Z}_p$.
 - Compute $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r m$, $v = c^r$.
 - The ciphertext, \mathcal{C} , is (u_1, u_2, e, v) .
- **DEC**(\mathcal{C}, SK, PK): Decryption works in the following way: given the ciphertext (u_1, u_2, e, v) and secret key (x_1, x_2, z) ,
 - it first tests if $u_1^{x_1} u_2^{x_2} \stackrel{?}{=} v$.
 - If this condition does not hold, the decryption algorithm outputs \perp ; otherwise, it outputs

$$m = \frac{e}{u_1^z}.$$

Security of Cramer-Shoup's Light Version

\mathcal{B} is given as input a 4-tuple (g, g^a, g^b, Z) . The task of \mathcal{B} is to determine whether Z is equal to g^{ab} or a random element of G . \mathcal{B} solves this problem by interacting with \mathcal{A} in the IND-CCVA game as follows.

- **Simulation of Key Generation (KG):** \mathcal{B} proceeds as follows:
 - Sets $g_1 = g$.
 - Chooses $s \xleftarrow{\mathcal{R}} \mathbb{Z}_p$ and sets $g_2 = g_1^s$.
 - Chooses $x_1, x_2 \xleftarrow{\mathcal{R}} \mathbb{Z}_p$ and sets $c = g_1^{x_1} g_2^{x_2}$.
 - Sets $h = g^b$.
 - Finally the 4-tuple (g_1, g_2, c, h) is made available as public key to \mathcal{A} by \mathcal{B} .

Security of Cramer-Shoup's Light Version

- **Simulation of Ciphertext Verification Oracle for Ciphertext Validity Check:**
 - Knowledge of (x_1, x_2) ensures that \mathcal{B} can perfectly answer the ciphertext verification queries asked by \mathcal{A} .
- **Simulation of Challenge Ciphertext:**
 - In Challenge Phase, \mathcal{A} chooses and outputs two messages m_0 and m_1 to \mathcal{B} .
 - \mathcal{B} then chooses a bit $\tau \xleftarrow{\mathcal{R}} \{0, 1\}$ and it proceeds to encrypt m_τ .
 - \mathcal{B} sets

$$u_1 = g^a, \quad u_2 = (g^a)^s, \quad e = Z \cdot m_\tau \quad \text{and} \quad v = (g^a)^{x_1} (g^a)^{s x_2}.$$

- The challenge ciphertext (u_1, u_2, e, v) is given to \mathcal{A} by \mathcal{B} .

Finally in the Guess Phase, \mathcal{A} answers a bit τ' . If $\tau = \tau'$ then \mathcal{B} announces the input instance to be a valid DDH tuple. This completes the description of \mathcal{B} .

- Cramer-Shoup's light version is IND-CCVA secure if DDH assumption holds in G .
- Cramer-Shoup's light version is not IND-CCA secure.

ElGamal-ElGamal Encryption Scheme

- **KG(1^λ):** The key generation algorithm runs as follows.
 - Choose a group G of prime order p , where $2^{\lambda-1} < p < 2^\lambda$
 - Choose $g_1 \xleftarrow{\mathcal{R}} G, g_2 \xleftarrow{\mathcal{R}} G$ and $x_1, x_2 \in \mathbb{Z}_p$.
 - Compute $c_1 = g_1^{x_1}$ and $c_2 = g_2^{x_2}$.
 - The public key, PK , for this scheme is tuple (G, g_1, g_2, c_1, c_2) , with corresponding secret key, SK , is (x_1, x_2) .
 - message space = G .
 - message space = $G \times G \times G \times G$.
- **ENC(m, PK):** To encrypt a message $m \in G$, the encryption algorithm runs as follows.
 - Choose $r_1, r_2 \xleftarrow{\mathcal{R}} \mathbb{Z}_p$.
 - Compute $u_1 = g_1^{r_1}$, $u_2 = g_2^{r_2}$, $e_1 = mc_1^{r_1}$, $e_2 = mc_2^{r_2}$.
 - The ciphertext, \mathcal{C} , is (u_1, e_1, u_2, e_2) .

ElGamal-ElGamal Encryption Scheme

- **DEC(\mathcal{C}, SK, PK):** Decryption works in the following way: given the ciphertext (u_1, e_1, u_2, e_2) and secret key (x_1, x_2) ,
 - Compute $m_1 = \frac{e_1}{u_1^{x_1}}$
 - Compute $m_2 = \frac{e_2}{u_2^{x_2}}$
 - If $m_1 \neq m_2$ the decryption algorithm outputs \perp ; otherwise, it outputs m_1

Security of ElGamal-ElGamal scheme

- IND-CPA secure if DDH assumption holds in G .

Security of ElGamal-ElGamal scheme

- IND-CPA secure if DDH assumption holds in G .
- Not IND-CCVA secure.

$$C_b = (u_1^b, e_1^b, u_2^b, e_2^b)$$

↓

$$C'_b = (u_1^b, e_1^b, u_2^{b'}, e_2^{b'})$$

↓

if chosen ciphertext verification oracle returns 1, $b = b'$, else $b \neq b'$

Generic Construction

Let Π be a public key encryption scheme with \mathcal{K} as key space, \mathcal{M} as message space, and \mathcal{C} as ciphertext space. In general, we have

$$\cup_{k \in \mathcal{K}} \text{Enc}(\mathcal{M}) \subseteq \mathcal{C}.$$

Generic Construction

Let Π be a public key encryption scheme with \mathcal{K} as key space, \mathcal{M} as message space, and \mathcal{C} as ciphertext space. In general, we have

$$\cup_{k \in \mathcal{K}} \text{Enc}(\mathcal{M}) \subseteq \mathcal{C}.$$

If

- Π is IND-CPA secure but not IND-CCA secure, and
- $\cup_{k \in \mathcal{K}} \text{Enc}(\mathcal{M}) = \mathcal{C}$

Generic Construction

Let Π be a public key encryption scheme with \mathcal{K} as key space, \mathcal{M} as message space, and \mathcal{C} as ciphertext space. In general, we have

$$\cup_{k \in \mathcal{K}} \text{Enc}(\mathcal{M}) \subseteq \mathcal{C}.$$

If

- Π is IND-CPA secure but not IND-CCA secure, and
- $\cup_{k \in \mathcal{K}} \text{Enc}(\mathcal{M}) = \mathcal{C}$

then,

- There exists an IND-CPA secure encryption scheme which is not IND-CCVA secure, and
- There exists an IND-CCVA secure encryption scheme which is not IND-CCA secure

Thank You

Identity-Based Encryption: A 30-Minute Tour

Palash Sarkar

Applied Statistics Unit
Indian Statistical Institute, Kolkata
India
palash@isical.ac.in

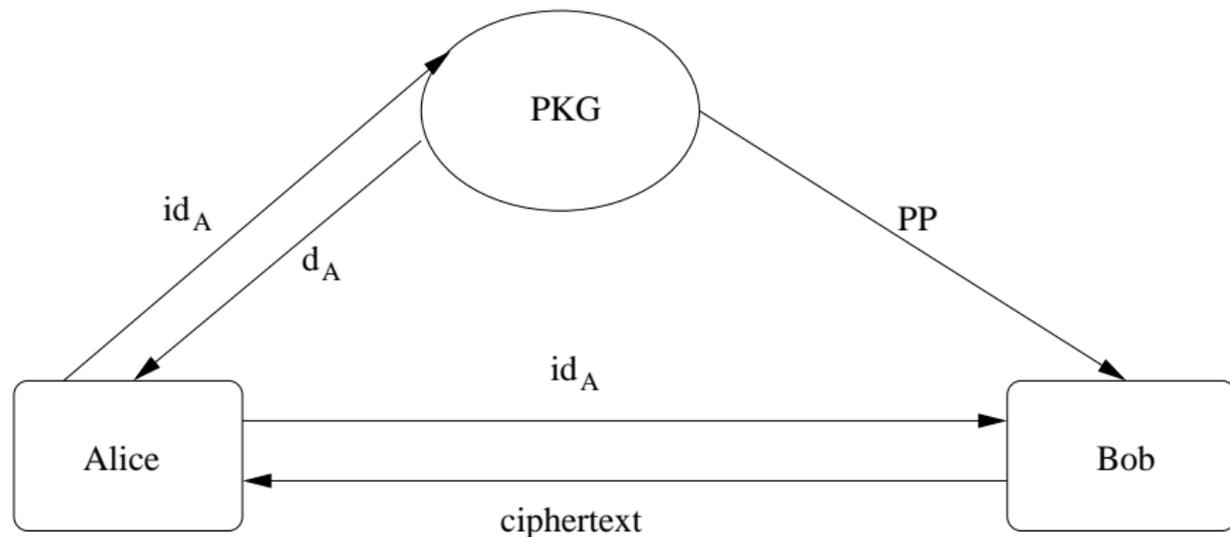


Structure of the Presentation

- A brief overview of IBE.
- Some constructions.
- Some issues.



Identity-Based Encryption



Bob sends a message to Alice.



Identity-Based Encryption

Proposed by Shamir in 1984.



Identity-Based Encryption

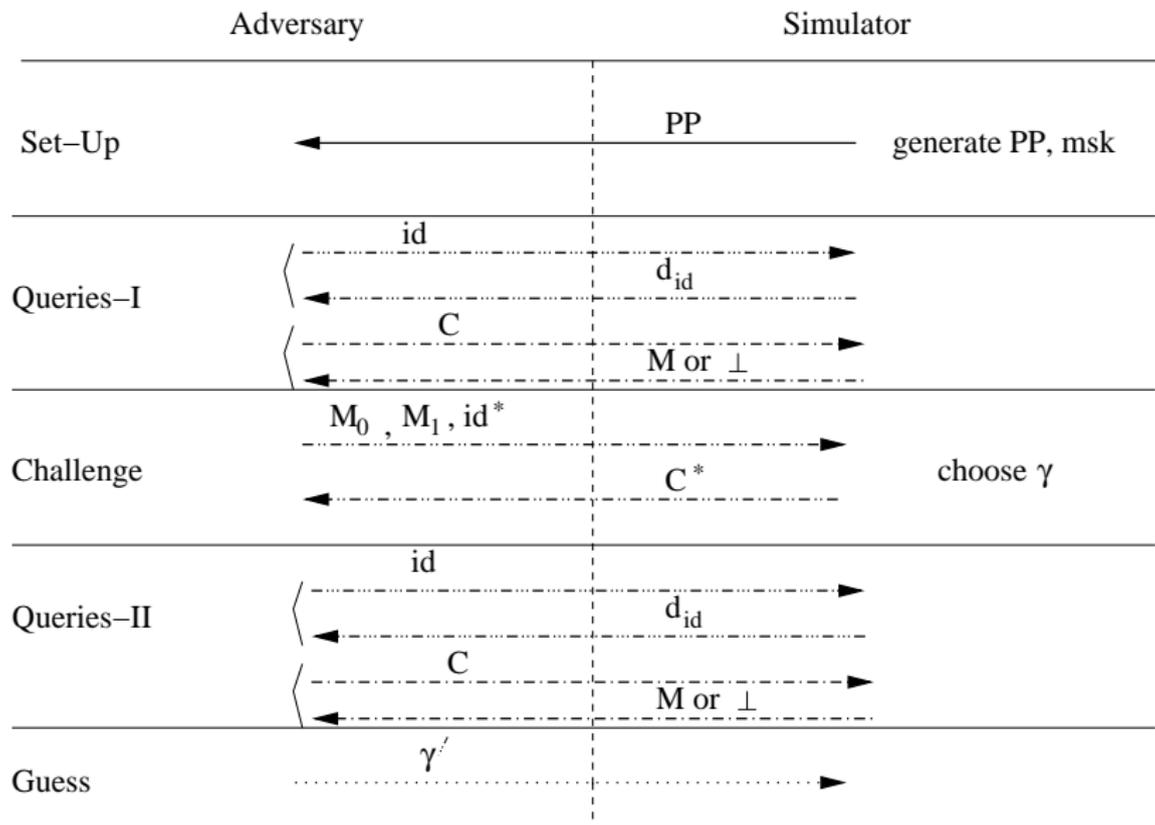
Proposed by Shamir in 1984.

Solutions:

- Cocks: 2001.
- Sakai, Ohgishi and Kasahara: 2000.
 - Described an identity-based key agreement scheme.
- Boneh and Franklin: 2001.
- Cocks' solution was based on quadratic residues.
- SOK and BF were based on bilinear maps.
- BF provided an appropriate security model.
- The BF work spurred a great deal of later research.



Identity-Based Encryption: Security Model



Identity-Based Encryption: Security Model

“**Full**” **model**: supports adaptive-identity and adaptive-ciphertext queries in an interleaved fashion.



“**Full**” **model**: supports adaptive-identity and adaptive-ciphertext queries in an interleaved fashion.

Restricted Models:

- CPA-secure: Ciphertext queries not allowed.
- Selective-identity: The challenge identity id^* is to be provided by the adversary even before receiving the PP.



Construction Approaches

- Based on quadratic residues.
- Based on lattices.
- Based on bilinear pairings of elliptic curve groups.



Setting: $N = pq$;

$J(N)$: set of elements with Jacobi symbol 1 modulo N ;

$QR(N)$: set of quadratic residues modulo N .



Setting: $N = pq$;

$J(N)$: set of elements with Jacobi symbol 1 modulo N ;

$QR(N)$: set of quadratic residues modulo N .

Public Parameters.

N ; $u \xleftarrow{\$} J(N) \setminus QR(N)$;

(u is a random pseudo-square;)

hash function $H()$ which maps identities into $J(N)$.

Master Secret Key: p and q .



Setting: $N = pq$;

$J(N)$: set of elements with Jacobi symbol 1 modulo N ;

$QR(N)$: set of quadratic residues modulo N .

Public Parameters.

N ; $u \xleftarrow{\$} J(N) \setminus QR(N)$;

(u is a random pseudo-square;)

hash function $H()$ which maps identities into $J(N)$.

Master Secret Key: p and q .

Key Gen: identity id .

$R = H(id)$; $r = \sqrt{R}$ or \sqrt{uR} according as R is square or not;

$d_{id} = r$.



Encryption: bit m , identity id .

- $R = H(\text{id}); t_0, t_1 \xleftarrow{\$} \mathbb{Z}_N$;
- compute $d_a = (t_a^2 + u^a R)/t_a$ and $c_a = (-1)^m \cdot (\frac{t_a}{N})$;
- ciphertext: $((d_0, c_0), (d_1, c_1))$.

Decryption: ciphertext $((d_0, c_0), (d_1, c_1))$, identity id ; $d_{\text{id}} = r$:

- $R = H(\text{id})$; set $a \in \{0, 1\}$ such that $r^2 = u^a R$;
- set $g = d_a + 2r$; (note $g = \left(\frac{(t_a+r)^2}{t_a}\right)$ and so, $(\frac{g}{N}) = (\frac{t_a}{N})$);
- compute $(-1)^m$ to be $c_a \cdot (\frac{g}{N})$.



- Ciphertext expansion is large; efficiency not good.
- Boneh, Gentry and Hamburg (2007) obtained improved space efficiency by reusing randomness; but, encryption and decryption efficiencies are worse.
- Jhanwar and Barua (2008) consider the problem of improving efficiency.



- Ciphertext expansion is large; efficiency not good.
- Boneh, Gentry and Hamburg (2007) obtained improved space efficiency by reusing randomness; but, encryption and decryption efficiencies are worse.
- Jhanwar and Barua (2008) consider the problem of improving efficiency.

This approach currently does not lead to practical schemes.



Gentry, Peikert and Vaikuntanathan (2008).

- Based on a technique called efficient Pre-Image Sampling.
 - This technique naturally leads to a signature scheme.
 - By considering the decryption key corresponding to an identity to be the PKG's signature on the identity (cf. Naor) suggests an IBE scheme.
- Security is based on the hardness of the Learning With Errors (LWE) problem.
- Later work have improved efficiency and provided constructions of hierarchical IBE (HIBE) schemes.



Motivation:

- Multi-precision arithmetic not required;
- Security based on the hardness of worst-case instance;
- No known quantum algorithm for solving lattice problems.



Motivation:

- Multi-precision arithmetic not required;
- Security based on the hardness of worst-case instance;
- No known quantum algorithm for solving lattice problems.
- These apply to all lattice problems and are not specific to lattice-based IBE.



Lattice-Based Approach: Pros and Cons

Motivation:

- Multi-precision arithmetic not required;
- Security based on the hardness of worst-case instance;
- No known quantum algorithm for solving lattice problems.
- These apply to all lattice problems and are not specific to lattice-based IBE.

Cons:

- The sizes of keys and ciphertexts are far too large compared to pairing-based schemes.



$$e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T.$$

- \mathbb{G}_1 and \mathbb{G}_2 are sub-groups of points on an elliptic curve; \mathbb{G}_T is a sub-group of the multiplicative group of a finite field.
- Types of pairings:
 - **Type-1:** $\mathbb{G}_1 = \mathbb{G}_2$ (symmetric pairing).
 - **Type-2:** An efficiently computable isomorphism from \mathbb{G}_2 to \mathbb{G}_1 is known.
 - **Type-3:** There is no known efficiently computable isomorphism from \mathbb{G}_2 to \mathbb{G}_1 (or vice versa).
- **Type-3** pairings are the fastest to compute and provide the most compact parameter sizes.



