Comparison-Based Attacks Against Noise-Free Fully Homomorphic Encryption Schemes

Alessandro Barenghi, Nicholas Mainardi(✉), and Gerardo Pelosi

Department of Electronics, Information and Bioengineering – DEIB, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{alessandro.barenghi,nicholas.mainardi,gerardo.pelosi}@polimi.it

Abstract. Homomorphic Encryption provides one of the most promising means to delegate computation to the cloud while retaining data confidentiality. We present a plaintext recovery attack against fully homomorphic schemes which have a polynomial time distinguisher for a given fixed plaintext, and rely on the capability of homomorphically compare a pair of encrypted integer values. We improve by a constant factor the computational complexity of an exhaustive search strategy, which is linear in the recovered plaintext value, and show that it significantly increases the number of recoverable plaintexts. We successfully validate our attack against two noise-free fully homomorphic encryption schemes, which fulfill the mentioned requisite and were claimed to be secure against plaintext recovery attacks.

Keywords: FHE · Noise-free schemes · Plaintext recovery attack

1 Introduction

Fully Homomorphic Encryption (FHE) is a powerful cryptographic primitive, which allows to perform computation on encrypted data, retaining the correctness of the computation once the result is decrypted. The idea of FHE was first proposed by Rivest in 1978 with the name of *Privacy Homomorphisms* [20]. Designing a FHE scheme remained an open problem for three decades, during which only partially homomorphic encryption (PHE) schemes, which allow to perform only a set of operations (e.g., additions), or SomeWhat Homomorphic Encryption (SWHE) schemes, which allow to perform only a limited number of additions and multiplications, were proposed. In 2009 Gentry [10] proposed the first FHE scheme, allowing to perform an unbounded number of additions and multiplications on encrypted data. Despite the low computational efficiency, FHE has gained attention as it provides a way to outsource computation on private data to a third party such as a cloud-hosted service without revealing any information neither about the data involved in the computation nor about its result, since it is decrypted by the client alone, differently from other primitives

D. Naccache et al. (Eds.): ICICS 2018, LNCS 11149, pp. 177–191, 2018.
https://doi.org/10.1007/978-3-030-01950-1_11

such as Secure Multi-Party Computation [24] or Functional Encryption [1]. Since Gentry's seminal work, several schemes achieving better performances were proposed [6,7,9,12], as well as new techniques to speed up homomorphic computations, such as *batching* [11,21]. Nevertheless, FHE schemes still have two practical concerns to be solved before wide adoption is possible: (i) ciphertext expansion and (ii) the computational overhead imposed on homomorphic operations to preserve the correctness of the decrypted result. Indeed, the ciphertext space of SWHE/FHE schemes is consistently larger than the plaintext one, therefore even a single operation on ciphertexts is quite time consuming. The preservation of the correctness of the decrypted result needs to cope with a certain amount of randomness, typically called *noise*, that is added to the ciphertext when processing it. The amount of noise cannot be too high, lest a decryption failure occurs. Unfortunately, each homomorphic operation, especially multiplication, increases the amount of noise in the ciphertexts. Therefore, after a while, the computation must be halted (as in SWHE schemes), or a procedure to refresh the ciphertext, i.e., decrease the noise without decrypting, must be run. Such a procedure, introduced by Gentry in his original scheme [10], is called *bootstrapping* and it allows to transform a SWHE scheme, satisfying certain constraints, in a FHE one. However, this procedure is quite cumbersome, and needs to be periodically performed, slowing down the overall computation. To overcome this burden, alternative noise management techniques have been proposed, such as modulus switching [5] and scale-invariant schemes [2,3]. Acknowledging the difficulties in noise handling, some noise-free schemes have also been proposed: their ciphertexts have no noise, thus an unbounded number of operations can be performed without any costly noise management technique being involved. Nevertheless, while common noisy SWHE/FHE schemes are based on well-known and scrutinized mathematical problems, such as the Learning With Errors problem [16], noise-free schemes usually rely on less common algebraic trapdoors, which typically do not have widely scrutinized reductions to hard problems. Indeed, Liu in [15] proposed a noise-free FHE scheme, based on the *approximate greatest common divisor* problem, that was subsequently broken in [23]. Kipnis in [14] proposed a FHE scheme based on commutative rings, provably secure against ciphertext-only attacks; however knowing two plaintext-ciphertext pairs is sufficient to break the scheme [22]. Li in [13] proposed to employ non commutative rings to build FHE schemes, while Nuida [18] introduced a framework to construct FHE schemes based on group presentations obfuscated by Tietze transformations. The open challenge with schemes in [13,18] resides in the definition of a mapping between integer plaintext values and the elements of the mentioned algebraic structures, without losing neither security guarantees nor homomorphic capabilities. Lastly, Wang in [23] introduced two noise-free octonion-based FHE schemes (called `OctoM` and `JordanM`) with trapdoors based on solving quadratic modular equations (with a composite modulus) and proved their security in a ciphertext-only scenario. Due to this property, they are, to the best of our knowledge, the only noise-free FHE schemes suitable for practical usage.

