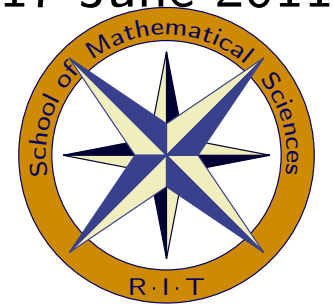
Integer-Based Fully Homomorphic Encryption

Michael Snook

$\mathbb{R} \cdot \mathbb{I} \cdot \mathbb{T}$

17 June 2011



RSA Encryption

If m is an integer (mod N) to encrypt, RSA encrypts it as:

$$\mathcal{E}(m) = m^e \bmod N$$

RSA Encryption

If m is an integer (mod N) to encrypt, RSA encrypts it as:

$$\begin{aligned}
\mathcal{E}(m) &= m^e \bmod N \\
\mathcal{E}(m_1 m_2) &= (m_1 m_2)^e \bmod N \\
&= (m_1^e m_2^e) \bmod N \\
&= (m_1^e \bmod N)(m_2^e \bmod N) \\
&= \mathcal{E}(m_1)\mathcal{E}(m_2)
\end{aligned}$$

Addition in RSA

Let $N = 91, e = 7$. Compare $\mathcal{E}(1) + \mathcal{E}(2)$ to $\mathcal{E}(1+2)$:

$$\begin{aligned}
\mathcal{E}(1) + \mathcal{E}(2) &= (1^7 \bmod 91) + (2^7 \bmod 91) \\
&= (1 + 128) \bmod 91 \\
&= 38 \\
E(1+2) &= 3^7 \bmod 91 \\
&= 2187 \bmod 91 \\
&= 3
\end{aligned}$$

Homomorphic Encryption

Definition (Homomorphic Encryption)

Homomorphic encryption is a method of encryption that allows specific operations to be performed on ciphertexts, so that the result is a valid encryption of a related operation performed on the underlying plaintexts.

Fully Homomorphic Encryption

Definition (Fully Homomorphic Encryption)

A fully homomorphic cryptosystem is a homomorphic cryptosystem such that any computable function on plaintexts may be evaluated homomorphically by applying operations to the corresponding ciphertexts.

Common Properties

- Most Homomorphic systems work on either $\mathbb{Z}$ or $\mathbb{Z}_n$
- These systems usually preserve either addition or multiplication
- Some systems convert addition to multiplication or the reverse

Common Properties

- Most Homomorphic systems work on either $\mathbb{Z}$ or $\mathbb{Z}_n$
- These systems usually preserve either addition or multiplication
- Some systems convert addition to multiplication or the reverse
- For some applications either addition or multiplication is enough
- Having the ability to perform both is much more powerful

Applications

Electronic Voting Additively homomorphic encryption allows one party to total the ballots without being able to tell what they are. The Paillier cryptosystem, in which multiplication of ciphertexts corresponds to addition of plaintexts, is a common candidate.

Cloud Computing Fully homomorphic encryption allows users to store their data securely on a remote server, allowing the server to perform any necessary computations on it without compromising the data's security or privacy.

Secure Multi-Party Communication

Combining information from different people without revealing any one person's information to the others.

Millionaire Problem Two millionaires wish to know which of them is richer without either one finding out how much money the other has.

Secret Auction A generalization of the Millionaire problem: Determine which bidder put in the highest bid without revealing the individual bids to the bidders.

Average Pay A group of employees could use additively homomorphic encryption to determine the sum of their salaries — and therefore their average salary — without revealing who makes how much.

