Cryptanalysis of Brenner et al.'s Somewhat Homomorphic Encryption Scheme

Russell Paulet **Xun Yi**

School of Engineering and Science
Victoria University, Australia
Email: russell.paulet@live.vu.edu.au, xun.yi@vu.edu.au

Abstract

Recently, Brenner et al. proposed a symmetric somewhat homomorphic encryption scheme and applied it to solve some practical problems, such as the Millionaires' problem, which only need to evaluate circuits of limited depth. It is claimed that the security of their scheme is built on the hardness of integer factorization. In this paper, we use the Euclidean Greatest Common Divisor (GCD) algorithm to perform cryptanalysis on Brenner et al.'s scheme. We present several algorithms to find the secret key of their scheme. Our experiments have shown that our cryptanalysis is feasible and efficient.

Keywords: encryption,cryptanalysis,privacy

1 Introduction

Since the introduction of public key cryptography (Diffie & Hellman 1976, Rivest, Shamir & Adleman 1978) there has been great interest their homomorphic properties. In general terms, this means being able to operate on the encrypted data as you would on unencrypted data. This idea was suggested by (Rivest, Adleman & Dertouzos 1978) under the term *privacy homomorphisms*.

Since then there has been great interest in homomorphic encryption, and the ability to construct a Fully Homomorphic Encryption (FHE) scheme. That is, having a complete set of operations as one does with the unencrypted data. The first provably secure solution was given by (Gentry 2009), which confirmed that such a scheme is possible.

All efforts before Gentry's breakthrough result were only partially or half homomorphic. In other words, they were homomorphic under one algebraic operation: either addition (Paillier 1999) or multiplication (Rivest, Shamir & Adleman 1978, ElGamal 1985). Although, these partially homomorphic encryption schemes still had their uses (e.g. Paillier is popular with voting systems), a scheme that would allow both simultaneously would prove exceedingly useful.

The initial result by Gentry was constructed using the properties of ideal lattices. Using them, he was able to achieve a limited additions and multiplications. This limitation existed because for each addition and multiplication operation, the noise components of each ciphertext were combined. Eventually, the ciphertexts would become too noisy and fail to decrypt properly. Realising that the decryption function for the scheme could be represented as a combination of Boolean gates, it might be possible for the scheme to evaluate its own decryption circuit and reduce the noise associated with a ciphertext.

Since the unmodified decryption circuit had greater depth than the scheme could evaluate, the decryption circuit had to be compressed (squashed) to permit this process. Many FHE encryption schemes (Smart & Vercauteren 2010, van Dijk, Gentry, Halevi & Vaikuntanathan 2010) followed this approach: create a somewhat encryption scheme (homomorphic for a limited number of operations) and then bootstrap the process (enable the evaluation of its own decryption circuit). In particular, the scheme by (Smart & Vercauteren 2010) represents lattices by using large integers, simplifying the manipulations. The scheme by (van Dijk et al. 2010), on the other hand required no knowledge of lattices for its basic construction. There have been improvements, due to (Stehle & Steinfeld 2010), to increase the performance of the original scheme proposed by Gentry. So far, schemes that require the bootstrapping step has been shown to be too inefficient for practical usages (Gentry & Halevi 2011).

Different optimisations and techniques have been introduced to improve the performance of fully homomorphic schemes. The original fully homomorphic encryption scheme was based on ideal lattices, which contained additional structure that potentially be exploited. To overcome this potential weakness, a scheme based on the standard Learning With Errors (LWE) problem was introduced (Brakerski & Vaikuntanathan 2011). The scheme included a unique re-linearisation that was used after multiplication of two ciphertexts to transform a quadratic into a standard linear size ciphertext.

A modulus switching technique has been proposed as a better method for managing the noise associated with ciphertexts (Brakerski, Gentry & Vaikuntanathan 2012). Essentially, we can choose a large q and have a series of levels down to zero, where at each level we scale back the ciphertext by (p/q), which reduces the noise without requiring the bootstrapping procedure.

