

# Can homomorphic encryption be practical?

Michael Naehrig

TU/e  
michael@cryptojedi.org

joint work with  
Kristin Lauter and Vinod Vaikuntanathan

Cryptography Working Group  
30 September 2011

## An application scenario – private medical records

- ▶ Health care providers upload all your medical records in public key encrypted form to a cloud service.
- ▶ You control access to your data.
- ▶ You can do encrypted search on your data.
  
- ▶ Different monitoring devices stream encrypted data, e.g. your blood pressure, heart rate, blood sugar level.
- ▶ The cloud service can compute statistical functions on your data, determine risks, send alerts to you, your doctor.
  
- ▶ Parts of this can already be realized (Benaloh et al. 2009).
- ▶ Computing functions on your encrypted data could be done with homomorphic encryption.

# Homomorphic encryption

- ▶ Many crypto systems have homomorphic properties: RSA, ElGamal, Benaloh, Paillier, but only provide additive or multiplicative homomorphism, not both.
- ▶ First system that could do both: Boneh-Goh-Nissim 2005 many additions and one multiplication (uses pairings).
- ▶ Fully homomorphic encryption allows to do arbitrary computations on encrypted data without knowing the secret key,
- ▶ in particular it allows doing an arbitrary number of additions and multiplications.

# Fully homomorphic encryption

Gentry proposed the first fully homomorphic encryption scheme in 2009 based on ideal lattices.

- ▶ The basis is a somewhat homomorphic encryption scheme that can evaluate low-degree polynomials on encrypted data.
- ▶ Ciphertexts are “noisy” and the noise grows slightly during addition and strongly during multiplication.
- ▶ If the SWHE scheme can evaluate its own decryption circuit, then a bootstrapping step can refresh ciphertexts by homomorphically decrypting using an encrypted secret key.
- ▶ Only works by “squashing” the decryption circuit.
- ▶ So far quite inefficient.

# Fully homomorphic encryption

- ▶ Recently, many improvements, but still inefficient. Implementation (Gentry, Halevi 2011),
  - ▶ toy setting: encrypt a bit in 0.2s, decrypt in 6s, public key: 17MB
  - ▶ large setting: encrypt in 3min, decrypt in 31min public key: 2.3GB
- ▶ New variants, mostly following Gentry's blueprint.
- ▶ Recent variants based on the LWE problem or RLWE problem.
- ▶ Applications might not need fully homomorphic encryption, somewhat homomorphic could be sufficient.
- ▶ This talk: somewhat homomorphic encryption scheme by Brakerski and Vaikuntanathan (Crypto 2011) based on RLWE.

# The Learning with Errors (LWE) Problem

(Regev 2005)

Let  $n \in \mathbb{N}$ ,  $q \in \mathbb{Z}$ ,  $\chi$  a probability distribution on  $\mathbb{Z}$ .

Distinguish the following distributions of pairs  $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ :

*Uniform distribution*

- ▶ Sample  $(a_i, b_i) \in \mathbb{Z}_q^{n+1}$  uniformly at random.

*LWE distribution*

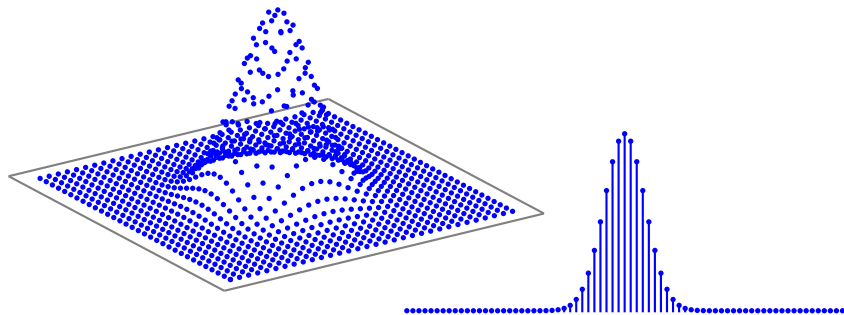
- ▶ Draw  $s \in \mathbb{Z}_q^n$  uniformly at random.
- ▶ Sample  $a_i \in \mathbb{Z}_q^n$  uniformly at random,
- ▶ sample  $e_i \leftarrow \chi$ ,  $\bar{e}_i \in \mathbb{Z}_q$ ,
- ▶ set  $b_i = \langle a_i, s \rangle + \bar{e}_i$ .