Millionaire Problem

Suppose Alice has a dollars, and Bob has b dollars. They wish to know whether $a > b$ or $b > a$. With an impartial third party, Charlie, one solution is as follows:

1. Using Charlie's public key, Alice computes $\mathcal{E}(a)$ and Bob computes $\mathcal{E}(-b)$.

Millionaire Problem

Suppose Alice has a dollars, and Bob has b dollars. They wish to know whether $a > b$ or $b > a$. With an impartial third party, Charlie, one solution is as follows:

1. Using Charlie's public key, Alice computes $\mathcal{E}(a)$ and Bob computes $\mathcal{E}(-b)$.
2. Together, Alice and Bob compute $\mathcal{E}(a) + \mathcal{E}(-b) = \mathcal{E}(a - b)$.

Millionaire Problem

Suppose Alice has a dollars, and Bob has b dollars. They wish to know whether $a > b$ or $b > a$. With an impartial third party, Charlie, one solution is as follows:

1. Using Charlie's public key, Alice computes $\mathcal{E}(a)$ and Bob computes $\mathcal{E}(-b)$.
2. Together, Alice and Bob compute $\mathcal{E}(a) + \mathcal{E}(-b) = \mathcal{E}(a - b)$.
3. Alice and Bob tell Charlie $\mathcal{E}(a - b)$. Charlie then determines $a - b$.

Millionaire Problem

Suppose Alice has a dollars, and Bob has b dollars. They wish to know whether $a > b$ or $b > a$. With an impartial third party, Charlie, one solution is as follows:

1. Using Charlie's public key, Alice computes $\mathcal{E}(a)$ and Bob computes $\mathcal{E}(-b)$.
2. Together, Alice and Bob compute $\mathcal{E}(a) + \mathcal{E}(-b) = \mathcal{E}(a - b)$.
3. Alice and Bob tell Charlie $\mathcal{E}(a - b)$. Charlie then determines $a - b$.
4. Charlie tells Alice and Bob whether $a - b > 0$ or $a - b < 0$.

Bootstrapping

- Fully homomorphic encryption was first proposed in 1978 by Rivest, Adelman and Dertouzos, who used the term *privacy homomorphism*.

Bootstrapping

- Fully homomorphic encryption was first proposed in 1978 by Rivest, Adelman and Dertouzos, who used the term *privacy homomorphism*.
- Since then, a number of partially homomorphic systems have been devised as well as applications for homomorphic encryption.

Bootstrapping

- Fully homomorphic encryption was first proposed in 1978 by Rivest, Adelman and Dertouzos, who used the term *privacy homomorphism*.
- Since then, a number of partially homomorphic systems have been devised as well as applications for homomorphic encryption.
- In 2009, Craig Gentry presented a new technique called bootstrapping. This technique removes the bounds on complexity from certain bounded cryptosystems.

How Bootstrapping Works

- Takes systems that can evaluate both addition and multiplication but are limited in number of operations
- Removes limit on number of operations
- Works by homomorphically evaluating decryption function, effectively resetting the number of operations performed
- Requires the decryption function to be small enough to be evaluated

Overview

In 2009, a team of researchers proposed a bootstrappable cryptosystem based off of simple addition and multiplication of integers. Their system encrypts a single bit at a time; we can consider these bits to be elements of $\mathbb{Z}_2$ so they have a ring structure.
The private key is denoted p, and is a positive, odd integer η bits long.

Encryption and Decryption

Let m be a plaintext bit to encrypt (m is either 0 or 1). If p is the private key, take two random integers q and r a form the ciphertext c:

$$c = pq + 2r + m \tag{1}$$

The variables q and r have specific lengths, but no additional restrictions.
Decryption is obtained by taking

$$m = (c \bmod p) \bmod 2 \tag{2}$$

Additive Homomorphism

Given two plaintexts m_1 and m_2, set c_1 and c_2 according to equation (1), and take $c_1 + c_2$:

$$\begin{aligned} c_1 + c_2 &= pq_1 + 2r_1 + m_1 + pq_2 + 2r_2 + m_2 \\ &= p(q_1 + q_2) + 2(r_1 + r_2) + (m_1 + m_2) \end{aligned}$$