While in general the homomorphic capabilities of a cryptosystem do not weaken the security guarantees per se, they may increase the adversarial power, if combined with other vulnerabilities. The advantages provided by homomorphic capabilities to the attacker were discussed in [4], focusing on the so-called *linearly decryptable* schemes, i.e., cryptosystems whose decryption function can be expressed as a *dot product* between key and ciphertext values represented in a multi-dimensional vector space. Linearly decryptable schemes usually employ a significant amount of noise to hinder Known Plaintext Attacks (KPAs). Nevertheless, in [4] the authors shown that if the scheme can homomorphically evaluate the majority function, then a KPA becomes practically viable. Moreover, in [23] the authors introduced, for linearly decryptable schemes, an algorithm to determine if the plaintext corresponding to a given ciphertext is equal to the integer value 1. We remark that noise free `OctoM` and `JordanM` FHE schemes are linearly decryptable, and thus affected by the aforementioned issues.

Contributions. We present a plaintext recovery attack, against FHE schemes having plaintexts in $\mathbb{Z}_n$, with $n > 2$, and where it is possible to devise an efficient algorithm able to determine if a generic ciphertext under a given key k is the encryption of a fixed plaintext m, which we denote as *m-distinguisher*. Although, to the best of our knowledge, such a distinguisher has been proposed for linearly decryptable schemes only, our attack will be applicable to any FHE scheme for which such a distinguisher can be found. Our attack, which is performed in a ciphertext-only scenario, leverages the capability to homomorphically compare two encrypted integer values, obtaining a computational complexity which is linear in the plaintext integer value being recovered and improving over an exhaustive search strategy by a significant constant factor. We successfully validate the proposed attack against two linearly decryptable noise free octonion-based cryptosystems [23] (`OctoM` and `JordanM`), which were claimed to be computationally secure in a ciphertext-only attack scenario. Furthermore, we apply our attack to retrieve enough plaintexts from the said schemes so that mounting a KPA to recover the key becomes viable.

2 Preliminaries

Definition 1 (Negligible Function). *A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is negligible if, for every univariate positive polynomial, $poly(x) \in \mathbb{R}[x]$, there exists an integer $c > 0$ such that $\forall\, x > c$, $|\epsilon(x)| \leq \frac{1}{poly(x)}$.*

Definition 2 (Indicator Function). *Given a set S and a subset $A \subseteq S$, the indicator function of the elements of A over the ones included in S is defined as: $\mathbb{1}_A : S \to \{0, 1\}$, where $\mathbb{1}_A(x) = 1$ if $x \in A$, 0 otherwise.*

2.1 Homomorphic Encryption Algorithms

Our definition of Fully Homomorphic Encryption follows [7], without constraining the encryption function to deal with a single bit at time.

An homomorphic encryption (HE) scheme specifies three sets: $\mathcal{M}$, $\mathcal{C}$ and $\mathcal{F}$. The set of plaintexts $\mathcal{M}$ usually coincides with the set of integer values ranging from 0 to $n-1$, with $n > 2$, assumed to be the representatives of the residue classes modulo n, i.e., $\mathbb{Z}_n \equiv \mathbb{Z}/n\mathbb{Z}$. The ciphertext space $\mathcal{C}$ includes elements with an algebraic representation that depends on the specific HE scheme at hand. The set of polynomials $\mathcal{F} \subseteq \mathbb{Z}_n[x_1, x_2, \ldots, x_a]$, with $a \geq 1$ and degree greater or equal to zero, defines the functions that the HE scheme at hand allows to be evaluated. That is, each of these polynomials computes a function $f : \mathcal{M}^a \to \mathcal{M}, a \geq 1$ over the plaintexts, and is also referred to as an *arithmetic circuit* composed by gates performing multiplications and additions in $\mathbb{Z}_n$. We provide the definition of an HE scheme starting from an asymmetric HE scheme, and describe a symmetric one by difference.