Boneh-Franklin IBE

- **Setup:** $\mathbb{G}_1 = \langle P \rangle$, $s \xleftarrow{\$} \mathbb{Z}_p$, $Q = sP$;
 $PP = (P, Q, H_1(), H_2()), \text{msk} = s.$
- **Key-Gen:** Given id , compute $Q_{\text{id}} = H_1(\text{id})$; $d_{\text{id}} = sQ_{\text{id}}$.
- **Encrypt:** Choose $r \xleftarrow{\$} \mathbb{Z}_p$; $C = (rP, M \oplus H_2(\underbrace{e(Q, Q_{\text{id}})^r}))$
- **Decrypt:** Given $C = (U, V)$ and d_{id} compute
 $V \oplus H_2(\underbrace{e(U, d_{\text{id}})}) = M.$



Boneh-Franklin IBE

- **Setup:** $\mathbb{G}_1 = \langle P \rangle$, $s \xleftarrow{\$} \mathbb{Z}_p$, $Q = sP$;
 $PP = (P, Q, H_1(), H_2())$, $msk = s$.
- **Key-Gen:** Given id , compute $Q_{id} = H_1(id)$; $d_{id} = sQ_{id}$.
- **Encrypt:** Choose $r \xleftarrow{\$} \mathbb{Z}_p$; $C = (rP, M \oplus H_2(\underbrace{e(Q, Q_{id})^r}))$
- **Decrypt:** Given $C = (U, V)$ and d_{id} compute
 $V \oplus H_2(\underbrace{e(U, d_{id})}) = M$.

Correctness:

$$e(U, d_{ID}) = e(rP, sQ_{ID}) = e(sP, Q_{ID})^r = e(Q, Q_{ID})^r.$$



Boneh-Franklin IBE

- **Setup:** $\mathbb{G}_1 = \langle P \rangle$, $s \xleftarrow{\$} \mathbb{Z}_p$, $Q = sP$;
 $PP = (P, Q, H_1(), H_2())$, $msk = s$.
- **Key-Gen:** Given id , compute $Q_{id} = H_1(id)$; $d_{id} = sQ_{id}$.
- **Encrypt:** Choose $r \xleftarrow{\$} \mathbb{Z}_p$; $C = (rP, M \oplus H_2(\underbrace{e(Q, Q_{id})^r}))$
- **Decrypt:** Given $C = (U, V)$ and d_{id} compute
 $V \oplus H_2(\underbrace{e(U, d_{id})}) = M$.

Correctness:

$$e(U, d_{ID}) = e(rP, sQ_{ID}) = e(sP, Q_{ID})^r = e(Q, Q_{ID})^r.$$

The scheme is CPA-secure; can be converted to CCA-secure using standard techniques such as the Fujisaki-Okamoto conversion.



Pros:

- Simple, elegant, efficient, compact, ...
- Leads naturally to signature scheme, HIBE and other primitives.
- Best known practical attack: Solve DL in \mathbb{G}_1 or \mathbb{G}_2 .



Pros:

- Simple, elegant, efficient, compact, ...
- Leads naturally to signature scheme, HIBE and other primitives.
- Best known practical attack: Solve DL in \mathbb{G}_1 or \mathbb{G}_2 .

Cons:

- Security argument is based on random oracles.
- Security reduction to the Decisional Bilinear Diffie-Hellman (DBDH) problem is not tight.



Boneh-Boyen (2004): BB-IBE1 (also BB-IBE2).

- Selective-id secure.
- Introduced the so-called “commutative blinding” framework and algebraic techniques to handle key-extraction queries.
- Described using Type-1 pairings; can be easily modified to Type-3 pairings.
- Extends easily to HIBE.
- Later used by Boyen-Mei-Waters to convert CPA-secure pairing-based schemes to CCA-secure schemes.



Some Important Pairing-Based IBE Schemes

Waters (2005):

- Adaptive-id secure without random oracles.
- Builds on BB-IBE1 and another work by Boneh and Boyen.
- Public parameter size rather large (≈ 160 EC points for 80-bit security).
 - Independent follow up work by Naccache (2005) and Chatterjee-Sarkar (2005) showed how to reduce the PP size; trade-off is a looser security reduction.
- Original description in the Type-1 setting.
 - Converted to Type-2 setting by Bellare and Ristenpart (2009).
 - Converted to Type-3 setting by Chatterjee and Sarkar (2010).
- Security analysis introduced a technique called artificial abort.
 - Later analysis by Bellare-Ristenpart showed how to avoid artificial abort, but, at the cost of losing tightness.



Gentry (2006):

- Adaptive-id secure, no random oracles, tight reduction, efficient.
- But, based on the hardness of a non-static assumption, i.e., the number of elements in the instance depends on the number of queries made by the adversary.



Some Important Pairing-Based IBE Schemes

Waters (2009):

- Introduces a new technique called dual-system encryption.
- Adaptive-id secure, no random oracles, standard (static) assumption.
- Constant size public parameters.
 - For Waters (2005) and its variants the size of the PP asymptotically grows with the security parameter.
- Extends to HIBE and BE schemes.
- Uses the Type-1 setting.
 - Simplification and conversion to Type-3 setting by Ramanna-Chatterjee-Sarkar (2011).



Some Important Pairing-Based IBE Schemes

Waters (2009):

- Introduces a new technique called dual-system encryption.
- Adaptive-id secure, no random oracles, standard (static) assumption.
- Constant size public parameters.
 - For Waters (2005) and its variants the size of the PP asymptotically grows with the security parameter.
- Extends to HIBE and BE schemes.
- Uses the Type-1 setting.
 - Simplification and conversion to Type-3 setting by Ramanna-Chatterjee-Sarkar (2011).

Lewko-Waters (2010):

- Dual-system based IBE; extends to constant-size ciphertext HIBE.
- Using pairing over composite order groups and also Type-3 setting.
 - An improved variant in the Type-3 setting (coming).



An Open Problem

Obtain an IBE scheme with the following properties.

- Adaptive-id secure.
- No random oracles.
- Standard hardness assumptions.
- (Efficient – constant size parameters; constant number of scalar multiplications, pairings; ...)
- *Tight security reduction.*



An Open Problem

Obtain an IBE scheme with the following properties.

- Adaptive-id secure.
- No random oracles.
- Standard hardness assumptions.
- (Efficient – constant size parameters; constant number of scalar multiplications, pairings; ...)
- *Tight security reduction.*

Or show that this cannot be done.



Which IBE scheme should I use?



Which IBE scheme should I use?

- QR, lattice-based or pairing-based?



Which IBE scheme should I use?

- QR, lattice-based or pairing-based?
- For pairing-based schemes, the best known attack on all proposed schemes is to solve DL. So, do I use BF?



Which IBE scheme should I use?

- QR, lattice-based or pairing-based?
- For pairing-based schemes, the best known attack on all proposed schemes is to solve DL. So, do I use BF?
- For pairing-based schemes, should I care about using Type-1 versus Type-3 pairings.
 - From a security point of view, is the use of Type-3 pairing weaker because of the *assumption* that isomorphisms between \mathbb{G}_1 and \mathbb{G}_2 cannot be computed?



Which IBE scheme should I use?

- QR, lattice-based or pairing-based?
- For pairing-based schemes, the best known attack on all proposed schemes is to solve DL. So, do I use BF?
- For pairing-based schemes, should I care about using Type-1 versus Type-3 pairings.
 - From a security point of view, is the use of Type-3 pairing weaker because of the *assumption* that isomorphisms between \mathbb{G}_1 and \mathbb{G}_2 cannot be computed?
- Should I care about security reductions? If so, then
 - Should I care about selective-id versus adaptive-id models?
 - Should I care about the underlying assumptions? Should I care about static versus non-static assumptions? Among static assumptions, should I care about standard versus non-standard assumptions?
 - Should I care about the tightness of reduction?



Thank you for your attention!



A new perturbation strategy for NTRUSign

John M. Schanck

Security Innovation

January 12, 2012

Outline

- Brief description of NTRUSign
- How it has been broken
- Recent advances in similar (but provably secure!) schemes
- Barriers to adapting these techniques to NTRUSign

Background

- Introduced in 2001
- Efficient instantiation of Goldreich-Goldwasser-Halevi (GGH) signatures
- Private key (**B**): Good basis.
 - Short, nearly orthogonal, vectors
- Public key (**H**): Bad basis.
 - Hermite Normal Form of **B**

- Signing
 - Hash document to a point $m \in \mathbb{Z}_q^{2N}$
 - Use private basis to find nearby lattice point
 - Babai's round-off CVP approximation algorithm
 - $s = \lfloor mB^{-1} \rfloor B$
 - $\lfloor \cdot \rfloor$ rounds each component to the nearest integer.

Background

- The NTRU ring
 - $R = \mathbb{Z}_q[X]/(X^N - 1)$
 - Polynomials of degree $\leq N - 1$; coefficients in \mathbb{Z}_q
 - Ring operations: component-wise addition and cyclic convolution
 - N prime; q power of two
 - Isomorphism between R and circulant matrices
 - For $a \in R$:

$$C(a) = \begin{pmatrix} a_0 & a_1 & a_2 & \dots & a_{N-1} \\ a_{N-1} & a_0 & a_1 & \dots & a_{N-2} \\ a_{N-2} & a_{N-1} & a_0 & \dots & a_{N-3} \\ & & & \ddots & \\ a_1 & a_2 & a_3 & \dots & a_0 \end{pmatrix}$$

- NTRU Lattice

- $B = \begin{pmatrix} f & F \\ g & G \end{pmatrix} \cong \begin{pmatrix} C(f) & C(F) \\ C(g) & C(G) \end{pmatrix}$

- f and g are chosen randomly

- F and G are found which satisfy $f * G - F * g = q$

- NTRU Lattice

- $B = \begin{pmatrix} f & F \\ g & G \end{pmatrix} \cong \begin{pmatrix} C(f) & C(F) \\ C(g) & C(G) \end{pmatrix}$

- f and g are chosen randomly

- F and G are found which satisfy $f * G - F * g = q$

- $B^{-1} = \frac{1}{q} \begin{pmatrix} G & -F \\ -g & f \end{pmatrix}$

- NTRU Lattice

- $B = \begin{pmatrix} f & F \\ g & G \end{pmatrix} \cong \begin{pmatrix} C(f) & C(F) \\ C(g) & C(G) \end{pmatrix}$

- f and g are chosen randomly

- F and G are found which satisfy $f * G - F * g = q$

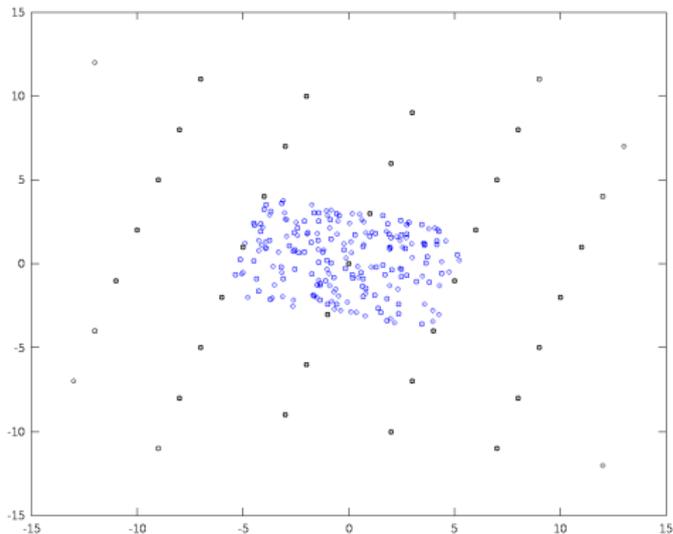
- $B^{-1} = \frac{1}{q} \begin{pmatrix} G & -F \\ -g & f \end{pmatrix}$

- $h = f * g^{-1}$

- $H = \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}$

- Very effective attack on vanilla NTRUSign

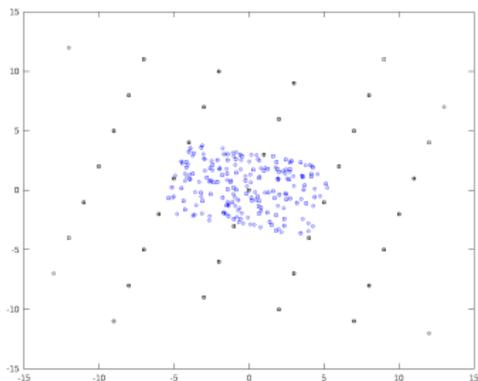
- Very effective attack on vanilla NTRUSign
- Transcript of (*signature* – *message*) yields points in $U(\mathcal{P}(B))$



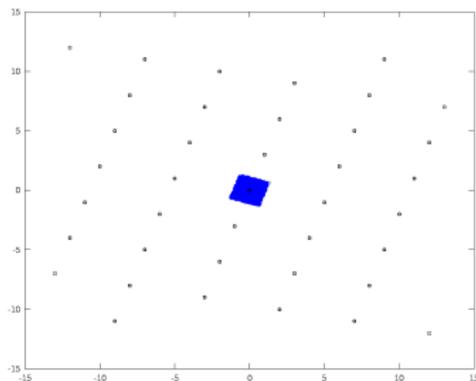
- $\text{Cov}[\text{transcript}] \approx \text{Cov}[U(\mathcal{P}(B))] = \frac{1}{3}G$
- Gram matrix: $G = B^t B = \begin{pmatrix} \bar{f}f + \bar{g}g & \bar{f}F + \bar{g}G \\ \bar{F}f + \bar{G}g & \bar{F}F + \bar{G}G \end{pmatrix}$

- $\text{Cov}[\text{transcript}] \approx \text{Cov}[U(\mathcal{P}(B))] = \frac{1}{3}G$
- Gram matrix: $G = B^t B = \begin{pmatrix} \bar{f}f + \bar{g}g & \bar{f}F + \bar{g}G \\ \bar{F}f + \bar{G}g & \bar{F}F + \bar{G}G \end{pmatrix}$
- $G^{-1} = L^t L$

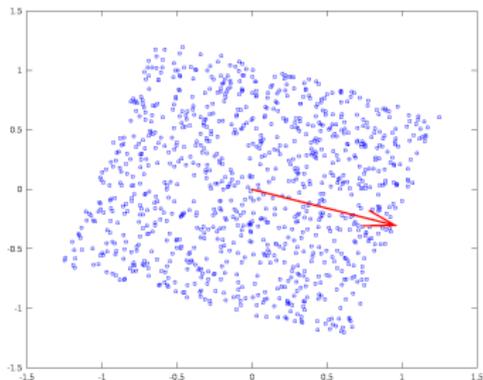
- $\text{Cov}[\text{transcript}] \approx \text{Cov}[U(\mathcal{P}(B))] = \frac{1}{3}G$
- Gram matrix: $G = B^t B = \begin{pmatrix} \bar{f}f + \bar{g}g & \bar{f}F + \bar{g}G \\ \bar{F}f + \bar{G}g & \bar{F}F + \bar{G}G \end{pmatrix}$
- $G^{-1} = L^t L$



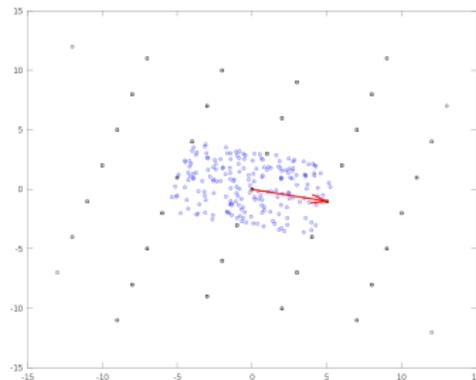
$L \rightarrow$



- Independent Component Analysis (ICA) on orthogonalized transcript can reveal the private basis
 - Gradient descent on kurtosis



L^{-1} →



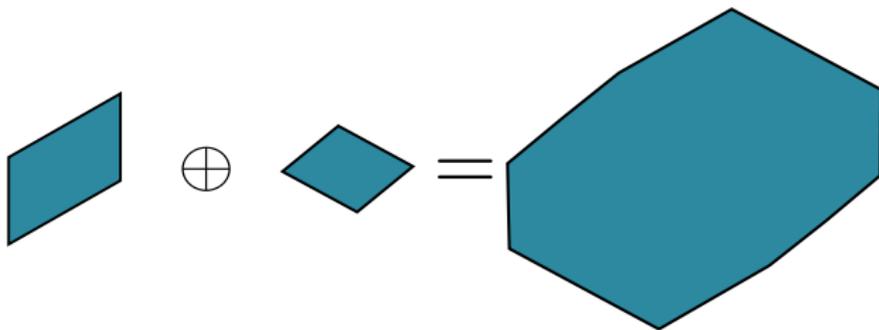
Countermeasures: Perturbations

Signing with perturbations

- Additional perturbation keys are created during keygen and kept private
- The message, m_0 , is signed with the first perturbation key, yielding m_1 , which is signed with the second perturbation key yielding $m_2...$
- m_k is signed with the private key, yielding s .