Very recently, a new technique was explored to achieve SIMD fully homomorphic encryption (Gentry, Halevi & Smart 2012). Due to the noise associated with adding and multiplying, they design a scheme whereby many messages can be packed into one ciphertext. Hence, we can add and multiply ciphertexts and affect many messages. They also describe a clever permutation trick that allows one to achieve a complete set of operations.

Certain applications, like cloud computing, require

This paper appeared at Australasian Information Security Conference (ACSW-AISC), Adelaide, Australia. It is published as Conferences in Research and Practice in Information Technology, Vol. 138, Eds. C. Thomborson and U. Parampalli.

endless computation on data since we want to delegate our data processing and storage to a third party. However, it seems that a somewhat homomorphic scheme (without bootstrapping) is also useful if the function to be evaluated is simple and a known stopping point exists. In other words, we know the exact function we wish to compute, and we can set the parameters of the scheme such that it avoids decryption errors.

The security of these somewhat homomorphic encryption schemes are based on new hardness assumptions. These are not as well known and not as well studied as classical security assumptions like the discrete logarithm problem. A recent work by (Brenner, Perl & Smith 2012) aims to fill this gap by constructing a somewhat homomorphic encryption scheme that is as hard as factoring a large semiprime integer.

1.1 Our Contributions

Brenner et al. proposed a symmetric somewhat homomorphic encryption scheme and applied it to solve some practical problems, such as the Millionaires' problem (Yao 1982), which only requires the evaluation of bounded depth circuits.

They claim the security of their scheme is built on the hardness of integer factorization. As such, they choose a large prime number for the private key, so that they can justify reducing the relative noise associated with a ciphertext.

In this paper, we use the Euclidean Greatest Common Divisor (GCD) algorithm to perform cryptanalysis on Brenner et al.'s scheme. We present several algorithms to find the secret key of their scheme. Our algorithms are built on the two characteristics of Brenner et al.'s scheme, i.e., a prime number as secret key and small initial noises.

Our experiments demonstrate that our cryptanalysis is feasible and efficient based on the proposed parameters of the Brenner et al.'s scheme, and the result can be verified when we have access to an encryption oracle. The likely way to subvert our attack is to increase the noise dramatically.

1.2 Paper Organisation

The rest of the paper is organised as follows. Section 2 will summarise and review the somewhat homomorphic encryption scheme of (Brenner et al. 2012) and associated security claims. Section 3 will present a cryptanalysis of the encryption scheme and design an attack. Section 4 will describe and conduct an experiment based on the attack described in Section 3. Section 5 will conclude this paper and make recommendations for further research.

2 Brenner et al.'s Homomorphic Scheme

This section reviews the symmetric somewhat encryption scheme due to (Brenner et al. 2012). We review the basic construction, homomorphic properties, parameter choice and security claims.

2.1 Basic Construction

The scheme has the following parameters:

- the security parameter λ
- the bit length of the initial noise η
- the modulus p, which is a large prime integer of order 2^λ
- the bit length ρ of the message space, which is $\lambda - \eta$

Based on these security parameters, the scheme is composed as a tuple $(P, C, K, E, D, \oplus, \otimes)$, where elements are defined as the following:

P is the plaintext and contains elements from $\mathbb{N}^+$ limited by the prime integer p of order 2^λ such that for two plaintext operands $a, b \in \mathbb{N}^P, a \cdot b < p$ and $\mathbb{N}^P := \{x | x < 2^\eta\}$

C is the ciphertext space and contains elements from $\mathbb{N}^+$.

K is the key generator. The secret key is a large prime p, with an auxiliary public compression argument d with $d \leftarrow 2s + rp$, where $r \in \mathbb{N}^+$ and $s \in \mathbb{N}^C$ with $\forall x \in \mathbb{N}^C, \forall y \in \mathbb{N}^P, 2x < y$.