Additive Homomorphism

Given two plaintexts m_1 and m_2, set c_1 and c_2 according to equation (1), and take $c_1 + c_2$:

$$\begin{aligned} c_1 + c_2 &= pq_1 + 2r_1 + m_1 + pq_2 + 2r_2 + m_2 \\ &= p(q_1 + q_2) + 2(r_1 + r_2) + (m_1 + m_2) \end{aligned}$$

Decrypting this gives

$$\begin{aligned} m &= ((c_1 + c_2) \bmod p) \bmod 2 \\ &= ((2(r_1 + r_2) + (m_1 + m_2)) \bmod p) \bmod 2 \\ &= (2(r_1 + r_2) + (m_1 + m_2)) \bmod 2 \\ &= m_1 + m_2 \bmod 2 = m_1 \oplus m_2 \end{aligned}$$

Note that if $2(r_1 + r_2) + (m_1 + m_2) \geq p$, then the red line need not hold.

Multiplicative Homomorphism

Given two plaintexts m_1 and m_2, set c_1 and c_2 according to equation (1), and take $c_1 \cdot c_2$:

$$\begin{aligned} c_1 \cdot c_2 &= (pq_1 + 2r_1 + m_1)(pq_2 + 2r_2 + m_2) \\ &= p^2 q_1 q_2 + 2pq_1 r_2 + pq_1 m_2 + 2pq_2 r_1 + 4r_1 r_2 \\ &\quad + 2r_1 m_2 + pq_2 m_1 + 2r_2 m_1 + m_1 m_2 \end{aligned}$$

Multiplicative Homomorphism

Given two plaintexts m_1 and m_2, set c_1 and c_2 according to equation (1), and take $c_1 \cdot c_2$:

$$\begin{aligned} c_1 \cdot c_2 &= (pq_1 + 2r_1 + m_1)(pq_2 + 2r_2 + m_2) \\ &= p^2 q_1 q_2 + 2pq_1 r_2 + pq_1 m_2 + 2pq_2 r_1 + 4r_1 r_2 \\ &\quad + 2r_1 m_2 + pq_2 m_1 + 2r_2 m_1 + m_1 m_2 \end{aligned}$$

Decrypting this gives

$$\begin{aligned} m &= ((c_1 \cdot c_2) \bmod p) \bmod 2 \\ &= (4r_1 r_2 + 2r_1 m_2 + 2r_2 m_1 + m_1 m_2) \bmod 2 \\ &= m_1 m_2 \bmod 2 = m_1 \wedge m_2 \end{aligned}$$

Similar to the additive case, if $(2r_1 + m_1)(2r_2 + m_2) \geq p$ then this may fail.

Homomorphic Enough

As seen above, the encryption preserves both addition and multiplication as long as the noise term r is small enough in both ciphertexts; repeated operations increase the size of this noise, limiting the size of polynomials that the system can correctly evaluate. Adding bootstrapping removes this restriction.

Homomorphic Enough

As seen above, the encryption preserves both addition and multiplication as long as the noise term r is small enough in both ciphertexts; repeated operations increase the size of this noise, limiting the size of polynomials that the system can correctly evaluate. Adding bootstrapping removes this restriction.

If this system is bootstrapped, then we can evaluate any function comprised of addition and multiplication in $\mathbb{Z}_2$. Addition modulo 2 corresponds to Boolean XOR and multiplication corresponds to Boolean AND; together with the constant 1, these can form *any* Boolean function. Thus, the cryptosystem can certainly be considered fully homomorphic.