Definition 3 (Public-key Homomorphic Encryption Scheme). *A public-key Homomorphic Encryption scheme is defined as a tuple of four polynomial time algorithms* $\langle \texttt{KeyGen}, \texttt{Enc}, \texttt{Dec}, \texttt{Eval} \rangle$*:*

- ***Key Generation.*** $\langle sk, pk, evk \rangle \leftarrow \texttt{KeyGen}(1^\lambda)$ *is a probabilistic algorithm that, given the security parameter* λ*, generates the secret key* sk*, the public key* pk *and the public evaluation key* evk*.*
- ***Encryption.*** $c \leftarrow \texttt{Enc}(pk, m)$ *is a probabilistic algorithm that, given a message* $m \in \mathcal{M}$ *and the public key* pk*, computes a ciphertext* $c \in \mathcal{C}$
- ***Decryption.*** $m \leftarrow \texttt{Dec}(sk, c)$ *is a deterministic algorithm that, given a ciphertext* $c \in \mathcal{C}$ *and the secret key* sk*, outputs a message* $m \in \mathcal{M}$
- ***Evaluation.*** $c \leftarrow \texttt{Eval}(evk, f, c_1, c_2, \ldots, c_a)$ *is a probabilistic algorithm computing a ciphertext* $c \in \mathcal{C}$*, using an arithmetic circuit* $f \in \mathcal{F}$ *with* $a \geq 1$ *inputs, the ciphertexts* $c_1, c_2, \ldots, c_a$*, and the evaluation key.*

The following properties must hold:

- ***Decryption Correctness.*** $\forall\, m \in \mathcal{M} : \texttt{Dec}\,(sk, \texttt{Enc}(pk, m)) = m.$
- ***Evaluation Correctness.*** $\forall f \in \mathcal{F}, m_1, \ldots, m_a \in \mathcal{M}$*:*
  $\Pr\,(\,\texttt{Dec}(sk, \texttt{Eval}(evk, f, c_1, \ldots, c_a)) = f(m_1, \ldots, m_a)\,) = 1 - \epsilon(\lambda),$
  where $c_1 = \texttt{Enc}(pk, m_1) \wedge \cdots \wedge c_a = \texttt{Enc}(pk, m_a)$ *and* $\epsilon(\lambda)$ *is a negligible function of the security parameter of the scheme.*
- ***Compactness.*** $\forall\, f \in \mathcal{F}$*,* $c_1, \ldots, c_k \in \mathcal{C}$*:*
  $|\texttt{Eval}(evk, f, c_1, \ldots, c_k)| \leq poly(\lambda)$*, where* $|\cdot|$ *denotes the bit length of a ciphertext, while* $poly(\cdot)$ *denotes a positive univariate polynomial.*

The requirement on the evaluation correctness trivially states that by decrypting the output of the `Eval` algorithm we obtain the correct result of the computation homomorphically performed by `Eval` on the ciphertexts. In particular, the `Eval` algorithm evaluates a polynomial, defined over the plaintext space, in the sequence of input ciphertexts by replacing the modular additions and multiplications with the homomorphic operations `Add` and `Mul`, respectively, that are, in

turn, two probabilistic polynomial time algorithms defined over the ciphertext space $\mathcal{C}$:

- **Homomorphic Addition.** $c \leftarrow \texttt{Add}(evk, c_1, c_2)$ computes a ciphertext $c \in \mathcal{C}$ such that $\texttt{Dec}(sk, c) = \texttt{Dec}(sk, c_1) + \texttt{Dec}(sk, c_2)$
- **Homomorphic Multiplication.** $c \leftarrow \texttt{Mul}(evk, c_1, c_2)$ computes a ciphertext $c \in \mathcal{C}$ such that $\texttt{Dec}(sk, c) = \texttt{Dec}(sk, c_1) \cdot \texttt{Dec}(sk, c_2)$.