E is the encryption function. A bit value b is encrypted by picking a random integer a where $a \equiv b \ (mod\ 2)$ and adding a random multiple of the prime p, as $a' = a + (rp)$. For security, r must contain at least one large prime factor of order 2^λ. The noise component must belong to the set of positive natural numbers $a \in \mathbb{N}^+$.

D is the decryption function. The message is recovered by applying the modulus $p : a = a' \ (mod\ p)$.

$\oplus$ and $\otimes$ are the addition and multiplication on the ciphertext, respectively. Since these operations are similar to the related operations on the plaintext, they are mixed additive and multiplicative. In other words, the homomorphic properties are available to those who do not have the ability to encrypt.

2.2 Homomorphic Properties

The correctness of the homomorphic operations $\oplus$ and $\otimes$ when given two ciphetexts a' and b' are proven by Equations 1 and 2. The correctness still holds using the compression argument, under the condition that the noise is sufficiently small.

$$\begin{aligned} a' \oplus b' &= (a + r_1 p) + (b + r_2 p) \\ &= a + b + (r_1 + r_2)p \\ &= a + b \ (mod\ p) \end{aligned} \tag{1}$$

$$\begin{aligned} a' \otimes b' &= (a + r_1 p)(b + r_2 p) \\ &= ab + a(r_2 p) + b(r_1 p) + (r_1 r_2)p^2 \\ &= ab \ (mod\ p) \end{aligned} \tag{2}$$

The somewhat homomorphic scheme is also correct for one ciphertext and one plaintext, which is called mixed additive and multiplicative. This is shown below in Equations 3 and 4, where b is an integer with the same parity as the message.

$$\begin{aligned} a' + b &= (a + r_1 p) + b \\ &= a + b \ (mod\ p) \end{aligned} \tag{3}$$

$$\begin{aligned} a' \otimes b &= (a + rp)b \\ &= ab + rpb \\ &= ab \ (mod\ p) \end{aligned} \tag{4}$$

2.3 Parameter Choice

The scheme requires that the prime key p and factor r must be sufficiently large to make factorization infeasible. The RSA-challenges suggest that each factor be 512 bits in length, resulting in a 1024 integer. This will prevent a factorization attack on a single ciphertext given current technology. The proposed size for the initial noise of a ciphertext is 8 bits.

2.4 Security Claims

The somewhat homomorphic encryption scheme that has been revised here is symmetric. Simply, this means that there is no public key for encryption. However, the scheme still has utility as the homomorphic properties are possible without such information. We briefly recall the Lemmas of (Brenner et al. 2012) and their implications for security.

Lemma 1. *Let the Parameters $(p, q) \in (N)^+$ be of order 2^λ. Any Attack $\mathcal{A}$ running in time polynomial in λ against the encryption scheme can be converted into an algorithm $\mathcal{B}$ for solving the integer factorization problem of any composite integer number (pq) running in time polynomial in λ.*

This Lemma asserts that given a ciphertext $a' = (a+rp)$, according to the encryption function, then it is difficult to find the message, where r is composed of at least one prime q of order λ. Under this scenario, even if we remove the noise component a (which has the same parity as the message), then we would have to factor pq.

When the product of pq is sufficiently large and p and q are roughly the same size, this is a hard problem. The next Lemma, however, introduces a security concern.

Lemma 2. *The security of the encryption scheme is IND-CPA equivalent and the success probability $|Pr[Exp_{ind-cpa} = 1] - \frac{1}{2}|$ is negligible in λ for any $\mathcal{A}$ from PPT (probabilistic polynomial time) function.*

This Lemma builds on the previous assertion about security under the indistinguishability under a chosen plaintext attack (IND-CPA) model. This model allows an adversary $\mathcal{A}$ to sample as many ciphertexts from an encryption oracle $\mathcal{O}_{Enc}$ as it desires. Then it is asked to distinguish an encryption of a message from random. Since all ciphertexts have a large common element p, we can show that this assumption presents a problem. We explore this issue in more detail next.