Public Key Version

- $x_i = \mathcal{E}_i(0) = q_i p + 2r_i$

Public Key Version

- $x_i = \mathcal{E}_i(0) = q_i p + 2r_i$
- $\sum x_i = (\sum q_i)p + 2(\sum r_i) = \mathcal{E}^*(0)$

Public Key Version

- $x_i = \mathcal{E}_i(0) = q_i p + 2r_i$
- $\sum x_i = (\sum q_i)p + 2(\sum r_i) = \mathcal{E}^*(0)$
- $\mathcal{E}^*(0) + 2r + m = \mathcal{E}(m)$

Public Key Version

- $x_i = \mathcal{E}_i(0) = q_i p + 2r_i$
- $\sum x_i = (\sum q_i)p + 2(\sum r_i) = \mathcal{E}^*(0)$
- $\mathcal{E}^*(0) + 2r + m = \mathcal{E}(m)$
- Public key is the set $\{x_i\}_{i=0}^{\tau}$

Subset Sum Problem

- Given a set S and a sum N

Subset Sum Problem

- Given a set S and a sum N
- Is there some subset $A \subset S$ such that

$$\sum_{x \in A} x = N$$

Subset Sum Problem

- Given a set S and a sum N
- Is there some subset $A \subset S$ such that

$$\sum_{x \in A} x = N$$

- General case is hard to solve exactly

Subset Sum Problem

- Given a set S and a sum N
- Is there some subset $A \subset S$ such that

$$\sum_{x \in A} x = N$$

- General case is hard to solve exactly
- Specific cases and approximate solutions can be easier to solve

Subset Sum Problem

- Given a set S and a sum N
- Is there some subset $A \subset S$ such that

$$\sum_{x \in A} x = N$$

- General case is hard to solve exactly
- Specific cases and approximate solutions can be easier to solve

Example

Let S be the set $\{3, 7, 11, 23, 34, 64, 65, 69, 84\}$. Is there some subset of S whose sum is 123? One whose sum is 186?

Subset Sum Attack

- Given a ciphertext c and a public key $K = \{x_i\}_{i=0}^{\tau}$

Subset Sum Attack

- Given a ciphertext c and a public key $K = \{x_i\}_{i=0}^{\tau}$
- Does some subset of K sum to c?

Subset Sum Attack

- Given a ciphertext c and a public key $K = \{x_i\}_{i=0}^{\tau}$
- Does some subset of K sum to c?
- If so, $c = \mathcal{E}(0)$.

Subset Sum Attack

- Given a ciphertext c and a public key $K = \{x_i\}_{i=0}^{\tau}$
- Does some subset of K sum to c?
- If so, $c = \mathcal{E}(0)$.
- Does some subset of K sum to $c - 1$?

Subset Sum Attack

- Given a ciphertext c and a public key $K = \{x_i\}_{i=0}^{\tau}$
- Does some subset of K sum to c?
- If so, $c = \mathcal{E}(0)$.
- Does some subset of K sum to $c - 1$?
- If so, $c = \mathcal{E}(0) + 1 = \mathcal{E}(1)$.

This is why the extra noise is added during public key encryption: to make this problem harder for an attacker.

Approximate GCD Problem

- Given a set S of integers

Approximate GCD Problem

- Given a set S of integers
- $\forall s \in S, s \approx k_s p$

Approximate GCD Problem

- Given a set S of integers
- $\forall s \in S, s \approx k_s p$
- Where the error in approximation is bounded

Approximate GCD Problem

- Given a set S of integers
- $\forall s \in S, s \approx k_s p$
- Where the error in approximation is bounded
- and the length of p (in bits) is known

Approximate GCD Problem

- Given a set S of integers
- $\forall s \in S, s \approx k_s p$
- Where the error in approximation is bounded
- and the length of p (in bits) is known
- What is p?

Approximate GCD Example

- $S = \{37, 65, 107\}$
- p is 4 bits long
- For each $x \in S$, $k_x p < x < k_x p + 8$

Find p.