Countermeasures: Perturbations

Transcript points lie in Minkowski sum of parallelepipeds



Discrete Gaussian Sampling

- Gentry, Peikert and Vaikuntanathan (STOC 2008)
 - Samples directly from a discrete Gaussian distribution
 - Randomized variant of Babai's nearest plane algorithm
 - Rejection sampling
 - Inherently sequential and cubic in n

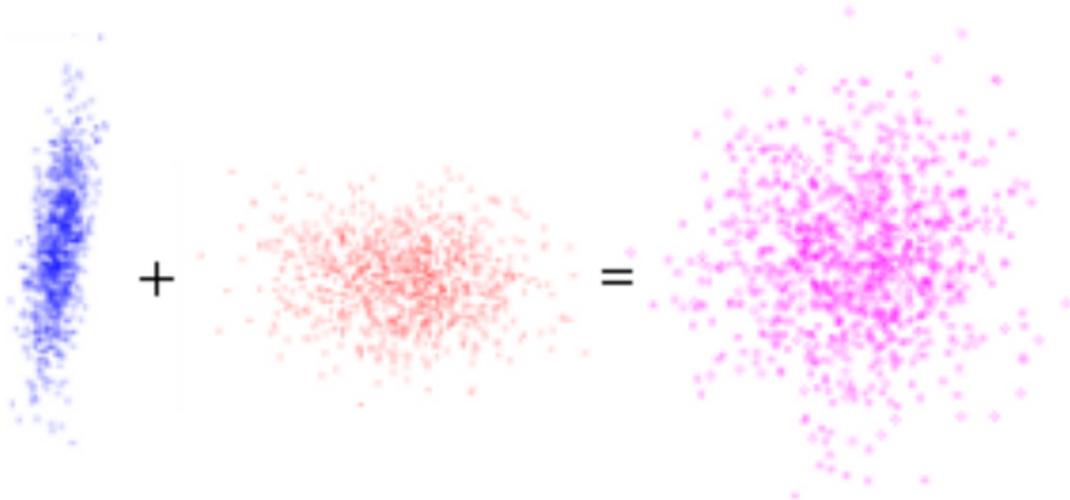
Discrete Gaussian Sampling

- Gentry, Peikert and Vaikuntanathan (STOC 2008)
 - Samples directly from a discrete Gaussian distribution
 - Randomized variant of Babai's nearest plane algorithm
 - Rejection sampling
 - Inherently sequential and cubic in n
 - Ultimately impractical

Discrete Gaussian Sampling

- Gentry, Peikert and Vaikuntanathan (STOC 2008)
 - Samples directly from a discrete Gaussian distribution
 - Randomized variant of Babai's nearest plane algorithm
 - Rejection sampling
 - Inherently sequential and cubic in n
 - Ultimately impractical
- Peikert (CRYPTO 2010)
 - Removes dependence on nearest plane algorithm
 - Still requires $O(n^2 \log q)$ bits of storage as well as operations on $n \times n$ matrices.

Peikert (CRYPTO 2010)



Offline phase:

- Input:
 - Basis B
 - Rounding parameter $r = \omega(\sqrt{\log n})$
 - Target distribution $\Sigma_{target} > B^t B$

Offline phase:

- Input:
 - Basis B
 - Rounding parameter $r = \omega(\sqrt{\log n})$
 - Target distribution $\Sigma_{target} \geq r^2(4B^t B + \mathbb{I}) > B^t B$

Offline phase:

- Input:
 - Basis B
 - Rounding parameter $r = \omega(\sqrt{\log n})$
 - Target distribution $\Sigma_{target} \geq r^2(4B^t B + \mathbb{I}) > B^t B$
- Compute:
 - $\Sigma_{basis} = B^t B$

Offline phase:

- Input:
 - Basis B
 - Rounding parameter $r = \omega(\sqrt{\log n})$
 - Target distribution $\Sigma_{target} \geq r^2(4B^t B + \mathbb{I}) > B^t B$
- Compute:
 - $\Sigma_{basis} = 2r^2 B^t B$

Offline phase:

- Input:

- Basis B
- Rounding parameter $r = \omega(\sqrt{\log n})$
- Target distribution $\Sigma_{target} \geq r^2(4B^t B + \mathbb{I}) > B^t B$

- Compute:

- $\Sigma_{basis} = 2r^2 B^t B$
- $\Sigma_{noise} = \Sigma_{target} - \Sigma_{basis}$
- $B_{noise} = \sqrt{\Sigma_{noise} - r^2 \mathbb{I}}$

Offline phase:

- Input:

- Basis B
- Rounding parameter $r = \omega(\sqrt{\log n})$
- Target distribution $\Sigma_{target} \geq r^2(4B^t B + \mathbb{I}) > B^t B$

- Compute:

- $\Sigma_{basis} = 2r^2 B^t B$
- $\Sigma_{noise} = \Sigma_{target} - \Sigma_{basis}$
- $B_{noise} = \sqrt{\Sigma_{noise} - r^2 \mathbb{I}}$

- Output:

- B_{noise}

Online phase:

- Input:
 - A vector $c \in \mathbb{Z}^n$
- Draw noise vector, x , from discrete Gaussian over \mathbb{Z}^n with covariance Σ_{noise} . $x \leftarrow \lfloor B_{noise} \cdot D_1 \rfloor_r$
- Add noise to c and sign. $s = \lfloor (c - x)B^{-1} \rfloor_r B$

Adapting Peikert '10 to NTRUSign

- Problems:
 - Operations on $2N \times 2N$ matrices, and storage of a $2N \times 2N$ matrix
 - Probably forces parameters too large
 - Too many standard Gaussian samples

A tiny step forward

We can find sets of vectors in the NTRU ring that take the place of B_{noise}

Circulant Eigendecomposition

N -Point DFT Matrix

$$W = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{(N-1)} & \omega^{2(N-1)} & \omega^{3(N-1)} & \dots & \omega^{(N-1)^2} \end{pmatrix}$$

All circulant matrices $C(p) = W^* \Lambda W$

Just a restatement of convolution theorem:

$$f * g = \mathcal{F}^{-1} \{ \mathcal{F}\{f\} \cdot \mathcal{F}\{g\} \}$$

Block-wise Diagonalization of an NTRU Lattice

$$(\mathbb{I}_2 \otimes W) \begin{pmatrix} C(f) & C(F) \\ C(g) & C(G) \end{pmatrix} (\mathbb{I}_2 \otimes W)^* = \begin{pmatrix} \Lambda_f & \Lambda_F \\ \Lambda_g & \Lambda_G \end{pmatrix}$$

Block-wise Diagonalization of an NTRU Lattice

Applications:

- Fast eigenvalue computation; Solve N 2×2 eigensystems instead of one $2N \times 2N$ eigensystem.

Block-wise Diagonalization of an NTRU Lattice

Applications:

- Fast eigenvalue computation; Solve N 2×2 eigensystems instead of one $2N \times 2N$ eigensystem.
- Generate noise distributions for use with Peikert's sampler.

Finding B_{noise} in NTRU module

- Given B , construct C such that (ignoring scaling factors)

$$B^t B + C^t C = s * \mathbb{I}_{2N}$$

Finding B_{noise} in NTRU module

- Block-wise diagonalize Gram matrix:

$$B^t B = \begin{pmatrix} \bar{f}f + \bar{g}g & \bar{f}F + \bar{g}G \\ \bar{F}f + \bar{G}g & \bar{F}F + \bar{G}G \end{pmatrix} \xrightarrow{(\mathbb{I}_2 \otimes W)} \begin{pmatrix} \Lambda_a & \Lambda_b \\ \Lambda_b^* & \Lambda_c \end{pmatrix}$$

- Permute to block diagonal form
- Blocks are Hermitian; positive semi-definite

$$\begin{pmatrix} \lambda_a & \lambda_b \\ \lambda_b^* & \lambda_c \end{pmatrix}$$

- So is

$$\begin{pmatrix} s - \lambda_a & \lambda_b \\ \lambda_b^* & s - \lambda_c \end{pmatrix}$$

Finding B_{noise} in NTRU module

- Invert permutation step
- Invert Fourier transform
- Result is a set of 4 polynomials in $\mathbb{Q}_q[X]/(X^N - 1)$
- Computation doesn't actually involve any $2N \times 2N$ matrices!
All operations can be performed on polynomials and 2×2 matrices.

Once more with polynomials

$$\begin{array}{cccc} f_1 & f_2 & f_3 & f_4 \\ g_1 & g_2 & g_3 & g_4 \end{array} \quad \begin{array}{cccc} F_1 & F_2 & F_3 & F_4 \\ G_1 & G_2 & G_3 & G_4 \end{array}$$

Once more with polynomials

$$\begin{array}{cccccccc} f_1 & f_2 & f_3 & f_4 & F_1 & F_2 & F_3 & F_4 \\ g_1 & g_2 & g_3 & g_4 & G_1 & G_2 & G_3 & G_4 \end{array}$$

DFT

$$\begin{array}{cccccccc} \hat{f}_1 & \hat{f}_2 & \hat{f}_3 & \hat{f}_4 & \hat{F}_1 & \hat{F}_2 & \hat{F}_3 & \hat{F}_4 \\ \hat{g}_1 & \hat{g}_2 & \hat{g}_3 & \hat{g}_4 & \hat{G}_1 & \hat{G}_2 & \hat{G}_3 & \hat{G}_4 \end{array}$$

Once more with polynomials

$$\begin{array}{cccccccc} f_1 & f_2 & f_3 & f_4 & F_1 & F_2 & F_3 & F_4 \\ g_1 & g_2 & g_3 & g_4 & G_1 & G_2 & G_3 & G_4 \end{array}$$

DFT

$$\begin{array}{cccccccc} \hat{f}_1 & \hat{f}_2 & \hat{f}_3 & \hat{f}_4 & \hat{F}_1 & \hat{F}_2 & \hat{F}_3 & \hat{F}_4 \\ \hat{g}_1 & \hat{g}_2 & \hat{g}_3 & \hat{g}_4 & \hat{G}_1 & \hat{G}_2 & \hat{G}_3 & \hat{G}_4 \end{array}$$

Shuffle

$$\begin{array}{cccccccc} \hat{f}_1 & \hat{F}_1 & \hat{f}_2 & \hat{F}_2 & \hat{f}_3 & \hat{F}_3 & \hat{f}_4 & \hat{F}_4 \\ \hat{g}_1 & \hat{G}_1 & \hat{g}_2 & \hat{G}_2 & \hat{g}_3 & \hat{G}_3 & \hat{g}_4 & \hat{G}_4 \end{array}$$

Once more with polynomials

$$\begin{array}{cccccccc} f_1 & f_2 & f_3 & f_4 & F_1 & F_2 & F_3 & F_4 \\ g_1 & g_2 & g_3 & g_4 & G_1 & G_2 & G_3 & G_4 \end{array}$$

DFT

$$\begin{array}{cccccccc} \hat{f}_1 & \hat{f}_2 & \hat{f}_3 & \hat{f}_4 & \hat{F}_1 & \hat{F}_2 & \hat{F}_3 & \hat{F}_4 \\ \hat{g}_1 & \hat{g}_2 & \hat{g}_3 & \hat{g}_4 & \hat{G}_1 & \hat{G}_2 & \hat{G}_3 & \hat{G}_4 \end{array}$$

Shuffle

$$\begin{pmatrix} \hat{f}_1 & \hat{F}_1 \\ \hat{g}_1 & \hat{G}_1 \end{pmatrix} \quad \begin{pmatrix} \hat{f}_2 & \hat{F}_2 \\ \hat{g}_2 & \hat{G}_2 \end{pmatrix} \quad \begin{pmatrix} \hat{f}_3 & \hat{F}_3 \\ \hat{g}_3 & \hat{G}_3 \end{pmatrix} \quad \begin{pmatrix} \hat{f}_4 & \hat{F}_4 \\ \hat{g}_4 & \hat{G}_4 \end{pmatrix}$$

Once more with polynomials

Cholesky factorize each 2x2 matrix

$$\begin{pmatrix} \hat{a}_1 & 0 \\ \hat{b}_1 & \hat{c}_1 \end{pmatrix} \quad \begin{pmatrix} \hat{a}_2 & 0 \\ \hat{b}_2 & \hat{c}_2 \end{pmatrix} \quad \begin{pmatrix} \hat{a}_3 & 0 \\ \hat{b}_3 & \hat{c}_3 \end{pmatrix} \quad \begin{pmatrix} \hat{a}_4 & 0 \\ \hat{b}_4 & \hat{c}_4 \end{pmatrix}$$

Invert permutation

$$\begin{array}{cccccccc} \hat{a}_1 & \hat{a}_2 & \hat{a}_3 & \hat{a}_4 & 0 & 0 & 0 & 0 \\ \hat{b}_1 & \hat{b}_2 & \hat{b}_3 & \hat{b}_4 & \hat{c}_1 & \hat{c}_2 & \hat{c}_3 & \hat{c}_4 \end{array}$$

Apply inverse DFT

$$\sqrt{B^t B} = \begin{pmatrix} a & 0 \\ b & c \end{pmatrix}$$

Going forward

- Parameters
- Analyze options w.r.t sampling from other distributions

Signature Schemes from Lattices

Ananth Raghunathan

Stanford University

ISI Kolkata, Jan 2012

Short Integer Solutions (SIS)

$$A: \mathbb{Z}^m \rightarrow (\mathbb{Z}_q)^n$$

$$\begin{matrix} m \\ \boxed{A} \\ n \end{matrix} \cdot \begin{matrix} \boxed{v} \end{matrix} = \begin{matrix} \boxed{u} \end{matrix} \pmod{q}$$

The SIS problem (hard):

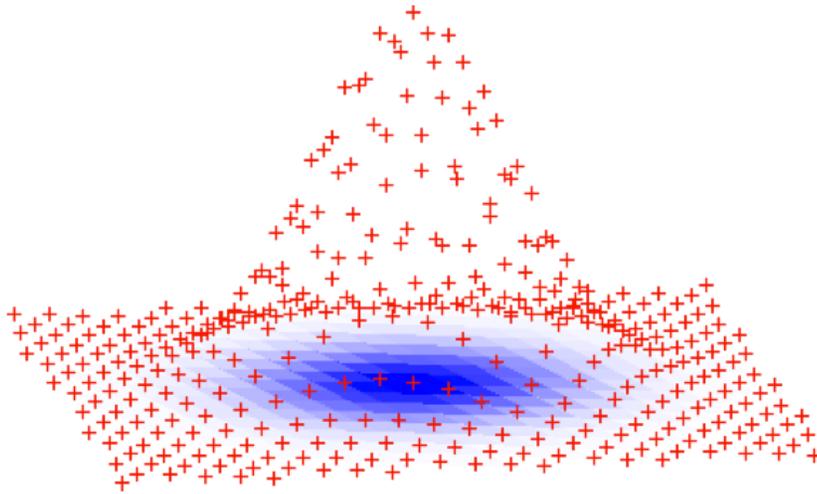
Given \mathbf{A} , \mathbf{u} find \mathbf{v} such that $\mathbf{A} \cdot \mathbf{v} = \mathbf{u}$ and $|\mathbf{v}|$ is “small”

The SIS problem has a trapdoor (for fixed \mathbf{A}): [\[GPV 08\]](#)

“Short” solutions to $\mathbf{A} \cdot \mathbf{x} = \mathbf{0} \Rightarrow$ can solve SIS for any \mathbf{u}

↳ Basis of lattice $\Lambda^\perp(\mathbf{A}) = \{ \mathbf{v} \mid \mathbf{A} \cdot \mathbf{v} = \mathbf{0} \}$

Spherical Gaussian Distributions



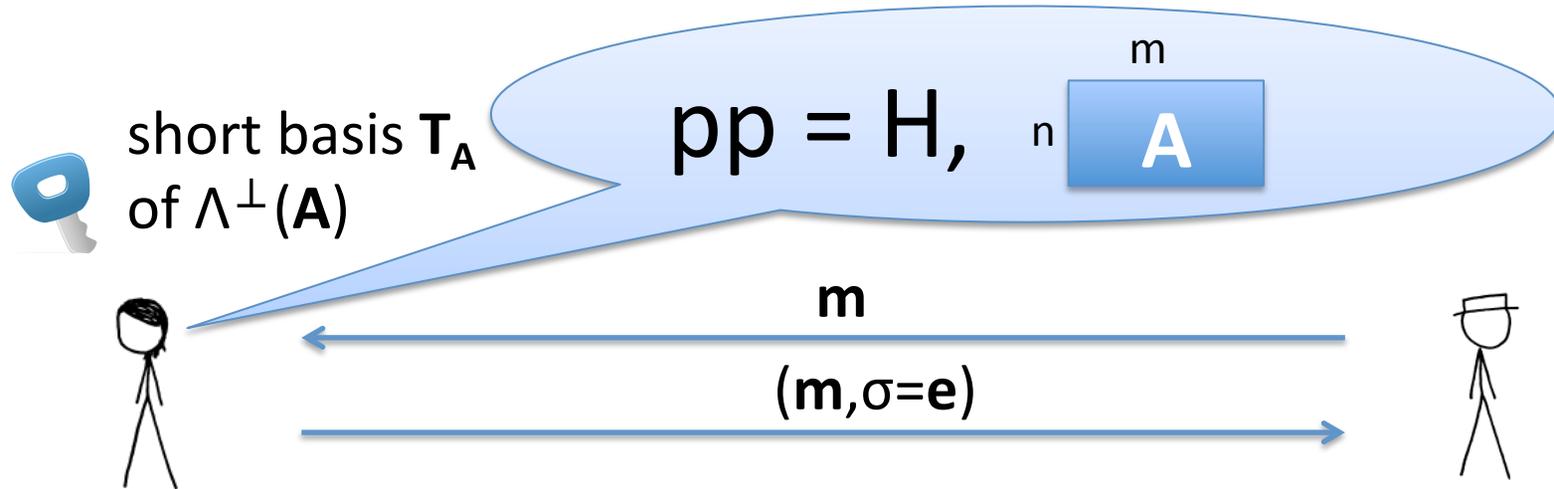
[images courtesy Peikert]

LWE: Looks “uniform” in \mathbf{R}^n when std-dev \geq **shortest basis**

Sampling Theorem: [GPV 08] Given basis \mathbf{T}_A of lattice Λ , can sample if std-dev $\geq \max |(T_A)_i|$

(leaks no information about \mathbf{T}_A)

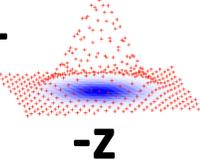
GPV signatures



Use T_A to
find short \mathbf{e} :
 $\mathbf{A} \cdot \mathbf{e} = H(\mathbf{m}) \pmod{q}$

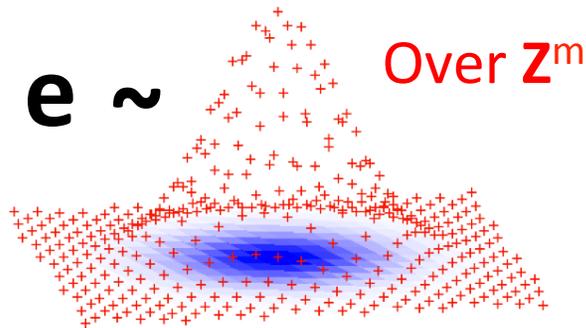
Verify given (\mathbf{m}, \mathbf{e}) that
 $\mathbf{A} \cdot \mathbf{e} = H(\mathbf{m}) \pmod{q}$
and \mathbf{e} is “short”

How? Any \mathbf{z} : $\mathbf{A} \cdot \mathbf{z} = H(\mathbf{m})$

Output $\mathbf{z} +$  $-\mathbf{z}$

Q: What does this
distribution look like?