3 Cryptanalysis

The security goal for the homomorphic scheme presented by (Brenner et al. 2012) is indistinguishability under a chosen plaintext attack (IND-CPA). That is, if an adversary $\mathcal{A}$ has access to an encryption oracle $\mathcal{O}_{Enc}$, then they cannot distinguish between an encryption of a message and a random number greater than a negligible probability. When we examine this scheme under this model we find that, due to the nature of the setup, information about the secret key p can leak.

Even if we don't allow an adversary $\mathcal{A}$ access to $\mathcal{O}_{Enc}$, partial or complete information about the secret key can be discovered from the cihpertexts alone. Intuitively, this information leakage depends on the security parameters. We will look at these two modes of attack, a Chosen Plaintext Attack and Ciphertext Only Attack, in more detail with respect to the proposed security parameters: $\lambda = 512$ and $\eta = 8$.

3.1 Chosen Plaintext Attack

If an adversary $\mathcal{A}$ has access to an encryption oracle $\mathcal{O}_{Enc}$, then this essentially allows $\mathcal{A}$ to encrypt without complete control over the input. In terms of this scheme, the adversary can only control the parity of a. So, under the IND-CPA model we can sample N ciphertexts and then try to find p. If m and n are both exact multiples of p, then it is trivial to find p due to the Euclidean algorithm (see Algorithm 1).

Algorithm 1 $GCD(m, n)$

Input: m, n
Output: the greatest common divisor
1: **while** $n <> 0$ **do**
2: **if** $m > n$ **then**
3: $l = n$
4: $n = m \ (mod \ n)$
5: $m = l$
6: **end if**
7: **end while**

There are two good properties about this algorithm. First it runs in logarithmic time of its inputs. Second we do not need to factorise either m or n in order to find the greatest common divisor. In the case of the somewhat homomorphic encryption scheme, we have a priori knowledge of p. From the description of the encryption scheme we know that p is a prime number.

This scheme is very similar to the first somewhat homomorphic scheme over the integers (van Dijk et al. 2010), where the security is reduced to the approximate-gcd problem, but we only required that p be an odd number. This distinction gives us some more power to find p, as we are able to eliminate potential values of p using efficient prime number tests. Thus we have a new variant of the approximate-gcd problem, which is given by the following definition.

Definition (Prime-Valued Approximate-GCD Problem) 1. *Let c_i be N ciphertexts for $1 \leq i \leq N$, where $c_i = a_i + q_i p$ with a_i and q_i chosen from suitable random distributions, and each a_i has the same parity of the message m_i. The problem is to find prime p.*

From this definition we can build a procedure that can find the private key p, since we have the additional fact that p is a prime. As a sketch of a simple procedure for finding p, we outline the following steps (with respect to the IND-CPA model).

1. Sample 2 ciphertexts c_1 and c_2 from the encryption oracle $\mathcal{O}_{Enc}$.
2. Generate a set of potential candidates C of p from the two ciphertexts c_1, c_2 by iterating over all possible noise values.
3. Eliminate invalid elements from the set of candidate values C that do not match the characteristics of p: a prime and an integer of λ bits.
4. Test the set of potential values of p by sampling ciphertexts c_i, for $1 \leq i \leq N$, from the encryption oracle $\mathcal{O}_{Enc}$ and testing whether decryption works correctly.

With this sketch in mind we present Algorithm 2 that generates a set of candidates from two ciphertexts. This algorithm iterates over the possible noise values, and generates the resulting GCD.

Under ideal conditions, we perform two tests. We test that the candidate c is prime using the *is_prime*

Algorithm 2 $GenerateCandidates(x, y, \eta)$

Input: x, y as ciphertexts and η as initial noise size
Output: C set of candidate values of p

$C \leftarrow \emptyset$
for $i = 1 \rightarrow 2^{\eta}$ **do**
  for $j = 1 \rightarrow 2^{\eta}$ **do**
    $x' = x - i$
    $y' = y - j$
    $C \leftarrow C \cup GCD(x', y')$
  end for
end for
return C

function. The function *is_prime* can be implemented by using methods such as the Miller-Rabin primality test or the Fermat primality test. We also test that p within one bit either side of λ. These two tests allow us to reduce the number of candidates to a more manageable size. The details of the process is given by Algorithm 3.