Approximate GCD Example

- $S = \{37, 65, 107\}$
- p is 4 bits long
- For each $x \in S$, $k_x p < x < k_x p + 8$

Find p.

$$37 = 2 \cdot 15 + 7$$
$$65 = 4 \cdot 15 + 5$$
$$107 = 7 \cdot 15 + 2$$

Approximate GCD Example

- $S = \{37, 65, 107\}$
- p is 4 bits long
- For each $x \in S$, $k_x p < x < k_x p + 8$

Find p.

$$37 = 2 \cdot 15 + 7$$
$$65 = 4 \cdot 15 + 5$$
$$107 = 7 \cdot 15 + 2$$

$$p = 15$$

Approximate GCD Attack

- The public key is a set $\{x_i\}_{i=0}^{\tau}$ of integers, where

Approximate GCD Attack

- The public key is a set $\{x_i\}_{i=0}^{\tau}$ of integers, where
- $x_i = q_i p + 2r_i$,

Approximate GCD Attack

- The public key is a set $\{x_i\}_{i=0}^{\tau}$ of integers, where
- $x_i = q_i p + 2r_i$,
- Each r_i is bounded by a known integer 2^{ρ} and

Approximate GCD Attack

- The public key is a set $\{x_i\}_{i=0}^{\tau}$ of integers, where
- $x_i = q_i p + 2r_i$,
- Each r_i is bounded by a known integer 2^{ρ} and
- Secret key p is known to be exactly η bits long.

Approximate GCD Attack

- The public key is a set $\{x_i\}_{i=0}^{\tau}$ of integers, where
- $x_i = q_i p + 2r_i$,
- Each r_i is bounded by a known integer 2^{ρ} and
- Secret key p is known to be exactly η bits long.
- Finding p from the public key is an instance of the approximate gcd problem.