Security



conditioned on
 $A \cdot e = H(m)$



IMP: Signatures independent of Alice's basis. Only depend on Λ .

Simulation proof in Random Oracle:

• To answer adversarial queries, pick e from discrete

Theorem (informal): Suppose that the SIS problem is hard, then in the random oracle model, GPV signatures are existentially unforgeable.

known "small" v

- $(\sigma^* - v)$ is a solution to SIS; short vector in Λ .

Lattice Delegation

$$\text{Basis of } \Lambda^\perp \left(\begin{array}{|c|} \hline \text{A} \\ \hline \end{array} \right) \Rightarrow \text{Basis of } \Lambda^\perp \left(\begin{array}{|c|c|} \hline \text{A} & \text{B} \\ \hline \end{array} \right)$$

$$\begin{array}{|c|c|} \hline \text{A} & \text{B} \\ \hline \end{array} \begin{array}{|c|c|} \hline \text{T}_A & \text{X} \\ \hline \text{0} & \text{I} \\ \hline \end{array} = \mathbf{0} \Rightarrow \begin{array}{|c|c|} \hline \text{A} & \text{X} \\ \hline \end{array} = - \begin{array}{|c|} \hline \text{B} \\ \hline \end{array}$$

More importantly: Can simulate!

Useful result: [CHKP10, ABB10a]

Without basis

of $\Lambda^\perp \left(\begin{array}{|c|} \hline \text{A} \\ \hline \end{array} \right)$

can output “random looking”

$\begin{array}{|c|} \hline \text{B} \\ \hline \end{array}$

with basis of $\Lambda^\perp \left(\begin{array}{|c|c|} \hline \text{A} & \text{B} \\ \hline \end{array} \right)$

ABB signatures



short basis T_A
of $\Lambda^\perp(A_0)$

$$pp = H, \quad A_0, \quad A_1, \quad A_2$$



$T_A \Rightarrow$ Basis of $\Lambda^\perp \left(\begin{array}{|c|c|} \hline A_0 & F_m \\ \hline \end{array} \right)$

$$F_m = A_1 + H(m) A_2$$



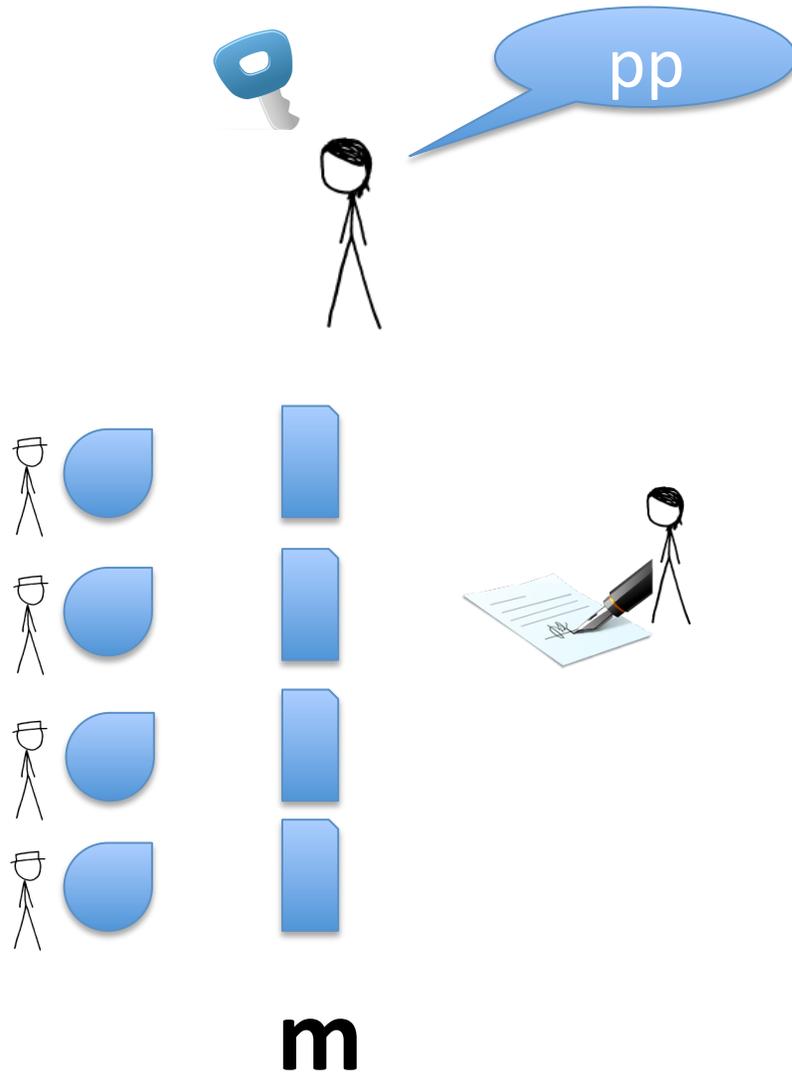
= “short vector” in $\Lambda^\perp \left(\begin{array}{|c|c|} \hline A_0 & F_m \\ \hline \end{array} \right)$

- Standard Model selectively-secure signatures
- Gives (H)IBE
- H maps messages to matrices with “full rank difference”

More schemes

- [CHKP10]: another signature and (H)IBE
 - \mathbf{F}_m is concatenation of matrices.
 - Longer public key.
 - Also selectively secure.
- [Boy10]:
 - \mathbf{F}_m is subset sum of appropriate matrices.
 - Fully secure signatures and IBE.
- [ABB10b]: signatures, IBE in **fixed dimension**

Threshold signatures



Correctness:

- Final signatures are EUF-CMA secure
 - Usually look like underlying signatures
- Any subset of t players can reconstruct a signature
- Do not reconstruct secret at any point in protocol
- No interactions between players

Security:

EUF-CMA security when adversary is given access to $t-1$ secret shares and signatures on chosen messages

Shamir Secret Sharing

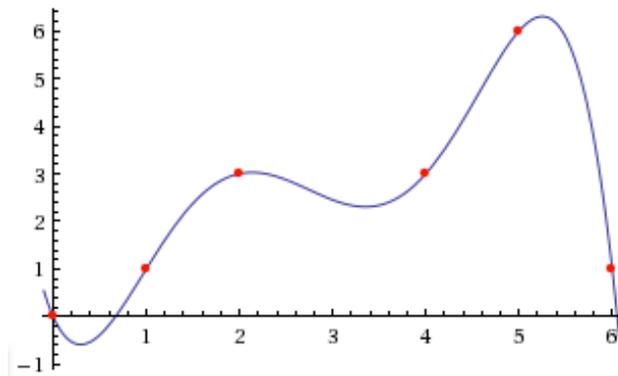
S

s_1

s_2

⋮

s_N



S

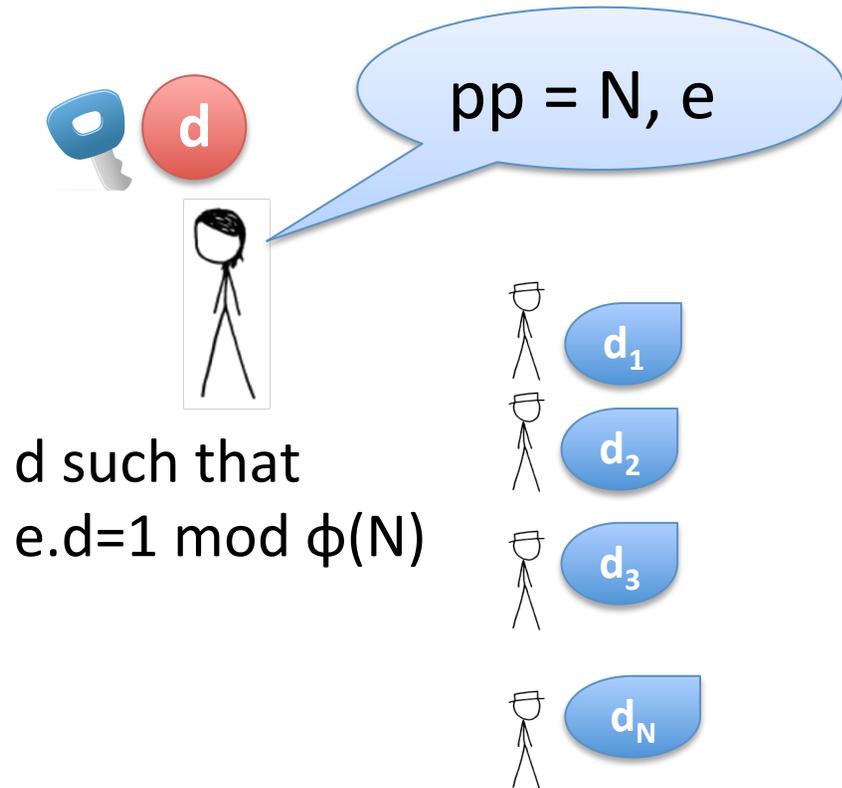
- (2,N) secret sharing
- In general (t,N) secret sharing

Properties:

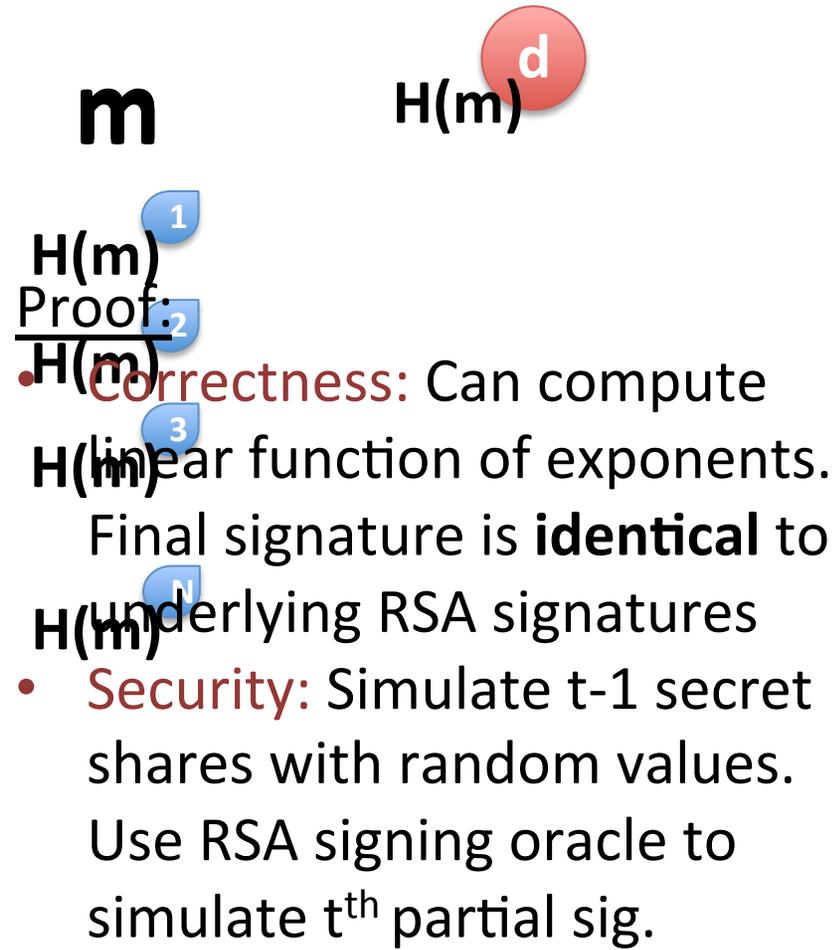
- $t-1$ shares **leak no information** about the secret
- Any **subset of t** players can reconstruct the secret

IMP: Secret reconstruction is a linear function of the secret shares

Threshold RSA signatures [Fra88, Sho00]

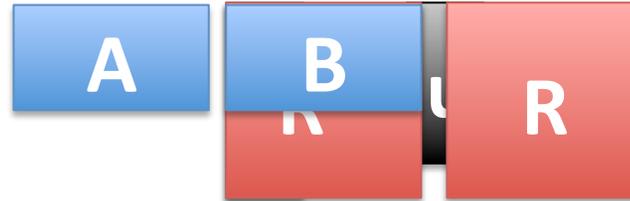


IMP: RSA allows linear function of partial sigs to be computed; final sig **identical** to underlying sig



A first attempt

Need: Linear scheme

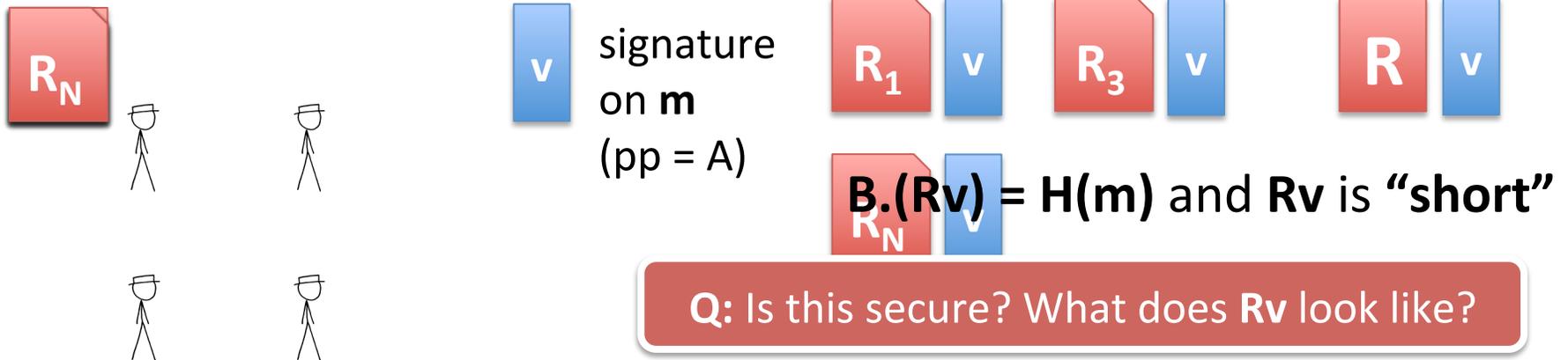


Basic Idea: Low norm transference matrix R

R transfers short vectors from $\Lambda^\perp(\mathbf{A})$ to $\Lambda^\perp(\mathbf{B})$ where $\mathbf{B}=\mathbf{A}\mathbf{R}^{-1}$

Basic idea behind [ABB10a, ABB10b, MP11]

Using GPV signatures: Shamir secret share R

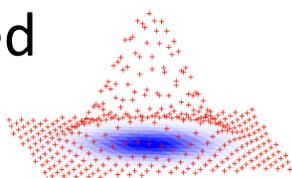


Skewed Gaussians

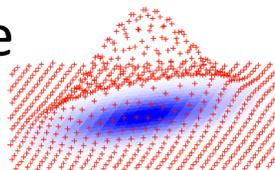
Recollect: In proof of GPV sigs, final distribution is **independent** of the secret (basis).