When defining a symmetric-key homomorphic encryption scheme, the only difference is the key generation algorithm $\texttt{KeyGen}(1^\lambda)$ outputting a tuple $\texttt{k} = \langle sk, pk, evk \rangle$ with $sk = pk$. Lastly, We recall the categorization of HE schemes depending on the specific choice of the set of functions $\mathcal{F}$ which can be evaluated. Specifically, a PHE scheme exhibits a function $f \in \mathcal{F}$ defined via an arithmetic circuit including a single type of gate (an additive one or a multiplicative one). A SWHE scheme exhibits a function $f \in \mathcal{F}$ defined via an arithmetic circuit with a depth no higher than a fixed (scheme-dependent) threshold. Finally, a FHE scheme exhibits a function $f \in \mathcal{F}$ defined via an unconstrained arithmetic circuit.

2.2 Homomorphic Comparisons

One of the requirement to apply our attack is the existence of an algorithm able to determine if a generic ciphertext is an encryption of a fixed plaintext m. Therefore, we now provide a formal definition for this algorithm, which we refer to as m-distinguisher.

Definition 4 (m-distinguisher). *Let $\langle \texttt{KeyGen}, \texttt{Enc}, \texttt{Dec}, \texttt{Eval} \rangle$ be a homomorphic encryption scheme with security margin λ, and let $\mathcal{M}$, $\mathcal{C}$ be the plaintext and ciphertext spaces, related by the generated key $\texttt{k} = \langle sk, pk, evk \rangle$. Let $A^m_{\texttt{k}} \subset \mathcal{C}$, be the set of ciphertexts corresponding to the encryption of a plaintext $m \in \mathcal{M}$, i.e.: $A^m_{\texttt{k}} = \{c \in \mathcal{C} \text{ s.t. } \texttt{Dec}(sk, c) = m\}$.*

Given a plaintext $m \in \mathcal{M}$, an m-distinguisher is a deterministic polynomial time algorithm $\texttt{A}_m$ taking as input a ciphertext $c \in \mathcal{C}$ and the public portion of $\texttt{k}$ (i.e., $\texttt{k}_{\texttt{pub}} = \langle pk, evk \rangle$ for public-key systems and $\texttt{k}_{\texttt{pub}} = \langle evk \rangle$ for symmetric ones), and computes the indicator function of the elements of $A^m_{\texttt{k}}$ over the set of ciphertexts, $\mathbb{1}_{A^m_{\texttt{k}}} : \mathcal{C} \rightarrow \{0, 1\}$, in such a way that

$$\frac{|\{c \in \mathcal{C} \text{ s.t. } \texttt{A}_m(c, \texttt{k}_{\texttt{pub}}) = \mathbb{1}_{A^m_{\texttt{k}}}(c)\}|}{|\mathcal{C}|} \geq 1 - \epsilon(\lambda),$$

where $\epsilon(\lambda)$ is a negligible function of the security margin of the system.

Given the existence of this m-distinguisher, our attack leverages the capability to homomorphically compare two encrypted integers. Therefore, we now present the main methods proposed in the literature to compute this functionality, including the one used in our attack. First of all, performing comparisons requires to homomorphically evaluate the *greater-than* function on a chosen integer interval.

Definition 5 (Greater-than Function). *Given a positive integer b and an interval of integers $D_t = \{0, 1, \ldots, t-1\}$, with $t \geq 2$, the greater-than function $GT_{t,b} : D_t \times D_t \to \{b-1, b\}$ is defined as:*

$$GT_{t,b}(x, y) = \begin{cases} b & \text{if } x \geq y, \\ b-1 & \text{otherwise} \end{cases}$$

To the extent of evaluating this function with an HE scheme, we need to find a polynomial $f_{gt} \in \mathcal{F} \subseteq \mathbb{Z}_n[x, y]$, such that $f_{gt}(x, y) = GT_{t,b}(x, y)$, with $2 \leq t \leq n$, $1 \leq b < n$, and x, y being the representatives of residue classes modulo n, (i.e., $x, y \in \mathbb{Z}_n$) considered as integers less than t. Such a polynomial can be easily found if the plaintext space is $\mathbb{Z}_2$: indeed, additions and multiplications become `xor` and `and` gates, while the input variables are the single-bit values in the binary encodings of x and y, and thus there are many logical circuits computing $GT_{t,b}(\cdot, \cdot)$ function.