Lattices

- Subset of $\mathbb{R}^n$

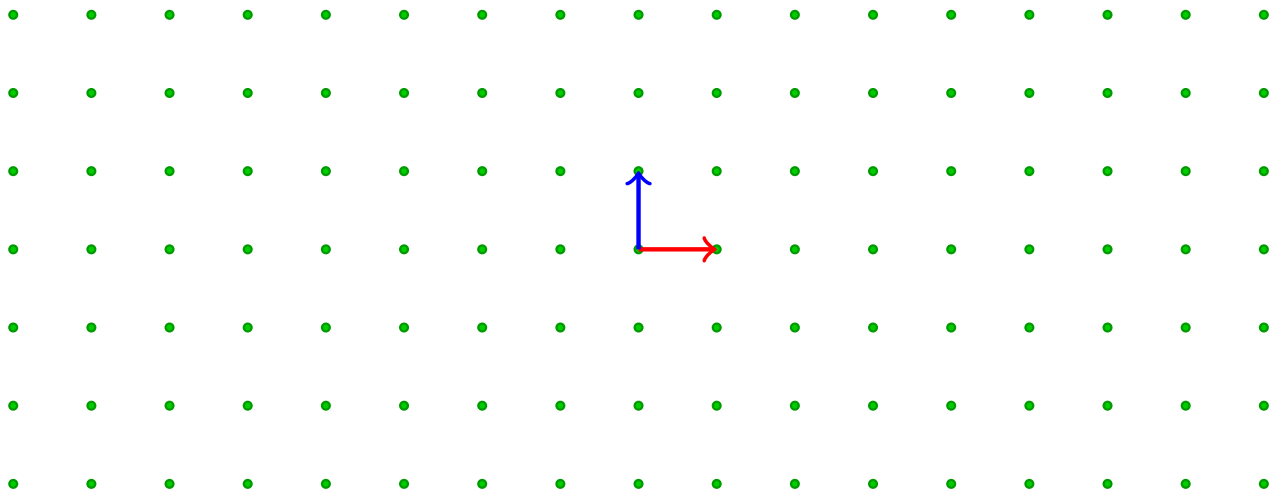


Figure: 2-dimensional Square Lattice

Lattices

- Subset of $\mathbb{R}^n$
- Closed under addition and scalar multiplication

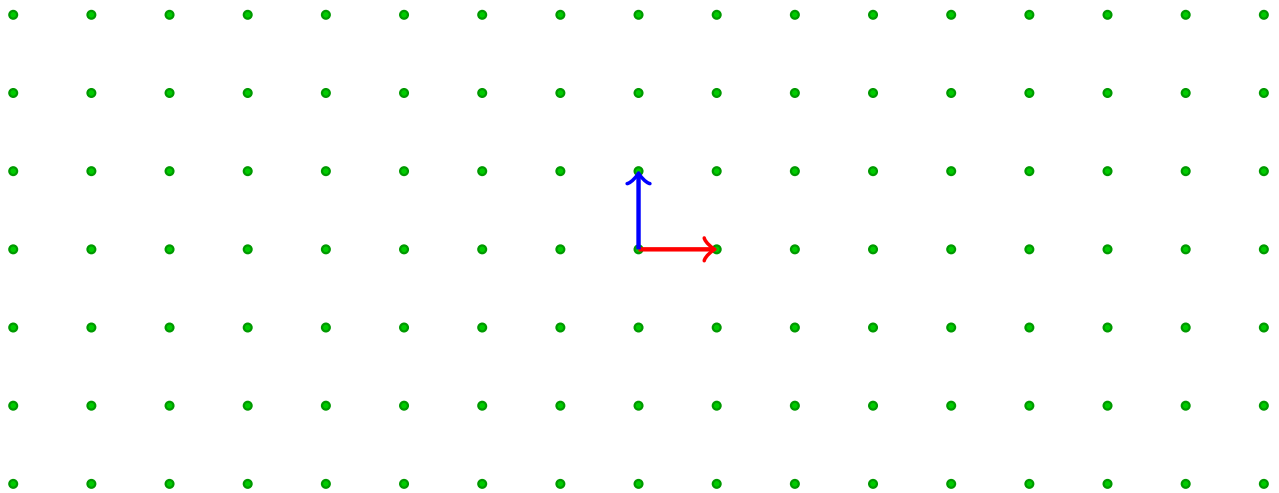


Figure: 2-dimensional Square Lattice

Lattices

- Subset of $\mathbb{R}^n$
- Closed under addition and scalar multiplication
- Scalars must be integers (unlike vector space)

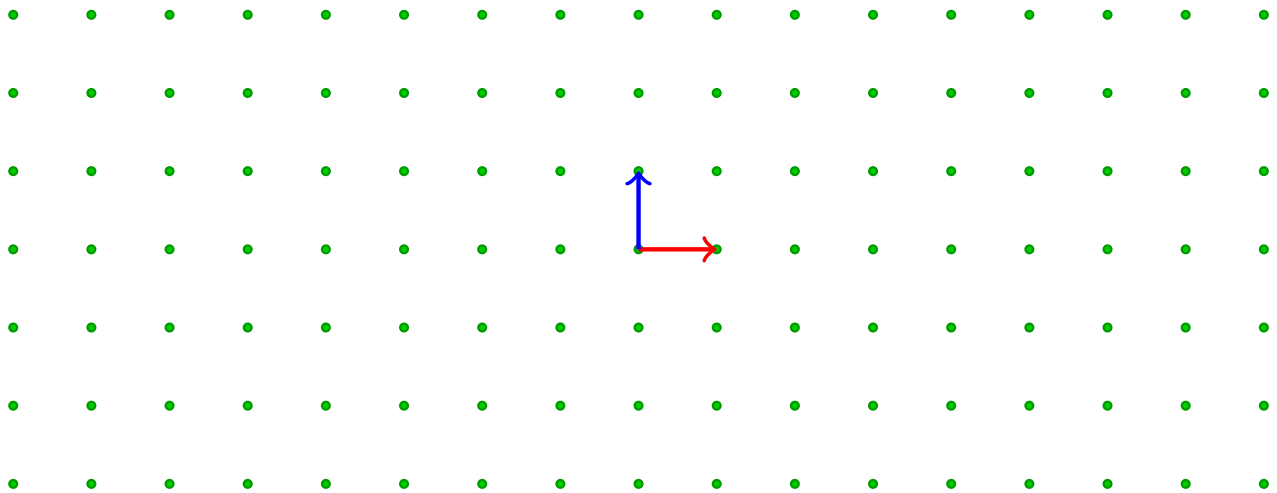


Figure: 2-dimensional Square Lattice

Lattices

- Subset of $\mathbb{R}^n$
- Closed under addition and scalar multiplication
- Scalars must be integers (unlike vector space)
- We shall consider lattices with only integer coordinates for every point.

