

# Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes

Nigel Smart and Fre Vercauteren

Department of Computer Science,  
University Of Bristol,  
and  
COSIC - Electrical Engineering,  
Katholieke Universiteit Leuven.

May 9, 2010

# Introduction

The big cryptographic breakthrough in 2009 was Gentry's **Fully Homomorphic Encryption** scheme.

Gentry's scheme is based on two key ideas:

- ▶ Using **ideal lattices**.
- ▶ Obtaining a fully homomorphic scheme from a **bootstrappable somewhat homomorphic scheme**.

In this talk we specialise Gentry's scheme in one particular way

- ▶ Producing a scheme which is conceptually easier to understand
- ▶ Producing a scheme which is much more efficient
- ▶ Removing (almost) all need to understand lattices

# Homomorphic Encryption Schemes

These are given by five algorithms:

KeyGen()

Encrypt( $M$ , PK)

Decrypt( $c$ , SK)

Add( $c_1$ ,  $c_2$ , PK)

Mult( $c_1$ ,  $c_2$ , PK)

# Homomorphic Encryption Schemes

We assume the message space is some form of ring

Apart from the usual property that decrypting an encryption of a message should give you what you started with we also require the following two properties:

$$\text{Decrypt}(\text{Add}(c_1, c_2, \text{PK}), \text{SK}) = \text{Decrypt}(c_1, \text{SK}) + \text{Decrypt}(c_2, \text{SK}).$$

$$\text{Decrypt}(\text{Mult}(c_1, c_2, \text{PK}), \text{SK}) = \text{Decrypt}(c_1, \text{SK}) \times \text{Decrypt}(c_2, \text{SK}).$$

With these properties the scheme can evaluate arithmetic circuits over the ring.

# Gentry's Scheme: 50000 feet

A **somewhat homomorphic scheme** is one which can evaluate homomorphically all arithmetic circuits from a given set  $\mathcal{C}$  (e.g. bounded depth).

A **bootstrappable** homomorphic encryption scheme is one for which the set  $\mathcal{C}$  contains its own decryption circuit (plus a little more).

- ▶ For large enough values of the security parameter

The main problem with Gentry's construction and ours from a practical perspective is that the security parameter has to be infeasibly large for bootstrapping to be possible.

- ▶ The appendix of our paper analyses this in great detail

## Gentry's Scheme: 10000 feet

Gentry essentially does the following

- ▶ Public key is a “large” representation of an ideal  $J$  in a large degree number field, plus a coprime “small” ideal  $I$ .
- ▶ Private key is a “nice” representation for the ideal  $J$ .

To encrypt we

- ▶ Take a message  $m \in \mathcal{O}/I$
- ▶ Add on random “small” element from the ideal  $I$
- ▶ Reduce the result modulo the public ideal  $J$ .
- ▶  $i$  needs to be **large** enough to give semantic security.
- ▶  $i$  needs to be **small** enough to enable decryption to work.

$$c = m + i \pmod{J}$$

To decrypt we use the knowledge of the “nice” representation to recover  $m \pmod{I}$ .

Note we ignore all talk of lattices here.

# Gentry's Scheme: 10000 feet

Why is this homomorphic?

$$\begin{aligned}c_1 + c_2 &= (m_1 + i_1 \pmod{J}) + (m_2 + i_2 \pmod{J}) \\ &= (m_1 + m_2) + (i_1 + i_2) \pmod{J}\end{aligned}$$

$$\begin{aligned}c_1 \cdot c_2 &= (m_1 + i_1 \pmod{J}) \cdot (m_2 + i_2 \pmod{J}) \\ &= (m_1 \cdot m_2) + (i_1 \cdot i_2 + i_1 \cdot m_2 + i_2 \cdot m_1) \pmod{J}.\end{aligned}$$

So we see that Add and Mult result in ciphertexts of the same form, since we are working with ideals, but the associated small element of  $I$  has got larger.

- ▶ We say the output ciphertext has become dirtier than the input ones

This will keep working as long as the ciphertext does not become too dirty, since then we will not be able to decrypt.

# Gentry's Scheme: 1000 feet

But how do you represent ideals?

- ▶ An ideal is a  $\mathbb{Z}$ -module, so we represent it via a basis.
- ▶ So an ideal is a lattice (sorry they will go in a moment)

Elements in the ring  $\mathcal{O}$  are represented as vectors.

Public key is now

- ▶ Matrix  $B_I$  representing the basis of  $I$
- ▶ Matrix  $B_J$  representing the “big” basis of  $J$

Secret key is a matrix  $S_J$  representing the “small” basis of  $J$ .

## Gentry's Scheme: 1000 feet

Now for a vector in  $\mathbf{m} \in \mathcal{O}/I$  we encrypt by forming

$$\mathbf{c} = \mathbf{m} + B_I \cdot \mathbf{r} \pmod{B_J}$$

for some “small” vector  $\mathbf{r} \in \mathbb{Z}^n$ .