In our scheme, **Rv** is **not independent** of **R** .

Need



Have



In addition to shares of **R** , Alice gives **pre-shared randomness** [CG 99] to each player that is used to perturb



Adversary sees (v, Rv) for *many* values of v .
 m tuples can be used to **recover R** .
Thus, the scheme is **not secure**.

How to correct: Perturb with appropriately skewed Gaussian.
Require: Convolution lemma for discrete Gaussians.



Note: Still linear

Open Problems

- Re-use or eliminate pre-shared randomness
- Make robust without rounds of communication.
 - Will require new lattice-based NIZKs to prove partial signatures are well-formed
- Other efficient threshold constructions
 - Can you compress lattice trapdoors?
Given trapdoors for $\Lambda^\perp(\mathbf{A}|\mathbf{B})$ and $\Lambda^\perp(\mathbf{A}|\mathbf{C})$ efficiently compute trapdoor for $\Lambda^\perp(\mathbf{A})$.
 - Leads to other applications (possibly).



Thank you!
Any questions?

Arithmetic of Extension Fields of Small Characteristics

Recent Developments

Abhijit Das

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

Indo-US Workshop
Indian Statistical Institute, Calcutta
January 14, 2012

Finite Fields

- A *finite field* is a field with only finitely many elements.
- Any finite field contains p^n elements ($p \in \mathbb{P}$ and $n \in \mathbb{N}$).
- For any $p \in \mathbb{P}$ and $n \in \mathbb{N}$, there is a *unique* finite field of size p^n .
- Denote this field by \mathbb{F}_{p^n} .
- The prime p is the *characteristic* of the field.
- *Prime field*: $n = 1$.
- *Extension field*: $n \geq 2$.
- Cryptographic applications
 - Cryptosystems based on discrete logarithms
 - Cryptosystems based on elliptic curves
 - Cryptosystems based on pairing
- For security, fields \mathbb{F}_q with suitably large q are used.

Arithmetic of Prime Fields

- Take the field \mathbb{F}_p with a suitably large prime p .

- $\mathbb{F}_p = \{0, 1, 2, 3, \dots, p - 1\}$.

- Arithmetic in \mathbb{F}_p is the integer arithmetic modulo p .

- $$a + b \pmod{p} = \begin{cases} a + b & \text{if } a + b < p \\ a + b - p & \text{if } a + b \geq p \end{cases}$$

- $$a - b \pmod{p} = \begin{cases} a - b & \text{if } a \geq b \\ a - b + p & \text{if } a < b \end{cases}$$

- $$ab \pmod{p} = (ab) \text{ rem } p.$$

- Take $a \in \mathbb{F}_p$, $a \neq 0$. There exist integers u, v with $1 = ua + vp$. Then, $a^{-1} = u \pmod{p}$.

- Multiple-precision integer arithmetic is used to implement arithmetic.

- Computational hurdles

- Addition and subtraction: Carry management is clumsy

- Multiplication and division: Double-precision words needed

Arithmetic of Extension Fields

- Let $q = p^n$ with $p \in \mathbb{P}$ and $n \geq 2$.
- Choose a monic irreducible polynomial $f(x) \in \mathbb{F}_p[x]$ of degree n .
- $f(x)$ is called the *defining polynomial*.
- $\mathbb{F}_q = \mathbb{F}_p[x]/\langle f(x) \rangle$.
- $\mathbb{F}_q = \{a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1} \mid a_i \in \mathbb{F}_p\}$.
- Arithmetic in \mathbb{F}_q is the polynomial arithmetic of $\mathbb{F}_p[x]$ modulo $f(x)$.
- Is it simpler than arithmetic of prime fields of similar sizes?
- In general, no.
- Special case $p = 2$: An element of \mathbb{F}_q is a bit vector of size n .
- Special case $p = 3$: An element of \mathbb{F}_q is two bit vectors of size n .
- Computational advantages for $p = 2, 3$:
 - No carry management
 - No double-precision words needed
 - Bit-wise operations suffice

Binary Fields

- \mathbb{F}_q with $q = 2^n$.
- Choose the defining polynomial $f(x)$ with as few non-zero coefficients as possible.
- $\alpha, \beta \in \mathbb{F}_q$ are bit vectors.
- Addition is bit-wise XOR.
- Multiplication is $(\alpha\beta) \bmod f(x)$ (polynomial multiplication followed by polynomial division).
- Squaring of α is $\alpha^2 \bmod f(x)$. Computing α^2 is easier than computing $\alpha\beta$.
- Modular reduction is efficient for sparse $f(x)$.
- Inverse is computed by extended gcd of polynomials. For $\alpha \in \mathbb{F}_q$, $\alpha \neq 0$, compute polynomials $u, v \in \mathbb{F}_q[x]$ such that $u\alpha + vf = 1$. Then $\alpha^{-1} = u \pmod{f}$.

Fast Multiplication in Binary Fields

Karatsuba-Ofman Multiplication

- Write $\alpha = x^m \alpha_1 + \alpha_0$ and $\beta = x^m \beta_1 + \beta_0$, where $m = \lceil n/2 \rceil$.
- $\alpha_1, \alpha_0, \beta_1, \beta_0$ are of degrees $\leq m - 1$.
- Compute three subproducts $\alpha_1 \beta_1, \alpha_0 \beta_0, (\alpha_1 + \alpha_0)(\beta_1 + \beta_0)$.
- $\alpha \beta = (\alpha_1 \beta_1) x^{2m} + [(\alpha_1 + \alpha_0)(\beta_1 + \beta_0) + \alpha_1 \beta_1 + \alpha_0 \beta_0] x^m + (\alpha_0 \beta_0)$.
- Subproducts can be computed recursively by Karatsuba-Ofman method.
- **Question:** How about Karatsuba-Ofman in fields of characteristic three?
- **Question:** Other fast multiplication algorithms?
- Toom-3: Directly applicable for $p \geq 5$.
- FFT: Apparently not effective for fields of cryptographic sizes.

1

A. Karatsuba and Yu. Ofman, *Multiplication of many-digital numbers by automatic computers*, Doklady Akad. Nauk. SSSR, Vol. 145, 293–294, 1962.

2

S. Ghosh, D. Roy Chowdhury and A. Das, *High speed cryptoprocessor for eta pairing on 128-bit secure supersingular elliptic curves over characteristic two fields*, CHES, Nara, Japan, 2011.

Fast Multiplication in Binary Fields

Comb Multiplication

- Precompute $x^j\alpha$ for $j = 0, 1, 2, \dots, w - 1$ (where w is the word size).
- Take $i \in \{0, 1, 2, \dots, n - 1\}$.
- Write $i = j + kw$.
- Add the j -th precomputed polynomial starting from k -th word.
- Other variants
 - Windowed comb method
 - Left-to-right comb method
- **Question:** Effectiveness in hardware implementations?

1 J. López and R. Dahab, *High-speed software multiplication in \mathbb{F}_{2^m}* , INDOCRYPT, 203–212, 2000.

Fast Modular Reduction in Binary Fields

- Take $f(x) = x^n + f_1(x)$ with:
 - 1 $f_1(x)$ has as few non-zero terms as possible,
 - 2 $\deg f_1(x)$ is as small as possible.
- Example: Irreducible trinomials and pentanomials for binary fields.
- Canceling the highest non-zero term in the long division process is effected by setting that coefficient to zero, and by adding a suitable shift of $f_1(x)$.
- If $\deg f_1 \ll n$, word-level XOR operations reduce complete words.
- **Question:** No straightforward adaptations of Montgomery and Barrett reductions are known.

Inverse in Binary Fields

- To compute α^{-1} , where $\alpha \in \mathbb{F}_{2^n}$.
- **Euclidean inverse:** Repeated long divisions of polynomials.
- **Binary inverse:** Maintains the invariance

$$\begin{aligned}u_1\alpha + v_1f &= r_1, \\u_2\alpha + v_2f &= r_2.\end{aligned}$$

In each iteration, replace r_1 or r_2 by $r_1 + r_2$ and correspondingly u_1 or u_2 by $u_1 + u_2$. Remove powers of x from r_1 or r_2 (and u_1 or $u_1 + f$ or u_2 or $u_2 + f$).

- **Almost inverse:** Maintains the invariance

$$\begin{aligned}u_1\alpha + v_1f &= x^k r_1, \\u_2\alpha + v_2f &= x^k r_2,\end{aligned}$$

for some k . Each iteration is similar to as in binary inverse except that $u_1 + f$ or $u_2 + f$ is not computed, but the exponent k is adjusted.

Fields of Characteristic Three

- Two bits are needed to encode the elements 0, 1, 2 of \mathbb{F}_3 .
- An element of \mathbb{F}_{3^n} is represented by two bit-vectors of length n .
- Bit-wise operations perform addition on these bit vectors.
- Natural encoding $(0, 0) \mapsto 0$, $(0, 1) \mapsto 1$ and $(1, 0) \mapsto 2$ requires seven bit-wise instructions.
- The encoding $(1, 1) \mapsto 0$, $(0, 1) \mapsto 1$ and $(1, 0) \mapsto 2$ requires six bit-wise instructions.
- No encoding can manage in less than six instructions.
- Karatsuba-Ofman and comb methods apply to multiplication.
- Modular reduction is efficient for $f(x) = x^n + f_1(x)$ with f_1 as sparse and low-degree as possible.
- **Question:** Efficient hardware implementations?

1

K. Harrison, D. Page and N. P. Smart, *Software implementation of finite fields of characteristic three*, LMS Journal of Computation and Mathematics, 5:181–193, 2002.

2

Y. Kawahara, K. Aoki and T. Takagi, *Faster implementation of η_T pairing over $GF(3^m)$ using minimum number of logical instructions for $GF(3)$ -addition*, Pairing, 283–296, 2008.

Optimal Extension Fields

- Fields of the form \mathbb{F}_{p^n} , where
 - p fits in a machine word,
 - $p = 2^n + c$ with $|c| \leq 2^{\lfloor n/2 \rfloor}$, and
 - we can take a defining polynomial of the form $x^n - \omega \in \mathbb{F}_p[x]$.
- Reduction in \mathbb{F}_p is efficient (one addition only) if $c = \pm 1$ (Type I fields).
- Polynomial reduction in \mathbb{F}_{p^n} involves replacing x^i by $x^{i-n}\omega$ for $2n - 2 \leq i \leq n$.
- OEFs are easy to find.
- **Question:** Efficient software and hardware implementations.

1

P. Mihăilescu, *Optimal Galois field bases which are not normal*, presented in FSE, 1997.

2

D. V. Bailey and C. Paar, *Optimal extension fields for fast arithmetic in public key algorithms*, Crypto, 472–485, 1998.

Towers of Extensions

- Pairing computations require working in extension \mathbb{F}_{q^m} , where q is already of the form 2^n or 3^n .
- m is usually small. Example: $\mathbb{F}_{(2^n)^4}$ and $\mathbb{F}_{(3^n)^6}$.
- Addition and subtraction in \mathbb{F}_{q^m} are straightforward.
- Multiplication in \mathbb{F}_{q^m} boils down to a sequence of multiplications in \mathbb{F}_q .
- Challenge: To reduce the number of \mathbb{F}_q -multiplications.
- Consider the extensions $\mathbb{F}_{3^n} \subseteq \mathbb{F}_{3^{2n}} \subseteq \mathbb{F}_{3^{6n}}$.
- Each $\mathbb{F}_{3^{6n}}$ -multiplication reduces to five $\mathbb{F}_{3^{2n}}$ -multiplications.
- Apply Karatsuba-Ofman strategy for each multiplication in $\mathbb{F}_{3^{2n}}$.
- Fifteen \mathbb{F}_{3^n} -multiplications suffice for each $\mathbb{F}_{3^{6n}}$ -multiplication.
- **Question:** Is this optimal?

Parallelization Platforms

Distributed parallelization

- Cheap. No extra computing hardware needed.
- Communication demands high-speed links. Still delay may be high.

Multi-core parallelization

- Cost varies of the number of cores.
- Communication is via shared memory.
- Synchronization may be problematic for fine-grained parallelism.

SIMD parallelization

- SIMD registers are available in many cheap processors.
- No synchronization overhead.
- Packing/unpacking from/to normal registers may be an overhead.
- Suited to fine-grained parallelization.
- Not effective for all algorithms.

GPU parallelization

- May be expensive.
- Suited usually to floating-point calculations.
- Crypto algorithms typically cannot exploit full potential.

Parallelization Possibilities

- **Cryptanalytic algorithms** are happy with coarse-grained parallelism.
 - Multi-core parallelization would be the best platform.
 - Even distributed parallelization may be practical.
 - **Question:** SIMD may additionally speed up multi-core implementations.
-
- **Cryptographic procedures** demand fine-grained parallelism.
 - Distributed parallelization is usually extremely inefficient.
 - Poor speedup is achieved if we divide each operation (like exponentiation or pairing computation) among multiple cores, synchronization overheads being abnormally high.
 - It is preferable to schedule different operations to different cores.
 - Large prime fields are crippled by carries and double-precision words.
 - Extension fields of small characteristics can exploit SIMD and GPU parallelization with some effectiveness.
 - The current technological developments renewed interests in extension fields of characteristics two and three.

SIMD and GPU Parallelization of Extension-field Arithmetic

- Potentially effective even at the level of each field operation.

Intra-operation (horizontal) parallelization

- Each individual field operation is parallelized.
- May be associated with some packing and unpacking overhead.
- Promising for fields of characteristics two and three.

Inter-operation (vertical) parallelization

- Multiple field operations of the same type are simultaneously parallelized.
- The different operations should follow (nearly) identical instructions.
- Appears to be more promising for fields of characteristics two and three.

Hybrid parallelization

- Both intra- and inter-operation parallelization techniques are combined.
- Some papers report effective use of hybrid parallelization.
- May be more suited to GPU platforms.

Some Open Research Problems

- Faster (than reported) implementations of finite-field arithmetic in both software and hardware.
- Faster implementations of compound primitives based on finite-field arithmetic (like pairing).
- Efficient parallelization.
- Proving lower bounds on counts of arithmetic operations in base fields.
- Attention to fields of small characteristics ≥ 5 .
- Attention to the arithmetic of optimal extension fields.
- Compound primitives based on fields of small characteristics (like finding families of pairing-friendly curves over fields of small characteristics and optimal extension fields).

Thanks for Your Attention!

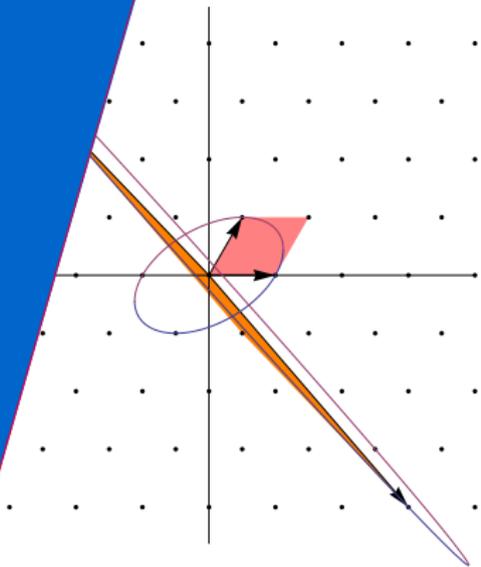
Hard Lattice Problems

Benne de Weger

(inspired by Joop van de Pol's MSc Thesis, 2011)

b.m.m.d.weger@tue.nl

Kolkata, India, Jan. 12, 2012



TU / **e**

Technische Universiteit
Eindhoven
University of Technology

January 11, 2012

Where innovation starts

A *lattice* L is a discrete additive subgroup of \mathbb{R}^n . It has a *basis*:

$$L(\mathbf{b}_1, \dots, \mathbf{b}_m) = \left\{ \sum_{i=1}^m x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}.$$

The *rank* is m , the lattice is *full rank* if $m = n$.

With the basis vectors as columns in the matrix \mathbf{B} :

$$L(\mathbf{B}) = \{ \mathbf{B}\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^m \}.$$

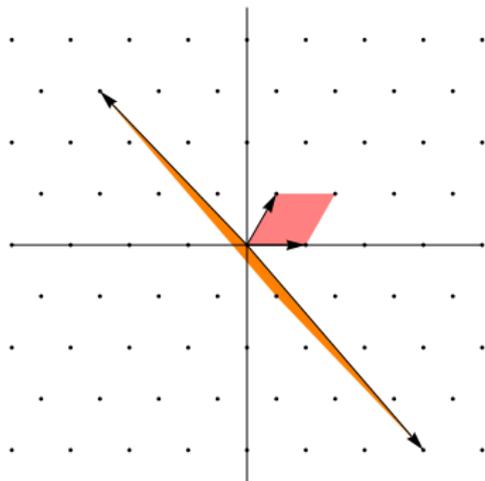
$L(\mathbf{B}_1) = L(\mathbf{B}_2)$ if and only if $\mathbf{B}_2 = \mathbf{B}_1 \mathbf{U}$ with \mathbf{U} an $m \times m$ unimodular matrix.