The  $b_i$  are noisy inner products of random  $a_i$  with a secret  $s$ .

# The Learning with Errors (LWE) Problem

(Regev 2005)

- ▶ Regev gave a quantum reduction of certain approximate SVP to LWE, i.e. if one can solve LWE, then there's a quantum algorithm to solve certain approximate SVP.
- ▶ Peikert (2009) gave a reduction using classical algorithms
- ▶ Assumption:  $q$  prime,  $\chi$  is a discrete Gaussian error distribution



# The Ring Learning with Errors (RLWE) Problem

(Lyubashevsky, Peikert, Regev 2010)

Here: special case.

- ▶ Let  $n = 2^k$ ,

$$f(x) = x^n + 1$$

( $2n$ -th cyclotomic polynomial).

- ▶ Define ring

$$R = \mathbb{Z}[x]/(f)$$

(ring of integers in  $2n$ -th cyclotomic number field).

- ▶ Let  $q$  be prime, define

$$R_q = R/qR \cong \mathbb{Z}_q[x]/(\overline{f}).$$

- ▶ Let  $\chi$  be an error distribution on  $R$ .



# The Ring Learning with Errors (RLWE) Problem

(Lyubashevsky, Peikert, Regev 2010)

Distinguish the following distributions of pairs  $(a_i, b_i) \in R_q^2$ :

*Uniform distribution on  $R_q^2$*

- ▶ Sample  $(a_i, b_i) \in R_q^2$  uniformly at random.

*RLWE distribution*

- ▶ Draw  $s \in R_q$  uniformly at random.
- ▶ Sample  $a_i \in R_q$  uniformly at random,
- ▶ sample  $e_i \leftarrow \chi, \bar{e}_i \in R_q,$
- ▶ set  $b_i = a_i \cdot s + \bar{e}_i.$

The  $b_i$  are noisy ring (number field) products of random  $a_i$  with a secret  $s$ .

# The Ring Learning with Errors (RLWE) Problem

(Lyubashevsky, Peikert, Regev 2010)

- ▶ Believed to be as hard as general LWE problem, i.e. would be solved with the same algorithms.
- ▶ Though there's a lot more structure!
- ▶ Recent results indicate RLWE problem easier than LWE, (Schneider 2011 claims in practice speedup is linear in  $n$ ).
- ▶ But much more efficient.
- ▶ Smaller keys, very efficient arithmetic in  $R_q$ .

Can be used to build a fully homomorphic encryption scheme.

## Slight modifications

- ▶ In both LWE and RLWE problems, it is okay to sample  $s \leftarrow \chi$  (and not uniformly at random).
  - ▶ Sample until  $(a_0, b_0 = a_0s + e_0)$  with  $a_0 \in R_q^*$  (invertible).
  - ▶ For every additional sample  $(a, b = as + e)$  consider

$$\begin{aligned}(a', b') &= (-a_0^{-1}a, b + a'b_0) \\ &= (a', as + e + a'(a_0s + e_0)) \\ &= (a', as + e - as + a'e_0) = (a', a'e_0 + e)\end{aligned}$$

- ▶ If one can solve RLWE with small secret, then one can solve it with uniform secret.
- ▶ It is also okay to use small multiples of the error terms, i.e. samples  $(a_i, b_i = a_i \cdot s + te_i)$  are still indistinguishable from random. For example, take  $t = 2$ .

# Somewhat homomorphic encryption

(Brakerski, Vaikuntanathan 2011)

## SH.Keygen

- ▶ Sample small element  $s \leftarrow \chi$ .

Set secret key

- ▶  $sk = s$ .

Sample RLWE instance:

- ▶ Sample  $a_1 \leftarrow R_q$  uniformly random,
- ▶ small error  $e \leftarrow \chi$ .

Set public key

- ▶  $pk = (a_0 = -(a_1s + te), a_1)$ .

# Somewhat homomorphic encryption

(Brakerski, Vaikuntanathan 2011)

Message space:

$$R_t = \mathbb{Z}_t[x]/(x^n + 1),$$

$t$  rel. prime to  $q$ , e.g.  $t = 2$ . Encode messages as elements in  $R_q$  with coefficients mod  $t$ .