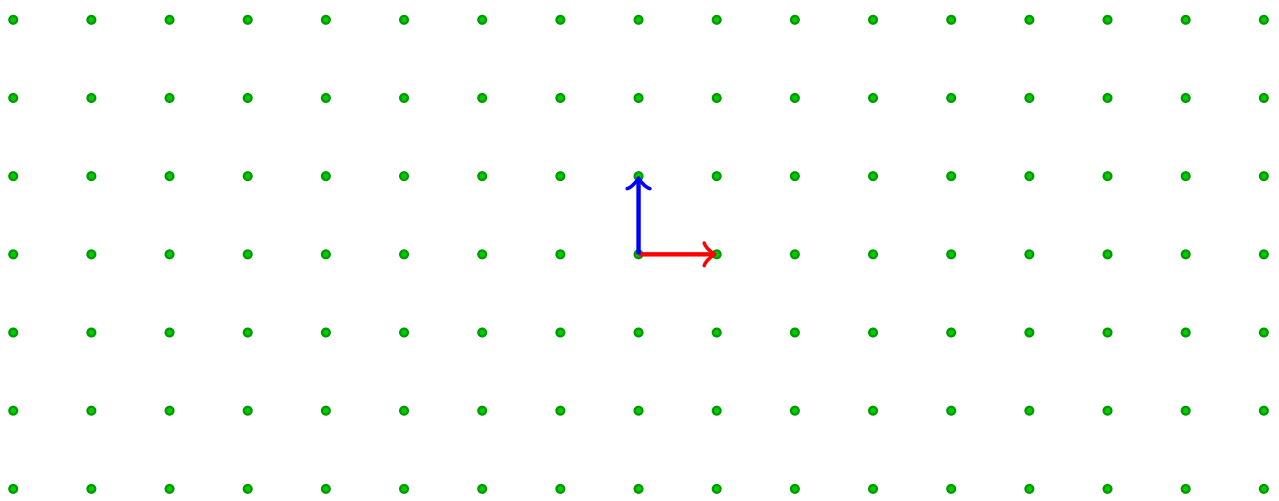


Figure: 2-dimensional Square Lattice

Shortest Vector

- Lattices, unlike real vector spaces, have non-zero vectors of minimal length.
- Given a lattice, finding one of these vectors is called the *shortest vector problem* (SVP).
- Exact solutions hard, approximate solutions easier.
- LLL (Lenstra-Lenstra-Lovász) basis reduction is one polynomial-time approximation technique.

Relation to Cryptosystem

- From a public key, we can create a lattice M

Relation to Cryptosystem

- From a public key, we can create a lattice M
- Private key p is related to short vectors in M

Relation to Cryptosystem

- From a public key, we can create a lattice M
- Private key p is related to short vectors in M
- Short vectors represent one approach to approximate gcd attack

Relation to Cryptosystem

- From a public key, we can create a lattice M
- Private key p is related to short vectors in M
- Short vectors represent one approach to approximate gcd attack
- System parameters must be chosen to make this attack difficult

System Parameters

The system depends on the following parameters:

λ	Security parameter
η	Length, in bits, of the private key p
ρ	Length of the noise r_i in each public key integer
ρ'	Length of the noise r during encryption
γ	Length of ciphertexts/integers in Public Key
τ	Number of integers in Public Key

Default Parameters

- $\rho = \lambda$
- $\rho' = 2\lambda$
- $\eta = \tilde{\mathcal{O}}(\lambda^2)$
- $\gamma = \tilde{\mathcal{O}}(\lambda^5)$
- $\tau = \gamma + \lambda$

This gives a public key size of $\mathcal{O}(\lambda^{10})$ and a message expansion ratio of $\mathcal{O}(\lambda^5)$.