We notice that Algorithm 3 may eliminate candidates that are very near multiples of p. For example, Algorithm 2 (GenerateCandidates) may return values like $2p$ or $4p$, which are not prime, but still may help us to find p. Using this fact, we create Algorithm 4 that factors out any 'small' prime factors. We remark that there is no need to test for primality, as the small prime factors that would cause the test to fail have been removed.

Once we have a filtered list of candidate values C of p, we need to verify that it decrypts correctly. We now develop an algorithm that tests a candidate value of p by using the decrypting function with σ ciphertexts, and checking whether the output matches the message. The outcome of each outcome results in either a success or failure. We only accept a value of p that correctly decrypts each time. Thus, the probability of having the correct p is $1 - \frac{1}{2^{\sigma}}$, where σ is the number of ciphertexts. The testing algorithm is described by Algorithm 5.

When the noise component is large (i.e. $\eta > 60$), iterating over all possible values becomes infeasible. We can only do a subset of the possible values, which are chosen at random. Hence, we can only achieve a probabilistic, instead than a deterministic, result.

3.2 Ciphertext Only Attack

In the IND-CPA attack model described above we consider sampling two ciphertexts from the encryption oracle $\mathcal{O}_{Enc}$ to mount our attack. In the real world, however, we don't have access to the encryption oracle because the scheme is defined as a secret key system. But if we examine the encryption scheme's description we find that an adversary $\mathcal{A}$ has at least one ciphertext, which is the compression argument $d = 2s + rp$. We can consider this to be an encryption of 0, since the noise component is $2s$. Hence, given one ciphertext (which is not the compression argument), we are still able to launch our attack as outlined above.

An application of a somewhat homomorphic encryption scheme that uses a single ciphertext (i.e. a single bit) would be very limited in what you could achieve. Hence it would be reasonable to consider that there are many ciphertexts available to an adversary $\mathcal{A}$, which includes at least one ciphertext as the compression modulus. If we consider n ciphertexts then there are a total of $\frac{n(n+1)}{2}$ possible pairs. We can arrange our attack in parallel and stop the first thread that finds p.

We can also use the fact that, by definition, the ciphertext is very malleable. This means we are able to modify the message by manipulating the ciphertext. Hence, if we are given a ciphertext c with an unknown message m, we can force the message to be either 0 or 1 by adding a small amount of noise to the ciphertext.

Algorithm 3 $Find_p(c_1, c_2, \lambda, \eta)$

Input: c_1, c_2 as two ciphertexts outputted by the encryption function, λ as the bit length of p and η as the initial noise size
Output: E as a set of potential values of p

$E \leftarrow \emptyset$
$C \leftarrow GenerateCandidates(c_1, c_2, \eta)$
for $c \in C$ **do**
  if $(is\_prime(c)) \wedge (2^{\lambda-1} \leq |c| \leq 2^{\lambda+1})$ **then**
    $E \leftarrow E \cup \{c\}$
  end if
end for
return E

Algorithm 4 $Find_p^{factor}(c_1, c_2, \lambda, \tau, \eta, P_{\beta})$

Input: c_1, c_2 as two ciphertexts outputted by the encryption function, λ as the bit length of p, τ as the bit range for p, η as the initial noise size and P_{β} denotes all primes less than β
Output: E as a set of potential values of p

$E \leftarrow \emptyset$
$C \leftarrow GenerateCandidates(c_1, c_2, \eta)$
for $c \in C$ **do**
  for $\alpha \in P_{\beta}$ **do**
    while $\alpha | c$ **do**
      $c = c/\alpha$
    end while
  end for
  if $(2^{\lambda-\tau} \leq |c| \leq 2^{\lambda+\tau})$ **then**
    $E \leftarrow E \cup \{c\}$
  end if
end for
return E