To decrypt we compute

$$\mathbf{m} = (\mathbf{c} \pmod{S_J}) \pmod{B_I}$$

Where

$$\mathbf{x} \pmod{B} = \mathbf{x} - B \cdot \lfloor B^{-1} \mathbf{x} \rfloor$$

for any basis matrix  $B$

# Gentry to SV Scheme

We obtain our scheme by performing four specialisations.

- ▶ Each one results in a computationally more efficient scheme
- ▶ Also enables greater functionality
- ▶ Reduces bandwidth as well

## Specialisation 1:

Gentry in one example suggests taking  $J$  to be principal and having  $S_J$  being the principal generator.

So lets do that

- ▶ Matrix  $S_J$  is replaced by a polynomial  $G(X)$  which represents an element in  $\mathcal{O}$ .

## Specialisation 2:

Gentry also suggests take  $I$  to be the ideal (2), we shall as well.

# Gentry to SV Scheme

## Specialisation 3:

Instead of taking the  $n$ -element  $\mathbb{Z}$ -module representation of the ideal  $J$  as the public basis, take the 2-element  $\mathcal{O}$ -module representation.

This is a standard trick in computational number theory.

The public basis of  $J$  now consists of two elements

- ▶ An integer (normally the norm of the ideal)
- ▶ A monic polynomial of degree  $m \leq n$  with coefficients in  $\mathbb{Z}$

This does not seem to gain much since if  $m = n$  we obtain the same size of the public key as Gentry

# Gentry to SV Scheme

## Specialisation 4:

Choose  $m = 1$ , and to make life slightly simpler  $\text{Nm}(J) = p$  a prime.

So the public key is a pair

$$(p, X - \alpha) \equiv (p, \alpha)$$

where  $\alpha \in \mathbb{F}_p$ .

We can now write the entire scheme (more simply) just using polynomial arithmetic

- ▶ Which we do in the following slides

# The SV Scheme

For a positive value  $r$ , we define the following type of “ball” of polynomials in  $\mathbb{Z}[X]$  centered at the origin:

$$\mathbb{B}_{\infty, N}(r) = \left\{ \sum_{i=0}^{N-1} a_i x^i : -r \leq a_i \leq r \right\}.$$

We also define the following half-ball

$$\mathbb{B}_{\infty, N}^+(r) = \left\{ \sum_{i=0}^{N-1} a_i x^i : 0 \leq a_i \leq r \right\}.$$

# The SV Scheme

## KeyGen()

- ▶ Set the plaintext space to be  $\mathbb{B}_{\infty, N}^+(2)$ .
- ▶ Choose a degree  $N$  monic irreducible polynomial  $F(x) \in \mathbb{Z}[x]$ .
- ▶ Repeat:
  - ▶  $S(x) \leftarrow_R \mathbb{B}_{\infty, N}(\eta/2)$ .
  - ▶  $G(x) \leftarrow 1 + 2 \cdot S(x)$ .
  - ▶  $p \leftarrow \text{resultant}(G(x), F(x))$ .
- ▶ Until  $p$  is prime.
- ▶  $D(x) \leftarrow \text{gcd}(G(x), F(x))$  over  $\mathbb{F}_p[x]$ .
- ▶ Let  $\alpha \in \mathbb{F}_p$  denote the unique root of  $D(x)$ .
- ▶ Apply the XGCD-algorithm over  $\mathbb{Q}[x]$  to obtain  $Z(x) = \sum_{i=0}^{N-1} z_i x^i \in \mathbb{Z}[x]$  such that
$$Z(x) \cdot G(x) = p \pmod{F(x)}.$$
- ▶  $B(x) \leftarrow Z(x) \pmod{2p}$ .
- ▶ The public key is  $\text{PK} = (p, \alpha)$ , whilst the private key is  $\text{SK} = (p, B)$ .

# The SV Scheme

Encrypt( $M, PK$ )

- ▶ Parse PK as  $(p, \alpha)$ .
- ▶  $R(x) \leftarrow_R \mathbb{B}_{\infty, N}(\mu/2)$ .
- ▶  $C(x) \leftarrow M + 2 \cdot R(x)$ .
- ▶  $c \leftarrow C(\alpha) \pmod{p}$ .
- ▶ Output  $c$ .

Note  $c \in \mathbb{F}_p$ , a single number only!

Decrypt( $c, SK$ )

- ▶ Parse SK as  $(p, B)$ .
- ▶  $M \leftarrow (c - \lfloor c \cdot B/p \rfloor) \pmod{2}$ .
- ▶ Output  $M$ .

Where  $\lfloor c \cdot B/p \rfloor$  means apply the rounding to every coefficient of the polynomial.

# The SV Scheme

Add( $c_1, c_2, PK$ )

- ▶ Parse PK as  $(p, \alpha)$ .
- ▶  $c_3 \leftarrow (c_1 + c_2) \pmod{p}$ .
- ▶ Output  $c_3$ .

Mult( $c_1, c_2, PK$ )

- ▶ Parse PK as  $(p, \alpha)$ .
- ▶  $c_3 \leftarrow (c_1 \cdot c_2) \pmod{p}$ .
- ▶ Output  $c_3$ .