- ▶ Can encode  $n$  bits at once.

## SH.Enc

Given  $pk = (a_0, a_1)$  and a message  $m \in R_q$ ,

- ▶ sample  $u \leftarrow \chi$ , and  $f, g \leftarrow \chi$ ,

Set ciphertext

- ▶  $ct = (c_0, c_1) := (a_0u + tg + m, a_1u + tf)$ .

# Somewhat homomorphic encryption

(Brakerski, Vaikuntanathan 2011)

## SH.Dec

Given  $sk = s$  and a ciphertext  $ct = (c_0, c_1)$ ,

- ▶ compute  $\tilde{m} = c_0 + c_1 s \in R_q$ .

Output the message

- ▶  $\tilde{m} \bmod t$ .

Correctness:

$$\begin{aligned}\tilde{m} = c_0 + c_1 s &= (a_0 u + t g + m) + (a_1 u + t f) s \\ &= -(a_1 s + t e) u + t g + m + a_1 u s + t f s \\ &= m + t(g + f s - e u).\end{aligned}$$

Reduction modulo  $t$  gives back  $m$  as long as the error terms are not too large. Gives bound on standard deviation of the Gaussian.

# Somewhat homomorphic encryption

(Brakerski, Vaikuntanathan 2011)

Homomorphic operations

## SH.Add

Given  $ct = (c_0, c_1)$  and  $ct' = (c'_0, c'_1)$ , set the new ciphertext

- ▶  $ct_{\text{add}} = (c_0 + c'_0, c_1 + c'_1)$   
 $= (a_0(u + u') + t(g + g') + (m + m'), a_1(u + u') + t(f + f'))$ .

## SH.Mult

Given  $ct = (c_0, c_1)$  and  $ct' = (c'_0, c'_1)$ ,

- ▶ compute  
 $(c_0 + c_1X)(c'_0 + c'_1X) = c_0c'_0 + (c_0c'_1 + c'_0c_1)X + c_1c'_1X^2$
- ▶  $ct_{\text{mlt}} = (c_0c'_0, c_0c'_1 + c'_0c_1, c_1c'_1)$

Errors multiply!

$$(m + t(g + fs - eu))(m' + t(g' + f's + eu')) = mm' + t(\dots)$$

# Somewhat homomorphic encryption

(Brakerski, Vaikuntanathan 2011)

- ▶ Homomorphic operations increase size of error terms.
- ▶ Homomorphic multiplication increases the size of the ciphertext.
- ▶ Homomorphic addition, multiplication, and decryption generalize to longer ciphertexts.

## SH.Dec

Given  $sk = s$  and a ciphertext  $ct = (c_0, c_1, \dots, c_\delta)$ ,

- ▶ compute  $\tilde{m} = \sum_{i=0}^{\delta} c_i s^i \in R_q$ .

Output the message

- ▶  $\tilde{m} \pmod{t}$ .



# Relinearization

(Brakerski, Vaikuntanathan 2011)

There is a way to go from 3-element ciphertext  $ct = (c_0, c_1, c_2)$  back to a 2-element ciphertext.

- ▶ We have

$$c_2 s^2 + c_1 s + c_0 = te_{\text{mult}} + mm'$$

- ▶ Publish a “homomorphism key”

$$h_i = (a_i, b_i = -(a_i s + te_i) + t^i s^2) \quad \text{for } i = 0, \dots, \lceil \log_t q \rceil - 1$$

- ▶ Write  $c_2$  in its base- $t$  representation  $c_2 = \sum c_{2,i} t^i$ .

# Relinearization

(Brakerski, Vaikuntanathan 2011)

- ▶ Replace  $ct$  by  $(c_0^{\text{relin}}, c_1^{\text{relin}})$  with

$$c_1^{\text{relin}} = c_1 + \sum_{i=0}^{\lceil \log_t q \rceil - 1} c_{2,i} a_i, \quad c_0^{\text{relin}} = c_0 + \sum_{i=0}^{\lceil \log_t q \rceil - 1} c_{2,i} b_i$$

- ▶ Then

$$c_0^{\text{relin}} + c_1^{\text{relin}} s = c_0 + c_1 s + c_2 s^2 - t e_{\text{relin}}$$

$$c_0^{\text{relin}} + c_1^{\text{relin}} s = t(e_{\text{mult}} - e_{\text{relin}}) + mm'$$

- ▶ Okay, ciphertext is smaller, but error has increased!
- ▶ Decryption still correct if final error  $e_{\text{mult}} - e_{\text{relin}}$  is small enough.