Preliminary Tests—Key Generation

λ	Time Taken	Key Size on Disk
2	1s–2s	152K
4	0.2s–0.4s	4.7M
8	$\approx$6m	474M
16	$>$ 1h, did not finish	$>$ 15G

Relaxing Parameters

Initial tests take too long and produce overly large keys, without even reaching reasonable security levels. One possible solution is to reduce the values for some parameters. Since γ is the most important factor in system size, followed by τ, the options are:

- Reduce γ. Since $\tau = \gamma + \lambda$, this will reduce the value of τ as well.
- Reduce τ but not γ.
- Reduce γ, and further reduce τ beyond $\gamma + \lambda$.

Relaxed Results

We use the relaxation $\gamma = \lambda^{2.5}$ for the following results. We keep $\tau = \gamma + \lambda$, for overall system complexity $\mathcal{O}(\lambda^5)$. For the large key, τ is further reduced to keep the computation within hardware limits.

λ	γ	τ	Time Taken (s)	Public Key (MiB)
16	1024	1040	0.346	4.2
64	32768	32832	86.795	386
256	1048576	2048	707.162	625

Table: Key Generation Results

Relaxed Results Continued

λ	Plaintext Size (B)	Cyphertext Size (KiB)	Expansion Ratio	Encryption Time (s)	Decryption Time (ms)
4	29	2.8	99.25	0.057	13.5
4	40	3.8	98.35	0.072	11.6
4	238	22.3	96.35	0.346	17.0
16	29	72.8	2573.00	0.390	18.5
16	40	99.6	2550.00	0.386	24.0
16	238	580.0	2498.00	2.181	70.0
64	29	2250.0	81664.00	25.800	258.0
64	40	3080.0	80916.00	21.300	348.0
64	238	17000.0	79274.00	135.000	1800.0

Effects of Relaxation

- Reducing γ provides a noticeable effect on the system performance.
- Reducing τ provides only marginal performance gains, and only for key generation.
- Reducing γ still leaves attacking the key with reasonable security against casual lattice-based attacks.
- Size of ciphertexts and public key still too large to be considered practical.

Summary

Bootstrapping requires homomorphically evaluating the function $(c \bmod p) \bmod 2$. The key step is evaluating the modulo p. Thus, encrypting p and evaluating $c \bmod p$ homomorphically is doable with only addition and multiplication. Unfortunately, doing so requires boolean circuits that are too large by a constant factor to be evaluated correctly.

Adding a Hint

In order to shrink the decryption to a bootstrappable size, we add a hint to the public key for finding the private key. This hint takes the form of a set of rational numbers in the interval $[0, 2)$ with a subset whose sum is related to the private key.

Adding a Hint

In order to shrink the decryption to a bootstrappable size, we add a hint to the public key for finding the private key. This hint takes the form of a set of rational numbers in the interval $[0, 2)$ with a subset whose sum is related to the private key.
Unfortunately, this hint is even larger than the original public key. It also, requires ciphertexts to carry additional information in order to actually use the hint for bootstrapping, which increases the size of ciphertexts as well. Even generating these rational numbers takes a prohibitively large time.

Conclusions

- Homomorphic encryption has many practical uses and is theoretically interesting.
- Gentry's work opened the door for fully homomorphic cryptosystems.
- There's still a lot of work to make fully homomorphic encryption feasible.
- Reducing some system parameters shows some progress, but not enough to count as sucessful.

Thanks To

- Dr. Agarwal & Dr. Radziszowski
  for advising me and guiding my work.
- The School of Mathematical Sciences
  for giving me a quality mathematical education.
- Martin van Dijk, Craig Gentry and the IBM Cryptography Research team
  for their prior contributions to the field of homomorphic encryption.

Questions