The *volume* $\text{vol}(L(\mathbf{B}))$ is the m -dimensional volume of $\{\mathbf{B}\mathbf{x} \mid \mathbf{x} \in [0, 1]^m\}$.

$$\text{vol}(L(\mathbf{B})) = \sqrt{\det(\mathbf{B}^\top \mathbf{B})}.$$

$\text{vol}(L(\mathbf{B})) = |\det(\mathbf{B})|$ in the full rank case.

It is independent of the basis choice, so $\text{vol}(L)$ is defined.



Lattice problems are about *distances (lengths)*.

Distances are defined in terms of *norms*:

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|.$$

A *norm* on \mathbb{R}^n is a function $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}$ satisfying

- ▶ $\|\mathbf{x}\| > 0$ for all $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$, and $\|\mathbf{0}\| = 0$,
- ▶ $\|t\mathbf{x}\| = |t|\|\mathbf{x}\|$ for all $\mathbf{x} \in \mathbb{R}^n$, $t \in \mathbb{R}$,
- ▶ $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

Main example: Euclidean norm $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^\top \mathbf{x}}$, many other examples exist.

Theory of lattice problems and lattice (basis reduction) algorithms can be set up for general norms.

For a full rank basis with matrix \mathbf{B} , there is a very nice norm:

$$\|\mathbf{x}\|_{\mathbf{B}} = \sqrt{\mathbf{x}^{\top} \mathbf{B}^{-1} \mathbf{B}^{-1} \mathbf{x}}.$$

Indeed, w.r.t. the inner product $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{B}} = \mathbf{x}^{\top} \mathbf{B}^{-1} \mathbf{B}^{-1} \mathbf{y}$ the basis is orthonormal

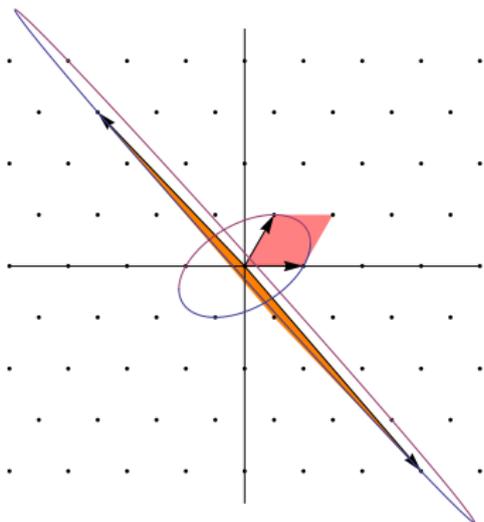
($\|\mathbf{x}\|_{\mathbf{B}} = 1$ is the ellipsoid containing the basis vectors)

Of course, this is not playing fair...

In practice one has a fixed norm (usually Euclidean)

and a basis that is bad w.r.t. this norm

and one wants a basis that is as good as reachable w.r.t. this norm



From now on, for convenience, only full rank, only Euclidean norm, assume $\text{vol}(L) \gg 1$.

Enumeration: Given lattice L and radius r , find all $\mathbf{y} \in L$ s.t. $\|\mathbf{y}\| \leq r$.
 L is given by a basis with matrix \mathbf{B} , so: find all $\mathbf{x} \in \mathbb{Z}^n$ s.t. $\|\mathbf{B}\mathbf{x}\| \leq r$.
Lemma: there exists a nonzero solution when $r > 2^{(n-1)/4} \text{vol}(L)^{1/n}$.

This looks easy: compute $\mathbf{U} = \mathbf{B}^{-1}$, write $\|\mathbf{U}\| = \max |U_{i,j}|$,

then $\|\mathbf{x}\| = \|\mathbf{U}\mathbf{y}\| \leq n\|\mathbf{U}\|\|\mathbf{y}\| \leq n\|\mathbf{U}\|r$,

enumerate over all $\mathbf{x} \in \mathbb{Z}^n$ with $\|\mathbf{x}\| \leq n\|\mathbf{U}\|r$.

Complexity: $(2n)^n \|\mathbf{U}\|^n r^n$, for one solution only: $2^{n(n+3)/4} n^n \|\mathbf{U}\|^n \text{vol}(L)$.

[[Slightly more efficient: use Gram-Schmidt to orthogonalize the basis and enumerate variables in the right order, basically this removes n^n .]]

Clearly we need small $\|\mathbf{U}\|$.

Though $|\det \mathbf{U}| = 1/\text{vol}(L)$ is small, $\|\mathbf{U}\|$ can be (relatively) very big (≈ 1).

Intuition: it gets smaller when \mathbf{B} gets closer to orthogonal.

There exist algorithms that compute good bases for which complexity becomes approximately $2^n r^n / \text{vol}(L)$, for one solution only: $2^{n(n+3)/4}$.

Heuristically these algorithms (Lattice Basis Reduction: LLL, BKZ) even give q^{n^2} for q as small as 1.013.

Heuristically, no algorithm can do better than $n^{n/8}$.

There are algorithms that compute in polynomial time a 'reduced' basis, notably LLL and BKZ.

Such a basis is useful for:

- ▶ finding lattice vectors of short length
 - just take reduced basis vectors
- ▶ finding lattice vectors close to a target vector \mathbf{t} outside the lattice
 - Babai's rounding technique: $\mathbf{B} \lfloor \mathbf{B}^{-1} \mathbf{t} \rfloor$ is a good guess
 - Babai's nearest plane technique: project \mathbf{t} to a suitably close hyperplane and iterate

Unfortunately (or, if you wish, fortunately) 'reduced' is not optimal.

The *first successive minimum* of the lattice L is

$$\lambda_1(L) = \min_{\mathbf{x} \in L, \mathbf{x} \neq \mathbf{0}} \|\mathbf{x}\|.$$

SVP - Shortest Vector Problem

Given a basis of L ,
find $\mathbf{y} \in L$ such that $\|\mathbf{y}\| = \lambda_1(L)$.

SVP $_\gamma$ - Approximate Shortest Vector Problem

Given a basis of L and an approximation factor $\gamma \geq 1$,
find $\mathbf{y} \in L$ such that $0 < \|\mathbf{y}\| \leq \gamma \lambda_1(L)$.

SVP is NP-hard for $\gamma = 2^{\log^{1/2-\epsilon} n}$

LLL solves SVP $_\gamma$ in polynomial time for $\gamma = (4/3 + \epsilon)^{(n-1)/2}$

HSVP $_{\gamma}$ - Hermite Shortest Vector Problem

Given a basis of L and an approximation factor $\gamma > 0$,
find $\mathbf{y} \in L$ such that $0 < \|\mathbf{y}\| \leq \gamma \text{vol}(L)^{1/n}$.

Note that $\lambda_1(L)$ is in general not known, but $\text{vol}(L)$ is known.

LLL solves HSVP $_{\gamma}$ in polynomial time for $\gamma = (4/3 + \epsilon)^{(n-1)/4}$.

LLL achieves in practice $\gamma = 1.022^n$, even 1.01^n might be within reach.

SLP $_{\gamma}$ - Shortest Length Problem

Given a basis of L and an approximation factor $\gamma > 0$,
find a λ with $\lambda_1(L) \leq \lambda \leq \gamma \lambda_1(L)$ such that there is a $\mathbf{y} \in L$ with $\|\mathbf{y}\| = \lambda$.

Note that this problem does not require that \mathbf{y} becomes known.

The i th successive minimum of the lattice L is

$$\lambda_i(L) = \min_{\mathbf{x}_1, \dots, \mathbf{x}_i \in L \text{ lin. indep.}} \max\{\|\mathbf{x}_1\|, \dots, \|\mathbf{x}_i\|\}.$$

SMP - Successive Minima Problem

Given a basis of L ,

find $\mathbf{y}_1, \dots, \mathbf{y}_n \in L$ such that $\|\mathbf{y}_i\| = \lambda_i(L)$ for $i = 1, \dots, n$.

SBP $_\gamma$ - Shortest Basis Problem

Given a basis of L and an approximation factor $\gamma \geq 1$,

find a basis $\mathbf{b}_1, \dots, \mathbf{b}_n$ of L with $\max_i \|\mathbf{b}_i\| \leq \gamma \min_{\mathbf{b}'_1, \dots, \mathbf{b}'_n \text{ basis}} \max_i \|\mathbf{b}'_i\|$.

DSVP - Decision Shortest Vector Problem

Given a basis of L and a radius $r > 0$,
decide whether there exists a $\mathbf{y} \in L$ such that $0 < \|\mathbf{y}\| \leq r$.

USVP $_{\gamma}$ - Unique Shortest Basis Problem

Given a basis of L and a gap factor $\gamma \geq 1$,
find (if it exists) the unique $\mathbf{y} \in L$ such that any $\mathbf{v} \in L$ with $\|\mathbf{v}\| \leq \gamma \|\mathbf{y}\|$ is
an integral multiple of \mathbf{y} .

This is equivalent to the SVP for lattices with $\lambda_2(L) > \gamma \lambda_1(L)$.

$d(\mathbf{t}, L)$ denotes the distance of $\mathbf{t} \in \mathbb{R}^n$ to the closest lattice vector.

CVP - Closest Vector Problem

Given a basis of L and a target $\mathbf{t} \in \mathbb{R}^n$,
find $\mathbf{y} \in L$ such that $\|\mathbf{t} - \mathbf{y}\| = d(\mathbf{t}, L)$.

CVP_γ - Approximate Closest Vector Problem

Given a basis of L , a target $\mathbf{t} \in \mathbb{R}^n$ and an approximation factor $\gamma \geq 1$,
find $\mathbf{y} \in L$ such that $\|\mathbf{t} - \mathbf{y}\| \leq \gamma d(\mathbf{t}, L)$.

It makes sense to assume that $\mathbf{t} \notin L$.

CVP_γ is probably NP-hard for $\gamma = 2^{\log^{1-\epsilon} n}$

Babai nearest plane (based on LLL) solves CVP_γ in polynomial time for
 $\gamma = 2(2/\sqrt{3})^n$.

CVP is at least as hard as SVP.

Idea: view SVP as CVP in suitable sublattices.

SVP_{γ} is heuristically at least as hard as $CVP_{\gamma'}$.

Idea: embedding technique: CVP_{γ} of dimension n with target vector \mathbf{t} and basis \mathbf{B} can be attacked by $SVP_{\gamma'}$ of dimension $n + 1$ with basis

$$\begin{pmatrix} \mathbf{t} & \mathbf{B} \\ 1 & \mathbf{0} \end{pmatrix}.$$

DCVP - Decision Closest Vector Problem

Given a basis of L , a target vector $\mathbf{t} \in \mathbb{R}^n$ and a radius $r > 0$, decide whether there exists a $\mathbf{y} \in L$ such that $\|\mathbf{y} - \mathbf{t}\| \leq r$.

CVPP - Closest Vector Problem with Preprocessing

Given a basis of L and a preprocessing function that returns another basis of L whose size is polynomially related to the input basis, solve CVP for the new basis.

Preprocessing can e.g. be lattice basis reduction.

CVPP is NP-hard.

GapSVP $_{\gamma}$ - Gap Shortest Vector Problem

Given a basis of L , a radius $r > 0$ and an approximation factor $\gamma > 1$,
return YES if $\lambda_1(L) \leq r$,
return NO if $\lambda_1(L) > \gamma r$,
and return either YES or NO otherwise.

Such a problem is called a *promise* problem.

BDD $_{\alpha}$ - Bounded Distance Decoding

Given a basis of L , a distance parameter $\alpha > 0$
and a target vector $\mathbf{t} \in \mathbb{R}^n$ such that $d(\mathbf{t}, L) < \alpha \lambda_1(L)$,
find a $\mathbf{y} \in L$ such that $d(\mathbf{y}, \mathbf{t}) = d(L, \mathbf{t})$.

This is NP-hard for $\alpha > 1/\sqrt{2}$.

SIS - Small Integer Solution

Given a modulus q , a matrix $\mathbf{A} \pmod{q}$ and a $\nu < q$,
find $\mathbf{y} \in \mathbb{Z}^n$ such that $\mathbf{A}\mathbf{y} \equiv \mathbf{0} \pmod{q}$ and $\|\mathbf{y}\| \leq \nu$.

Note that this is a lattice problem, as $\{\mathbf{x} \mid \mathbf{A}\mathbf{x} \equiv \mathbf{0} \pmod{q}\}$ is a lattice.
This problem is closely related to SVP and to USVP.

SIVP $_{\gamma}$ - Shortest Independent Vector Problem

Given a basis of L and an approximation factor $\gamma > 1$,
find a linearly independent set $\mathbf{y}_1, \dots, \mathbf{y}_n$ such that $\max \|\mathbf{y}_i\| \leq \gamma \lambda_n(L)$.

This is closely related to SBS.

Let q be a modulus. For $\mathbf{s} \in \mathbb{Z}_q^n$ and a probability distribution χ on \mathbb{Z}_q , let $A_{\mathbf{s}, \chi}$ be the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ with sampling as follows:
take $\mathbf{a} \in \mathbb{Z}_q^n$ uniform,
take $e \in \mathbb{Z}_q$ according to χ ,
then return $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \pmod{q}$.

LWE - Learning With Errors

Given n, q, χ and any number of independent samples from $A_{\mathbf{s}, \chi}$, find \mathbf{s} .

LPN - Learning Parity with Noise - is just LWE for $q = 2$.
For χ the discrete Gaussian distribution is a good choice.

For the sampled (\mathbf{a}_i, b_i) let \mathbf{A} be the matrix of the \mathbf{a}_i , and let \mathbf{b} be the vector of the b_i . Then $\{\mathbf{x} \mid \mathbf{x} \equiv \mathbf{A}^\top \mathbf{y} \pmod{q} \text{ for some } \mathbf{y}\}$ is a lattice in which $\mathbf{A}^\top \mathbf{s}$ is close to \mathbf{b} , so this is a CVP-variant, in fact, a BDD-variant. The LWE-lattice is the dual of the SIS-lattice.

DLWE - Decision Learning With Errors

Given n, q, χ and any number of independent samples from $A_{s,\chi}$, return YES if the samples come from $A_{s,\chi}$, and NO if they come from the normal distribution.

Ring-LWE

As with NTRU, one induces additional structure in the samples

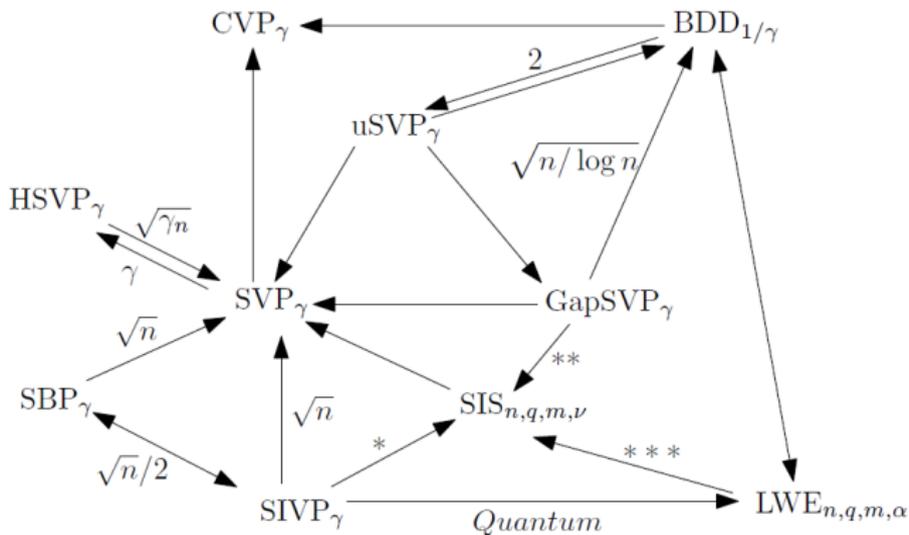


Figure 3.1: Relations among lattice problems.

(picture from Joop van de Pol's MSc-thesis)

Many other variants of lattice problems can be described.

One interesting kind: Cryptosystems usually have additional structure in the lattices. Examples: NTRU, *ideal* lattices (used e.g. in FHE).

A vector \mathbf{b} can be associated to a polynomial $\sum b_i x^i$ in a ring $\mathbb{Z}[x]/\langle f \rangle$ for a monic polynomial f of degree n .

Ideal lattices correspond in this sense with ideals in $\mathbb{Z}[x]/\langle f \rangle$.

The choice of f may have consequences for the reductions.

E.g. for ideal lattices in $\mathbb{Z}[x]/\langle x^n + 1 \rangle$ SVP becomes equivalent to SIVP, and $\text{GapSVP}_{\sqrt{n}}$ becomes trivial.

Many open questions in this area.

Approximate common divisors via lattices

Nadia Heninger

UC San Diego

January 13, 2012

Approximate common divisors

Factoring with bits known

Let $N = pq$.

Divide bits of p into halves p_ℓ, p_r :

$$p_\ell \cdot 2^{n/2} + p_r = p$$

Given N, p_ℓ , find p .

Approximate common divisor

Given $pq_1 + r_1, \dots, pq_m + r_m$, find p .

“Partial approximate common divisors”

Theorem (Howgrave-Graham)

Given a , N integers, can find all x_0 , such that

$$\gcd(a - x_0, N) \geq N^\beta$$

$$|x_0| \leq N^{\beta^2}$$

in time polynomial in $\log N$.