Algorithm 5 $Test_p(c_1, c_2, ..., c_{\sigma}, m, p)$

Input: $c_1, c_2, ..., c_{\sigma}$ as ciphertexts outputted by the encryption function, m as the message encrypted by the ciphertexts p as a candidate value to test
Output: β as the probability of correct p or $\perp$ as failure

for $i = 1 \rightarrow \sigma$ **do**
  if $D_p(c_i) \neq m$ **then**
    abort
    return $\perp$
  end if
end for
return $1 - \frac{1}{2^{\sigma}}$

4 Experimental Evaluation

We now experimentally explore the security of Brenner et al.'s symmetric somewhat homomorphic scheme. All of the experiments were programmed with Java version 7 (in a single thread model) and executed on an Intel i7-2600 3.4GHz machine with 16GB of RAM. Where possible, software from the Java standard library was used, with the GCD algorithm being one example.

4.1 Setup

In our experiment we consider the IND-CPA security model. This means we will construct an encryption function that will take a message m and output a ciphertext that encrypts m. For this encryption function we will fix $\lambda = 512$, but we will have five noise levels $\eta_1 = 8, \eta_2 = 9, \eta_3 = 10, \eta_4 = 11, \eta_5 = 12$ to show the relative performance between different noise values.

Using this setup we sample two ciphertexts c_1, c_2 from the encryption function and execute our two algorithms $Find_p$ and $Find_p^{factor}$ (Algorithms 3 and 4, respectively). For simplicity, we set τ (the range of p) for both algorithms as 1. We used a certainty value of 3 in the is_prime function in $Find_p$, which defines how many iterations of the Miller-Rabin primality test are executed. We set $\beta = 100$ for the small prime numbers to factor out.

4.2 Results

We present the results of our experiment in Table 1. We first observe as the noise gets larger, the time required for $Find_p$ and $Find_p^{factor}$ grows exponentially. This makes sense because as we must be prepared to search a larger space to find p. Since we are searching the entire noise space, then we are guaranteed to have a case where the noise component of each ciphertext is removed. What remains for each ciphertext is some multiple of p. Once this case has been identified, it is a simple matter of finding the prime p itself, where the output of the GCD algorithm may be p times a small multiple.

In our experiment we itereated over all possible noise values, both even and odd. Since we know that the noise is in fact even ($m = 0$), we can reduce the search space by half by iterating only on even noise values. We didn't use this fact in our experiment because we wanted to show the maximum time required for our approach. Even in this case, our approach is still feasible.

We also notice that there is negligible difference between the two algorithms, $Find_p$ and $Find_p^{factor}$. They seem to perform equally well, especially at the encryption scheme's proposed noise level of 8 bits.

	$Find_p$ time (s)	$Find_p^{factor}$ time (s)
η_1	4.243	4.29
η_2	16.349	16.411
η_3	65.083	64.288
η_4	260.614	259.975
η_5	1044.719	1048.559

Table 1: Performance when using 2 ciphertexts

4.3 Discussion

Obviously, increasing the noise associated with the ciphertexts greatly impacts the performance and accuracy of our method as the value of p is more difficult to uncover. Although adding noise in this way is removing the significance of the encryption scheme, which is basing new technologies on old hardness assumptions in order to reduce the size of the resulting noise component.

There have been more advanced methods for discovering p based on representing the problem as a lattice, and then solving for p. In this case of this encryption scheme, a lattice approach was unnecessary as the proposed noise is small.

5 Conclusion

In this paper we explored the security characteristics of a new symmetric somewhat homomorphic encryption scheme. The scheme is able to evaluate simple circuits, where there was a known stopping point (the ciphertext noise didn't grow too big, as to cause decryption to fail).