## Good Properties of the SV Scheme

As decryption works on each coefficient of the polynomial  $c \cdot B(x)/p$  in turn

- ▶ Reencryption (cleaning the ciphertext) can also be applied to each coefficient in turn
- ▶ This means we can obtain (theoretically) fully homomorphic encryption over the field  $\mathbb{F}_{2^N}$
- ▶ Hence over any subfield of  $\mathbb{F}_{2^N}$
- ▶ Gentry only obtains (theoretically) fully homomorphic encryption over  $\mathbb{F}_2$ .

The parameters  $\mu$  and  $\eta$  control how “deep” a circuit our somewhat homomorphic scheme can cope with

$$d \log 2 \leq \log \log \left( \frac{\sqrt{N} \cdot \eta}{2 \cdot \delta_\infty} \right) - \log \log(\delta_\infty \cdot \mu).$$

where  $\delta_\infty$  depends on the polynomial  $F(x)$ .

# Bad Properties of the SV Scheme

To get a fully homomorphic scheme we need the depth to be around 7 or 8.

In practice we can only generate keys which can cope with a depth of up to about 1.7.

Even these keys have a  $p$  of 92681 bits in length!

- ▶ But at least encryption/decryption is efficient at this level.

Gentry's original scheme has the same problem, but more so as encryption and decryption are more complicated than our scheme

See our paper for a fully detailed analysis of the problem of bootstrapping, as well as implementation details.

# Security of the SV Scheme

Cryptographers care about semantic security, but often the associated problem is not interesting from a computational number theory perspective.

Consider ElGamal:

- ▶  $DDH \equiv$  Semantic Security: CNT not interested in DDH.
- ▶  $CDH \equiv$  Message Recovery: CNT not interested much in CDH.
- ▶  $DLP \equiv$  Key Recovery: DLP well studied problem in CNT.

We feel not enough to give yet another hard problem which no one will study to justify semantic security

- ▶ Better to also give well studied problems which imply weaker properties than semantic security
- ▶ This aids confidence in the security of the scheme which provable security on its own cannot do.

# Semantic Security

Semantic security of our scheme can be reduced to the following problem:

## Polynomial Coset Problem (PCP)

The challenger first selects  $b \leftarrow_R \{0, 1\}$  and runs KeyGen as above to obtain a value of  $\alpha$  and  $p$ .

If  $b = 0$  then the challenger performs

- ▶  $R(x) \leftarrow_R \mathbb{B}_{\infty, N}(\mu)$ .
- ▶  $r \leftarrow R(\alpha) \pmod{p}$ .

Whilst if  $b = 1$  the challenger performs

- ▶  $r \leftarrow_R \mathbb{F}_p$ .

Given  $(r, PK)$  the problem is to guess whether  $b = 0$  or  $b = 1$ .

# Semantic Security

Note that the size of the space  $\mathbb{B}_{\infty, N}(\mu)$  is roughly  $\mu^N$ , whereas  $\mathbb{F}_p$  has size  $\eta^N$ .

So if  $\mu$  is much smaller than  $\eta$ , we are trying to distinguish a relatively small space within a larger one.

We call the problem the **Polynomial Coset Problem** as it is akin to Gentry's **Ideal Coset Problem**.

# Message Recovery

The best actual “attack” on our scheme we know of is one which tries to recover the message.

We essentially want to recover a polynomial  $C(x)$  with “small” coefficients such that

$$C(\alpha) = c \pmod{p}.$$

This is a **very** well studied problem in computational number theory and the best solution is via **lattice-basis reduction**

- ▶ Indeed via the “classic” lattice of the identity matrix with one non-trivial row.

It is this attack which gives us our way of estimating the security parameters.

# Key Recovery

As usual key recovery provides the most “famous” hard problems.

Recall our public key is the 2-element representation  $(p, X - \alpha)$  of an ideal  $J$ .

The private key is a “small” generator of this ideal.

This is a well studied problem in number theory:

- ▶ Indeed our scheme seems to be the first **cryptographically interesting** scheme whose security is based on a number theoretic problem which is not from **elementary number theory**.

We know of two methods to solve the key recovery problem

# Key Recovery

## Method 1

- ▶ Apply the sub-exp class group algorithms to obtain the set of fundamental units and the class group
- ▶ “Smooth” the ideal  $J$  so as it factors over the factorbase
- ▶ Obtain a generator as a product of small elements
- ▶ Produce a small element using fundamental units

Unclear whether the last step can be done in sub-exp time!

## Method 2

Directly solve the small generator problem using a Baby-Step/Giant-Step style method attributed to Buchmann. This last method has exponential complexity in the norm of the ideal

## Open Problem:

Determine if key recovery can be done in sub-exponential time.

# Conclusion

We have a variant of Gentry's scheme which

- ▶ Has smaller general ciphertext size
- ▶ Has smaller key size
- ▶ Theoretically can be made fully homomorphic over  $\mathbb{F}_{2^N}$  as opposed to  $\mathbb{F}_2$ .
- ▶ Security (key recovery) is related to a well studied problem in classical computational number theory.
- ▶ Can be described without resorting to lattices. This may aid future work on this or other schemes.

However

It is still not a practical fully homomorphic scheme!

Any Questions?