“General approximate common divisors”

Theorem (Howgrave-Graham)

Given a, b integers, can find all x_0, y_0 such that

$$\gcd(a - x_0, b - y_0) \geq N^\beta$$

$$|x_0|, |y_0| \leq N^{\frac{3}{8}\beta^2}$$

in time polynomial in $\log N$.

Proof outline.

Proof outline.

1. Create a polynomial $Q(x)$ so that
all desired x_0 are roots of Q over \mathbb{Z} .
2. Factor Q to find roots.



Proof idea

1. Create a new polynomial $Q(x)$ so that
all desired $x_0 < X$ are roots of Q over \mathbb{Z} .

- a. Look for Q in ideal $\langle x - a, N \rangle^k$.

$$(Q(x) = \sum c_{ij} x^j (x - a)^i N^{k-i} \implies Q(x_0) \equiv 0 \pmod{p^k}.)$$

- b. Find $Q(x) = \sum_i q_i x^i$ with small coefficients, so that

$$|Q(x_0)| \leq \sum_i |q_i| X^i < N^{\beta k} \leq p^k$$

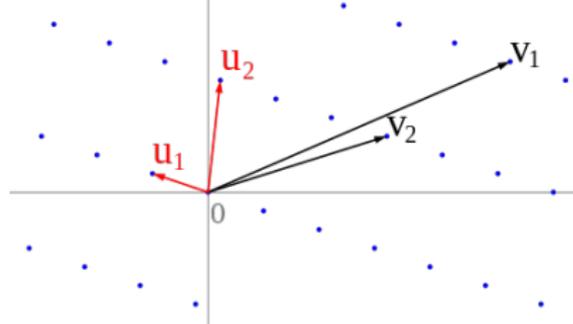
and thus x_0 is a root of Q over \mathbb{Z} .

- c. How to find Q with small coefficients?

Lattice basis reduction.

Lattice basis reduction

For these theorems, only need exponential approximation factor.



LLL: Can find a vector of length $|v| \leq 2^{\dim L/4} \det L^{1/\dim L}$ in polynomial time.

In this application,

integral basis for ideal \longrightarrow **basis for our lattice**

Concretely,

- lattice vectors are the coefficient vectors of our polynomials
- short lattice vectors represent polynomials with small coefficients

Multivariate approximate common divisors

Joint with Henry Cohn

Theorem (Partial common divisors)

Given $a_1 = pq_1 + r_1, \dots, a_m = pq_m + r_m, N = pq$, can find r_1, \dots, r_m when

$$p \geq a_i^\beta \quad a_i^{\beta^2} \gg 1$$

$$|r_i| \leq a_i^{(1+o(1))\beta(m+1)/m}$$

in time polynomial in $\log a_i$ and exponential in m .

Multivariate approximate common divisors

Joint with Henry Cohn

Theorem (General common divisors)

Given $a_1 = pq_1 + r_1, \dots, a_m = pq_m + r_m$, can find r_1, \dots, r_m when

$$p \geq a_i^\beta \quad a_i^{\beta^2} \gg 1$$

$$|r_i| \leq a_i^{(1 - \frac{\log m}{m})\beta^{m/(m-1)}}$$

in time polynomial in $\log a_i$ and exponential in m .

Proof outline

Proof outline.

1. Create $Q_1(x_1, \dots, x_m), \dots, Q_m(x_1, \dots, x_m)$ so that

$$Q_i(r_1, \dots, r_m) = 0 \text{ over } \mathbb{Z}.$$

2. Look for $Q_i \in \langle a_1 - x_1, \dots, a_m - x_m, N \rangle^k$.

3. Solve system to find roots.



Some possible issues

Need m short vectors instead of just one.

Some possible issues

Need m short vectors instead of just one.

Can bound

$$|v_1| \leq \dots \leq |v_m| \leq 2^{(\dim L)/4} \det L^{1/(\dim L+1-m)}.$$

Becomes $1 + o(1)$ in exponent of result.

Some possible issues

Need m short vectors instead of just one.

Can bound

$$|v_1| \leq \dots \leq |v_m| \leq 2^{(\dim L)/4} \det L^{1/(\dim L+1-m)}.$$

Becomes $1 + o(1)$ in exponent of result.

In practice, lattice acts like random lattice, so

$$|v_i| \approx 1.02^{\dim L} \det L^{1/\dim L}.$$

Some possible issues

m shortest vectors might not be algebraically independent.

Some possible issues

m shortest vectors might not be algebraically independent.

In theorem, requires heuristic assumption.

Some possible issues

***m* shortest vectors might not be algebraically independent.**

In theorem, requires heuristic assumption.

In practice, can always solve system.

- ▶ Algebraic dependencies in experiments always resulted from an obvious sublattice.
- ▶ Could always keep adding short vector equations to the system until it can be solved.

Some possible issues

***m* shortest vectors might not be algebraically independent.**

In theorem, requires heuristic assumption.

In practice, can always solve system.

- ▶ Algebraic dependencies in experiments always resulted from an obvious sublattice.
- ▶ Could always keep adding short vector equations to the system until it can be solved.

Solve using Gröbner bases to eliminate variables. Slow in theory, but in practice very fast to solve overconstrained system.

Some possible issues

Exponential approximation factor from LLL.

Some possible issues

Exponential approximation factor from LLL.

$$|Q(r_1, \dots, r_m)| \leq |v_1| \leq \sqrt{\dim L} 2^{\dim L} \det L^{1/\dim L} < N^{\beta k}$$

Some possible issues

Exponential approximation factor from LLL.

$$|Q(r_1, \dots, r_m)| \leq |v_1| \leq \sqrt{\dim L} 2^{\dim L} \det L^{1/\dim L} < N^{\beta k}$$

$$\frac{t^m}{k} + \log X \frac{t}{k} + \log N \frac{k^m}{t^m} < \beta \log N$$

\swarrow $t^m < \beta k \log N$ \searrow $k^m < \beta t^m$

$$k^m < \beta t^m < \beta^2 k \log N$$

Requirement $\beta^2 \log N \gg 1$.

Some possible issues

Exponential approximation factor from LLL.

$$|Q(r_1, \dots, r_m)| \leq |v_1| \leq \sqrt{\dim L} 2^{\dim L} \det L^{1/\dim L} < N^{\beta k}$$

$$\frac{t^m}{k} + \log X \frac{t}{k} + \log N \frac{k^m}{t^m} < \beta \log N$$

\swarrow $t^m < \beta k \log N$ \searrow $k^m < \beta t^m$

$$k^m < \beta t^m < \beta^2 k \log N$$

Given lattice basis reduction algorithm with approximation factor $2^{\dim L^\epsilon}$, get

$$\beta^{1+\epsilon} \log N \gg 1$$

Application to fully homomorphic encryption parameters

[van Dijk, Gentry, Halevi, Vaikuntanathan 10]

Security parameter λ

$$\log X = \lambda \quad \log p = \beta \log N = \lambda^2 \quad \log N = \lambda^5$$

$$\beta^2 \log N = \frac{1}{\lambda}$$

Can be solved with:

$m > 3$ samples a_i

LLL approximation factor $2^{\dim} L^{2/3}$.

Application to fully homomorphic encryption parameters

[Coron, Mandal, Maccache, Tibouchi 11]

Assuming LLL approximation factor of $1.02^{\dim L}$,

can solve approximate GCD problem for key sizes with lattices:

key size	$\max L$	m	$\dim L$
toy	1.6×10^5	3	165
small	8.6×10^5	3	680
medium	4.2×10^6	5	3003
large	1.9×10^7	7	11440

(Just need to run LLL, not find shortest vector.)

Analogy between \mathbb{Z} and $\mathbb{F}[z]$.

Well-known analogy between integers and polynomials.

ring of integers	ring of polynomials (with coeffs in a field)
primes	irreducible polynomials
absolute value	degree of polynomial

Things work the way you want them to:

division, unique factorization, GCDs, Chinese remaindering...

lattice over \mathbb{Z}	$\mathbb{F}[z]$-module
---	--

The theorem we just proved is over the integers.

Let's translate the theorem to polynomials!

Polynomial approximate common divisors

Theorem (for integers)

Given

a_1, \dots, a_m in \mathbb{Z}

N an integer,

$1/\sqrt{\log N} \ll \beta < 1$

can find all r_i

such that

$$\gcd(a_1 - r_1, \dots, a_m - r_m, N) \geq N^\beta$$

$$|r_i| \leq N^{\beta(m+1)/m}$$

Theorem (for polynomials)

Given

$f_1(z), \dots, f_m(z)$ in $\mathbb{F}[z]$,

$N(z)$ of degree n ,

$0 \leq \beta \leq 1$

can find all $r_i(z)$

such that

$$\deg \gcd(f_1 - r_1, \dots, f_m - r_m, N) \geq n\beta$$

$$\deg r_i \leq n\beta^{(m+1)/m}$$

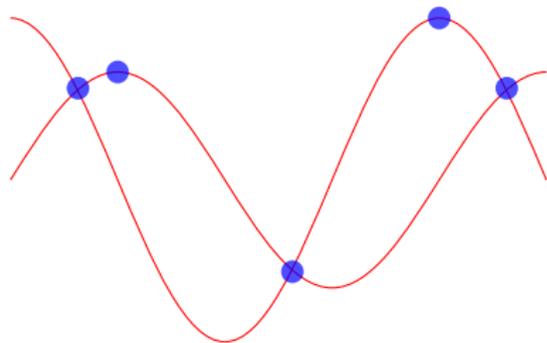
Reed-Solomon codes

Input: $\{(x_1, y_1), \dots, (x_n, y_n)\}$

Problem: Find all polynomials g of degree less than ℓ such that

$$g(x_i) = y_i$$

for at least βn pairs.



Theorem (Guruswami Sudan)

There is an efficient algorithm to do so for $\ell < \beta^2 n$.

Parvaresh-Vardy codes

Input: $(x_1, y_{11}), \dots, (x_n, y_{1n})$

\vdots

$(x_1, y_{m1}), \dots, (x_n, y_{mn})$

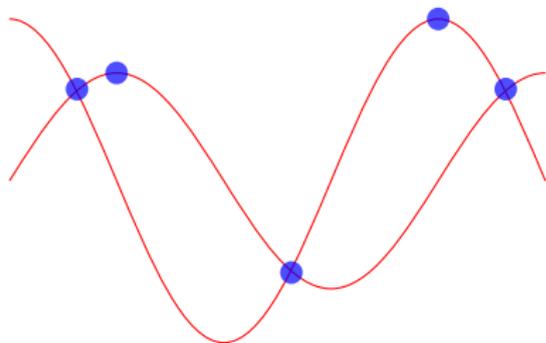
Problem: Find all polynomials g_1, \dots, g_m of degree less than ℓ such that

$$g_j(x_i) = y_{ji}$$

for at least βn values of x_i .

Theorem (Parvaresh Vardy)

There is an efficient algorithm to do so for $\ell < \beta^{(m+1)/m} n$ when g_j have a certain form.



Proof idea for univariate polynomial

1. Create a new polynomial $Q(x)$ (with coeffs in $\mathbb{F}[z]$) so that all desired $g(z)$ are roots of $Q(x)$.

- a. Look for Q in ideal $\langle x - f(z), N(z) \rangle^k$.

- b. Find $Q(x) = \sum_i q_i(z)x^i$ with small coefficients, so that

$$\deg Q(g(z)) \leq \max_i \deg q_i(z) + il < \beta k \deg N$$

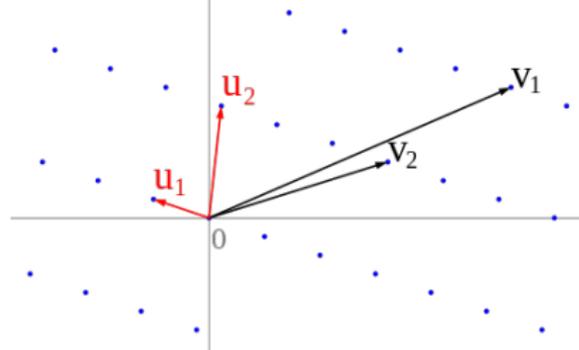
and thus $g(z)$ is a root of Q .

- c. How to find Q with low degree coefficients?

Lattice basis reduction.

Polynomial “lattices”

$\mathbb{F}[z]$ -module



$$v = (v_1(z), v_2(z), \dots, v_n(z))$$

$$|v| = \max_i \deg v_i(z)$$

Fact (von zur Gathen, Mulders and Storjohann) :

Can find exact shortest vector in polynomial time.

(True for any non-Archimedean valuation.)

Open problems

1. Remove $\beta \gg 1/\sqrt{\log N}$ restriction for integers.
2. Remove algebraic independence assumption.
3. Fully homomorphic encryption over ideal lattices: Extend to number fields without going via LLL on canonical embedding.

Approximate common divisors via lattices

<http://eprint.iacr.org/2011/437>

Implicit Factorization and Related Problems: Lattice Based Analysis

Santanu Sarkar

Kolkata, India

12 January, 2012



Outline of the Talk

Lattice based Root Finding of Polynomials

Implicit Factorization

Approximate Integer Common Divisor Problem (AICDP)

Conclusion

Finding roots of a polynomial

UNIVARIATE INTEGER POLYNOMIAL

- ▶ $f(x) \in \mathbb{Z}[x]$ with root $x_0 \in \mathbb{Z}$ efficient methods available

MULTIVARIATE INTEGER POLYNOMIAL

- ▶ $f(x, y) \in \mathbb{Z}[x, y]$ with root $(x_0, y_0) \in \mathbb{Z} \times \mathbb{Z}$ not efficient

UNIVARIATE MODULAR POLYNOMIAL

- ▶ $f(x) \in \mathbb{Z}_N[x]$ with root $x_0 \in \mathbb{Z}_N$ not efficient

Finding roots of a polynomial

UNIVARIATE INTEGER POLYNOMIAL

- ▶ $f(x) \in \mathbb{Z}[x]$ with root $x_0 \in \mathbb{Z}$ efficient methods available

MULTIVARIATE INTEGER POLYNOMIAL

- ▶ $f(x, y) \in \mathbb{Z}[x, y]$ with root $(x_0, y_0) \in \mathbb{Z} \times \mathbb{Z}$ not efficient

UNIVARIATE MODULAR POLYNOMIAL

- ▶ $f(x) \in \mathbb{Z}_N[x]$ with root $x_0 \in \mathbb{Z}_N$ not efficient

Lattice based techniques help in some cases.

Lattice

Definition (Lattice)

Let $\mathbf{v}_1, \dots, \mathbf{v}_n \in \mathbb{Z}^m$ ($m \geq n$) be n linearly independent vectors. A lattice L spanned by $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ is the set of all integer linear combinations of $\mathbf{v}_1, \dots, \mathbf{v}_n$. That is,

$$L = \left\{ \mathbf{v} \in \mathbb{Z}^m \mid \mathbf{v} = \sum_{i=1}^n a_i \mathbf{v}_i \text{ with } a_i \in \mathbb{Z} \right\}.$$

The determinant of L is defined as $\det(L) = \prod_{i=1}^n \|\mathbf{v}_i^*\|$.

Example

Consider two vectors $\mathbf{v}_1 = (1, 2)$, $\mathbf{v}_2 = (3, 4)$. The lattice L generated by $\mathbf{v}_1, \mathbf{v}_2$ is

$$L = \{ \mathbf{v} \in \mathbb{Z}^2 \mid \mathbf{v} = a_1 \mathbf{v}_1 + a_2 \mathbf{v}_2 \text{ with } a_1, a_2 \in \mathbb{Z} \}.$$

LLL Algorithm

Devised by A. Lenstra, H. Lenstra and L. Lovász in 1982

Main goal: Reduce a lattice basis in a certain way to produce a 'short (bounded)' and 'nearly orthogonal' basis called the *LLL-reduced* basis.

Lemma

Let L be an integer lattice of dimension n generated by the basis vectors $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$. Then the LLL algorithm applied on L outputs a reduced basis of L spanned by $\{\mathbf{r}_1, \dots, \mathbf{r}_n\}$ with

$$\|\mathbf{r}_1\| \leq \|\mathbf{r}_2\| \leq \dots \leq \|\mathbf{r}_i\| \leq 2^{\frac{n(n-1)}{4(n+1-i)}} \det(L)^{\frac{1}{n+1-i}}, \quad \text{for } i = 1, \dots, n$$

in time polynomial in the lattice dimension n and the bitsize of the entries of the matrix M corresponding to L .

Connecting LLL to Root finding

The clue was provided by Nick Howgrave-Graham in 1997.

Theorem

Let $h(x) \in \mathbb{Z}[x]$ be an integer polynomial with n monomials. Further, let m be a positive integer. Then, $h(x_0) = 0$ holds over integers if the following two conditions are satisfied.

$$h(x_0) \equiv 0 \pmod{N^m} \text{ with } |x_0| < X \quad \text{and} \quad \|h(xX)\| < \frac{N^m}{\sqrt{n}}$$

MAIN IDEA:

We can transform a modular polynomial $h(x)$ to an integer polynomial while preserving the root x_0 , subject to certain size constraints.

Connecting LLL to Root finding

METHODOLOGY

Given a modular polynomial to solve ...