## Specific parameter choices

Choosing parameters to “guarantee” security and correctness.

Correctness:

- ▶  $q$  must be large enough when compared to the size of the error terms and  $t$ .
- ▶ I.e. parameters are chosen s.t. the scheme can evaluate polynomials of a certain fixed degree  $D$  ( $D - 1$  multiplications and a bunch of additions).

Security:

- ▶ Against distinguishing attack with advantage  $2^{-32}$  by Micciancio/Regev 2009.
- ▶ Adjust analysis of Lindner/Peikert 2011 to our setting.
- ▶ Still assume RLWE is no easier than LWE.

## Specific parameters, key and ciphertext sizes

$t$	$D$	$n$	$\lceil \lg(q) \rceil$	$\lg(T)$	$l_{R_q}/10^3$	$(2 + \log_t q) \cdot l_{R_q}/10^3$
2	1	512	19	123	10	205
	2	1024	38	107	39	1557
	3	2048	64	134	132	8651
	5	4096	120	145	492	59966
	10	8192	264	117	2163	575276
1024	1	1024	30	164	31	154
	2	2048	58	164	119	927
	3	4096	95	215	390	4475
	5	8192	171	242	1401	26756
	10	16384	368	214	6030	233938

# Message encoding

Homomorphic operations reflect operations in  $R_t$ .

- ▶ Want operations on integers.
- ▶ Encode an integer  $m = (m_0, m_1, \dots, m_l)_2$ ,  $m_i \in \{0, 1\}$  as a polynomial of degree  $l$  with coefficients  $m_i$ . Get back  $m$  by evaluating at 2.
- ▶  $t = 2$  not useful for addition and multiplication since operations mod 2 are different from integer operations.
- ▶ Choose  $t$  large enough to allow for enough additions.
- ▶ Reduction modulo  $x^n + 1$  screws up integer multiplication.
- ▶ Choose  $l$  small enough to allow a certain number of multiplications before reaching degree  $n$ .

## Reference implementation

Implementation using the computer algebra system Magma

- ▶ Uses polynomial arithmetic in Magma,
- ▶ no specific optimization for multiplication, no DFT,
- ▶ no optimization for specific parameters (sizes),
- ▶ decryption for arbitrary length ciphertexts.

Big potential to improve efficiency

- ▶ Main cost is polynomial multiplication modulo  $x^n + 1$  in  $R_q$ .

# Timings

Intel Core 2 Duo @ 2.1 GHz

$t$	$D$	$n$	$\lceil \lg(q) \rceil$	$S_x$	Enc	Dec		Mult	Mult
				ms	prec. ms	deg 1 ms	deg 2 ms	ms	degred s
2	1	512	19	27	2	2	—	—	—
	2	1024	38	55	9	6	10	15	0.3
	3	2048	64	110	29	18	33	56	2.0
	5	4096	120	223	85	49	94	163	10.6
	10	8192	264	438	425	227	454	887	114.6
1024	1	1024	30	54	5	4	—	—	—
	2	2048	58	110	24	15	26	41	0.2
	3	4096	95	221	81	46	88	154	1.0
	5	8192	171	440	275	148	288	526	5.3
	10	16384	368	868	1260	664	1300	1593	48.2

- ▶ Compute the ciphertext of the sum of 100 numbers of size 128 bits from the single ciphertexts (for mean computation): < 20ms
- ▶ Ciphertexts for the sum and sum of squares of 100 such numbers (for mean and variance): < 6s

## Can homomorphic encryption be practical?

- ▶ Maybe...
- ▶ ...if somewhat homomorphic encryption is enough, then probably yes ...
- ▶ ...and if we are not too much off with our parameter choices ...
- ▶ ...maybe even fully homomorphic encryption...