The scheme based its security on a well-known hardness assumption: factoring a semi-prime integer $n = pq$ where p and q are large primes. We showed theoretically and experimentally that this assumption is misleading, as we were able to break the scheme under the proposed parameters, principally using the Euclidean (GCD) algorithm. From this we designed an experiment that demonstrated the practicality of our approach. Our results suggest that we only need to break the approximate-gcd problem to find p. This assumes we have enough ciphertexts available.

The current research effort for practical fully homomorphic encryption is strong. This is mainly due to the long list of attractive applications (homomorphic spam filtering and private email). Although, fully homomorphic encryption is not at a suitable practical level for industry, somewhat homomorphic encryption seems to have good applications as (Brenner et al. 2012) demonstrated. However, we must be cautious with new technologies based on new hardness assumptions, as there may be a mathematical back door that limits their potential.

Acknowledgements

We gratefully acknowledge the valuable critical feedback provided by the reviewers of this paper. It would not have been the same without their thoughtful comments.

References

Brakerski, Z., Gentry, C. & Vaikuntanathan, V. (2012), (leveled) fully homomorphic encryption without bootstrapping, *in* 'Proceedings of the 3rd Innovations in Theoretical Computer Science Conference', ITCS '12, ACM, New York, NY, USA, pp. 309–325.

Brakerski, Z. & Vaikuntanathan, V. (2011), Efficient fully homomorphic encryption from (standard) lwe, *in* 'Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on', pp. 97–106.

Brenner, M., Perl, H. & Smith, M. (2012), Practical Applications of Homomorphic Encryption, *in* Proceedings of the International Conference on Security and Cryptography (SECRYPT 2012), Rome, Italy, pp. 5–14.

Diffie, W. & Hellman, M. (1976), 'New directions in cryptography', *Information Theory, IEEE Transactions on* **22**(6), 644–654.

ElGamal, T. (1985), 'A public key cryptosystem and a signature scheme based on discrete logarithms'.

Gentry, C. (2009), Fully homomorphic encryption using ideal lattices, *in* 'STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing', ACM, New York, NY, USA, pp. 169–178.

Gentry, C. & Halevi, S. (2011), Implementing Gentry's fully-homomorphic encryption scheme, *in* K. Paterson, ed., 'Advances in Cryptology EUROCRYPT 2011', Vol. 6632 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 129–148.

Gentry, C., Halevi, S. & Smart, N. (2012), Fully homomorphic encryption with polylog overhead, *in* D. Pointcheval & T. Johansson, eds, 'Advances in Cryptology EUROCRYPT 2012', Vol. 7237 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 465–482.

Paillier, P. (1999), Public-key cryptosystems based on composite degree residuosity classes, *in* J. Stern, ed., 'Advances in Cryptology - EUROCRYPT99', Vol. 1592 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 223–238.

Rivest, R., Adleman, L. & Dertouzos, M. (1978), On data banks and privacy homomorphisms, Academic Press, pp. 169–177.

Rivest, R. L., Shamir, A. & Adleman, L. (1978), 'A method for obtaining digital signatures and public-key cryptosystems', *Commun. ACM* **21**(2), 120–126.

Smart, N. & Vercauteren, F. (2010), Fully homomorphic encryption with relatively small key and ciphertext sizes, *in* P. Nguyen & D. Pointcheval, eds, 'Public Key Cryptography PKC 2010', Vol. 6056 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 420–443.

Stehle, D. & Steinfeld, R. (2010), Faster fully homomorphic encryption, *in* M. Abe, ed., 'Advances in Cryptology - ASIACRYPT 2010', Vol. 6477 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 377–394.

van Dijk, M., Gentry, C., Halevi, S. & Vaikuntanathan, V. (2010), Fully homomorphic encryption over the integers, *in* H. Gilbert, ed., 'Advances in Cryptology - EUROCRYPT 2010', Vol. 6110 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 24–43.

Yao, A. C. (1982), Protocols for secure computations, *in* 'Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on', pp. 160–164.