- ▶ Construct a lattice using the given modular polynomial
- ▶ Reduce the lattice using LLL to satisfy the size constraints
- ▶ Find the appropriate integer polynomial with the same root(s)
- ▶ Find the root(s) using available techniques for integer polynomials

THE HARD PART

- ▶ Constructing the appropriate lattice structure for a polynomial

Example with Modular Polynomial

PROBLEM: Find a root x_0 of $f(x) = x^2 + ax + b \equiv 0 \pmod{N}$.

- ▶ Consider two more polynomials $g(x) = xN$ and $h(x) = N$
- ▶ Construct lattice from coefficient vectors of $f(xX)$, $g(xX)$, $h(xX)$

$$L = \begin{bmatrix} X^2 & aX & b \\ 0 & NX & 0 \\ 0 & 0 & N \end{bmatrix}$$

- ▶ Use LLL algorithm to reduce this lattice
- ▶ If the root is bounded by $|x_0| < N^{\frac{1}{3}}$, the reduction works!
- ▶ Can be improved up to $|x_0| < N^{\frac{1}{2}}$, for higher lattice dimension.

Explicit factorization

RIVEST AND SHAMIR (Eurocrypt 1985)

N can be factored given 2/3 of the LSBs of a prime

1001010100 $\overbrace{10100100101010010011}$

COPPERSMITH (Eurocrypt 1996)

N can be factored given 1/2 of the MSBs of a prime

$\overbrace{100101010010100}$ 100101010010011

BONEH ET AL. (Asiacrypt 1998)

N can be factored given 1/2 of the LSBs of a prime

100101010010100 $\overbrace{100101010010011}$

HERRMANN AND MAY (Asiacrypt 2008)

N can be factored given a random subset of the bits
(small contiguous blocks) in one of the primes

100 $\overbrace{1010100}$ 10100 $\overbrace{1001010100}$ 10011

IMPLICIT FACTORIZATION

Implicit Factorization

In PKC 2009, May and Ritzenhofen introduced Implicit Factorization

SCENARIO:

- ▶ Consider two integers N_1, N_2 such that $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$ where p_1, q_1, p_2, q_2 are primes.
- ▶ Suppose we know that p_1, p_2 share a few bits from LSB side, but we do not know the shared bits.

QUESTION:

How many bits do p_1, p_2 need to share for efficiently factoring N_1, N_2 ?

APPROXIMATE INTEGER COMMON DIVISOR PROBLEM

AICDP

Approximate Integer Common Divisor Problem (AICDP) was introduced by Howgrave-Graham in Calc 2001

THE MAIN IDEA:

- ▶ Given two large integers a, b , one can calculate $\gcd(a, b)$ efficiently.
- ▶ Is it possible to calculate $\gcd(a, b)$ efficiently when only some approximations of a, b are available?

Extensions and generalizations of AICDP

Extended Partially Approximate Common Divisor Problem (EPACDP)

Definition (EPACDP)

Let a_1, a_2, \dots, a_k be large integers (of same bitsize) and $g = \gcd(a_1, a_2, \dots, a_k)$, for $k \geq 2$. Consider that $\tilde{a}_2, \dots, \tilde{a}_k$ are the approximations of a_2, \dots, a_k respectively, and $\tilde{a}_2, \dots, \tilde{a}_k$ are of same bitsize too. Suppose that $a_2 = \tilde{a}_2 + \tilde{x}_2, \dots, a_k = \tilde{a}_k + \tilde{x}_k$.

The goal is to find $\tilde{x}_2, \dots, \tilde{x}_k$ from the knowledge of $a_1, \tilde{a}_2, \dots, \tilde{a}_k$.

Solving EPACDP

We construct the polynomials as follows:

$$\begin{aligned}h_2(x_2, \dots, x_k) &= \tilde{a}_2 + x_2, \\ &\vdots \\ h_k(x_2, \dots, x_k) &= \tilde{a}_k + x_k,\end{aligned}$$

- ▶ Note that g divides $h_i(\tilde{x}_2, \dots, \tilde{x}_k)$ for $2 \leq i \leq k$.
- ▶ We construct lattice to solve the problem using LLL reduction

Polynomials

$$H_{\underbrace{j_2, \dots, j_k}_{(k-1) \text{ many}}}(x_2, \dots, x_k) = h_2^{j_2} \cdots h_k^{j_k} a_1^{m-j_2-\cdots-j_k}$$

for non-negative integers j_2, \dots, j_k , such that $j_2 + \cdots + j_k \leq m$
and

$$H'_{i_2, 0, \dots, 0, j_2, \dots, j_k}(x_2, \dots, x_k) = x_2^{i_2} h_2^{j_2} \cdots h_k^{j_k},$$

with the following:

1. $1 \leq i_2 \leq t$, for a positive integer t , and
2. $j_2 + \cdots + j_k = m$, for non-negative integers j_2, \dots, j_k .

Theorem

Consider EPACDP with $g \approx a^{1-\alpha}$ and $\tilde{x}_2 \approx \dots \approx \tilde{x}_k \approx a^{\alpha+\beta}$.

Then one can solve EPACDP in $\text{poly}\{\log a, \exp(k)\}$ time when

$$\beta < \begin{cases} \frac{k^2(1-2\alpha)+k(5\alpha-2)-2\alpha+1-\sqrt{k^2(1-\alpha^2)+2k(\alpha^2-1)+1}}{k^2-3k+2}, & \text{for } k > 2 \\ 1 - 3\alpha + \alpha^2, & \text{for } k = 2 \end{cases}$$

with the constraint $2\alpha + \beta \leq 1$.

Improved Results for Larger k

$$\begin{aligned}a_1 &= gq_1, \\ \tilde{a}_2 &= gq_2 - \tilde{x}_2, \\ &\vdots \\ \tilde{a}_k &= gq_k - \tilde{x}_k.\end{aligned}$$

Let us construct the matrix

$$M = \begin{pmatrix} 2^{\rho} & \tilde{a}_2 & \tilde{a}_3 & \dots & \tilde{a}_k \\ 0 & -a_1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -a_1 \end{pmatrix}$$

where $2^{\rho} \approx \tilde{x}_2$.

Improved Results for Larger k

Note that $(q_1, q_2, \dots, q_k) \cdot M = (2^\rho q_1, -q_1 \tilde{x}_2, \dots, q_1 \tilde{x}_k) = \mathbf{b}$, say.

$$\|\mathbf{b}\| < \sqrt{k} \cdot a^{2\alpha+\beta}.$$

$$|\det(M)| = 2^\rho a_1^{k-1} \approx a^{\alpha+\beta+k-1}.$$

$$a^{2\alpha+\beta} < a^{\frac{\alpha+\beta+k-1}{k}} \Leftrightarrow \beta < \frac{k-1+\alpha-2\alpha k}{k-1}.$$

Implicit Factorization

EXISTING RESULT:

Let $q_1, q_2 \approx N^\alpha$. Then number of unshared bits $\beta \log_2 N$ will be at most $(1 - 3\alpha) \log_2 N$ for polynomial time factorization. [May et al]

OUR RESULTS:

- ▶ We have considered the case where some amount of LSBs as well as MSBs of p_1, p_2 are shared simultaneously
- ▶ We improved the existing result in some cases

IMPLICIT FACTORIZATION PROBLEM

relates to

APPROXIMATE INTEGER
COMMON DIVISOR PROBLEM

ACDP and Implicit Factorization Problem

OUR RESULT: We relate the approximate common divisor problem to the implicit factorization problem

THE RELATION:

- ▶ Suppose p_1, p_2 share certain amount of MSBs.
- ▶ One can write $p_1 - p_2 = x_0$, and hence
$$N_2 = p_2 q_2 = (p_1 - x_0) q_2$$
- ▶ Therefore, we have
$$\gcd(N_1, N_2 + x_0 q_2) = \gcd(p_1 q_1, p_1 q_2) = p_1.$$
- ▶ So, solving ACDP here will solve the Implicit Factorization

Using some tweaks, this approach works for case with LSB bits shared.

Our result for bits shared in MSBs and LSBs

Theorem

Let $q_1, q_2, \dots, q_k \approx N^\alpha$, and consider that $\gamma_1 \log_2 N$ many MSBs and $\gamma_2 \log_2 N$ many LSBs of p_1, \dots, p_k are the same. Also define $\beta = 1 - \alpha - \gamma_1 - \gamma_2$.

Then, one can factor N_1, N_2, \dots, N_k in $\text{poly}\{\log N, \exp(k)\}$ if

$$\beta < \begin{cases} C(\alpha, k), & \text{for } k > 2, \\ 1 - 3\alpha + \alpha^2, & \text{for } k = 2, \end{cases}$$

with the constraint $2\alpha + \beta \leq 1$, where

$$C(\alpha, k) = \frac{k^2(1 - 2\alpha) + k(5\alpha - 2) - 2\alpha + 1 - \sqrt{k^2(1 - \alpha^2) + 2k(\alpha^2 - 1) + 1}}{k^2 - 3k + 2}.$$

Comparison with the existing works

k	Bitsize of p_i, q_i $(1 - \alpha) \log_2 N, \alpha \log_2 N$	No. of shared LSBs May et al in p_i				No. of shared LSBs (our) in p_i			
		Theory	Expt.	LD	Time	Theory	Expt.	LD	Time
3	750, 250	375	378	3	< 1	352	367	56	41.92
* 3	700, 300	450	452	3	< 1	416	431	56	59.58
* 3	650, 350	525	527	3	< 1	478	499	56	74.54
# 3	600, 400	600	-	-	-	539	562	56	106.87
* 4	750, 250	334	336	4	< 1	320	334	65	32.87
* 4	700, 300	400	402	4	< 1	380	400	65	38.17
* 4	650, 350	467	469	4	< 1	439	471	65	39.18
* 4	600, 400	534	535	4	< 1	497	528	65	65.15

Table: For 1000 bit N , theoretical and experimental data of the number of shared LSBs in May et al and shared LSBs in our case. (Time in seconds)

Generalized EPACDP results

Theorem

Consider EPACDP with $g \approx a^{1-\alpha}$ and $\tilde{x}_2 \approx \dots \approx \tilde{x}_k \approx a^{\alpha+\beta}$.

Then one can solve EPACDP in $\text{poly}\{\log a, k\}$ time when,

$$\beta < 1 - \frac{2k-1}{k-1}\alpha.$$

This holds for large values of k , and improves the existing results.

Comparison with the existing works

k	Bitsize of p_i, q_i	No. of shared MSBs of Faugère et al in p_i				No. of shared MSBs (our) in p_i			
		Theory	Expt.	LD	Time (sec)	Theory	Expt.	LD	Time (sec)
10	874, 150	171	170	10	< 1	166	170	10	< 1
10	824, 200	227	225	10	< 1	220	225	10	< 1
10	774, 250	282	280	10	< 1	274	280	10	< 1
10	724, 300	338	334	10	< 1	328	332	10	< 1
10	674, 350	393	390	10	< 1	382	388	10	< 1
10	624, 400	449	446	10	< 1	435	444	10	< 1
40	874, 150	158	157	40	12.74	154	157	40	< 1
40	824, 200	209	206	40	17.42	205	206	40	< 1
40	774, 250	261	258	40	21.64	256	258	40	1.13
40	724, 300	312	309	40	24.17	307	308	40	1.26
40	674, 350	363	361	40	29.87	358	360	40	1.48
40	624, 400	414	412	40	34.69	409	410	40	1.75
100	874, 150	155	154	100	299.64	152	153	100	5.63
100	824, 200	206	205	100	525.67	202	204	100	9.36
100	774, 250	257	257	100	781.42	253	255	100	14.11
100	724, 300	307	307	100	1053.66	303	305	100	18.61
100	674, 350	358	357	100	1415.02	353	355	100	24.16
100	624, 400	408	408	100	2967.75	404	406	100	29.95

Table: For 1024-bit N , theoretical (bound for Faugère et al and in our case) and experimental data of the number of shared MSBs in Faugère et al and shared MSBs in our case.

Finding $q^{-1} \bmod p \equiv$ Factorization of N

Crypto 2009 Presentation: How can we use $q^{-1} \bmod p$?

Brief sketch of our approach:

- ▶ Let us denote $q_1 = q^{-1} \bmod p$.
- ▶ That is $qq_1 = 1 + k_1p$.
- ▶ Hence $q_1N = p + k_1p^2$
- ▶ Now $\gcd(q_1N - p, N^2) = p^2$
- ▶ From knowledge of q_1N and N^2 , one can find p^2 when $p > q$.
- ▶ When p, q are of same bit size, still we can find p^2 .

Finding $q^{-1} \bmod p \equiv$ Factorization of N

Theorem

Assume $N = pq$, where p, q are primes and $p \approx N^\gamma$. Suppose an approximation p_0 of p is known such that $|p - p_0| < N^\beta$. Given $q^{-1} \bmod p$, one can factor N deterministically in $\text{poly}(\log N)$ time when $\beta - 2\gamma^2 < 0$.

Lattice Construction

LET $x_0 = p - p_0 \Rightarrow \gcd(q_1 N - p_0 - x_0, N^2) = p^2$

Take $X = N^\beta$ as an upper bound of x_0 .

Polynomials

$$g_i(x) = (q_1 N - p_0 + x)^i N^{2(m-i)} \text{ for } 0 \leq i \leq m,$$

$$g'_i(x) = x^i (q_1 N - p_0 + x)^m \text{ for } 1 \leq i \leq t$$

$$g_i(-x_0) \equiv g'_i(-x_0) \equiv 0 \pmod{p^{2m}}.$$

Construct the lattice L spanned by the coefficient vectors of the polynomials $g_i(xX), g'_i(xX)$

Lattice Construction

Dimension of the lattice: $\omega = m + t + 1$

$$\det(L) = X^{\frac{(m+t)(m+t+1)}{2}} N^{2\frac{m(m+1)}{2}} = X^{\frac{(m+t)(m+t+1)}{2}} N^{m(m+1)}.$$

For finding the integer root, we need

$$2^{\frac{\omega-1}{4}} (\det(L))^{\frac{1}{\omega}} < \frac{p^{2m}}{\sqrt{\omega}}.$$

Finding smooth integers

Theorem

Let $S = \prod_{i=1}^n p_i^{a_i}$ where $a_i = \lfloor \frac{\log B}{\log p_i} \rfloor$ and p_1, \dots, p_n are all distinct primes not exceeding B . Let $I = [U, V]$ be an interval.

One can find all strongly B smooth integers $N \in I$ for which $\gcd(N, S) > d$ in $\text{poly}(\log S)$ time when $|I| < 2d^{\frac{\log d}{\log S}}$ and $V < 2d$.

EXISTING RESULT: $|I| < \frac{1}{4} d^{\frac{\log d}{\log S}}$.

[Boneh, STOC 2000]

B	$\log_2 d$	$\log_2(V - U)$	Our Time (sec.)	Boneh Time (sec.)
1000	450	130	15.51	21.33
1000	496	156	3.77	8.06
1000	496	161	36.88	64.71

Comparison with existing work

Recent papers:

- ▶ Approximate Integer Common Divisor Problem Relates to Implicit Factorization. S. Sarkar and S. Maitra.
 - ▶ IACR Cryptology ePrint Archive, 18 December 2009
 - ▶ IEEE Transactions on Information Theory, accepted December 15, 2010, published June, 2011
- ▶ Approximate common divisors via lattices. H. Cohn and N. Heninger. CoRR, 12 August, 2011.

Comparison:

- ▶ Both the results have same bounds for $k = 2, 3$.
- ▶ Cohn-Heninger's result is better than ours for $k > 3$.

Conclusion

SUMMARY AND FUTURE DIRECTION

Summary of the talk

In this talk, we have

- ▶ studied Lattice based techniques for finding root(s) of polynomials
- ▶ established a relation between Approximate GCD problem and Implicit Factorization
- ▶ and proposed two new applications of Approximate GCD problem

Future direction of research

OPEN PROBLEMS:

- ▶ Can one use some known random bits of $q^{-1} \bmod p$ to factor N ?
- ▶ Does the knowledge of random bits of $q^{-1} \bmod p$ reduce the required number of bits to be known for other private keys in case of factoring N ?
- ▶ Can one factor k balanced RSA moduli N_1, \dots, N_k in polynomial time when p_1, \dots, p_k share their MSBs?

Another interesting Lattice-based problem

Cryptanalysis of Multi-Prime Φ -Hiding Problem

- ▶ Applied by Kiltz, O'Neill and Smith in Crypto 2010.
- ▶ Base setup is multi-prime RSA with $N = p_1 \cdots p_m$
- ▶ N $m\Phi$ hides a prime e if $e|p_i - 1$ for $1 \leq i \leq m - 1$
- ▶ Distinguish primes which are $m\Phi$ hidden by N from those which that do not divide $p_i - 1$ for any i
- ▶ The main problem is to solve the system
$$ex_1 + 1 = p_1, \dots, ex_{m-1} + 1 = p_{m-1}.$$

Table: Our results for 2048 bit N and for 80 bit security.

Value of m	Lossiness in the work of Kiltz et al.		
	Before the work of Hermann	After the work of Hermann	After our work
4	806	778	768
5	872	822	778

Thank You!