Considering a plaintext ring $\mathcal{M} = \mathbb{Z}_n$, with $n > 2$, which is the case targeted in our work, finding an efficiently computable polynomial for the $GT_{t,b}(\cdot, \cdot)$ function is a challenging task. Çetin in [8] reports two methods to compute the $GT_{t,b}(\cdot, \cdot)$ function which do not require interaction between the secret key owner and the party who performs homomorphic evaluation. However, both of these methods are not suitable for our attack: indeed, the first one is not applicable to a composite module n; the second method computes an approximation of the $GT_{t,b}(\cdot, \cdot)$, while our attack needs an exact computation of this function.

A more effective solution is proposed in [17]: the greater-than function is computed as $GT_{t,b}(x, y) = SIGN_{t,b}(x - y)$, where $SIGN_{t,b}(z)$ is a function defined over $\overline{D_t} \subseteq \mathbb{Z} = \{-t+1, \ldots, 0, \ldots, t-1\}$ such that $SIGN_{t,b}(z) = b$ if $z \geq 0$, $b-1$ otherwise. The homomorphic evaluation of the function $SIGN_{t,b}(\cdot)$ requires a polynomial $f_{sign} \in \mathcal{F} \subseteq \mathbb{Z}_n[z]$ fulfilling $f_{sign}(z \bmod n) = SIGN_{t,b}(z)$, with $2 \leq t \leq \frac{n}{2}$, $1 \leq b < n$ and $z \in \overline{D_t}$. In [17], the polynomial f_{sign} is computed applying the Lagrange interpolation formula to $2t-1$ points having coordinates $(\, z,\, SIGN_{t,b}(z)\,)$, with $z \in \overline{D_t}$, and considering a prime modulus, i.e., $n = p$.

As we are considering as a plaintext space the ring $\mathbb{Z}_n$ with a generic modulus $n > 2$, we introduce an additional constraint on the integer t, formalized in Lemma 1, to extend the applicability of the aforementioned method to a generic ring $\mathbb{Z}_n$:

Lemma 1. *Given an integer $t \geq 2$, and a set $\overline{D_t} = \{-t+1, \ldots, 0, \ldots, t-1\}$, the polynomial $f(z) \in \mathbb{Z}_n[z]$, with $n > 2$, interpolating $2t-1$ points having the* x*-coordinate ranging over all values in $z \in \overline{D_t}$ exists if $t \leq \frac{q}{2}$, where q is the smallest prime factor of n.*

Proof. Considering $2t-1$ points $\{(x_1, y_1), \ldots, (x_{2t-1}, y_{2t-1})\}$ in $\mathbb{Z}_n \times \mathbb{Z}_n$, the interpolating polynomial $f \in \mathbb{Z}_n[x]$, with degree at most $2t-2$, can be computed by the Lagrange interpolation formula:

$$f(x) = \sum_{i=1}^{2t-1} y_i \prod_{j=1, j \neq i}^{2t-1} (x - x_j)(x_i - x_j)^{-1}$$

The existence of the multiplicative inverses (in $\mathbb{Z}_n$) required in this formula is ensured if all the values $x_i - x_j$ are co-prime with n. Assuming the x-coordinates to be mutually distinct and in $\overline{D_t}$, the constraint $t \leq \frac{q}{2}$ implies that $-q < -2t + 2 \leq x_i - x_j \leq 2t - 2 < q$. Since q is the smallest prime factor of n, then all the elements in $\mathbb{Z}_n \setminus \{0\} \cap \{-q+1, \ldots, q-1\}$ are co-prime with n, therefore all the values $x_i - x_j$ are co-prime with n, and thus invertible, allowing $f(x)$ to be interpolated by the Lagrange formula. □

In conclusion, by Lagrange interpolation we can obtain a polynomial $f_{sign} \in \mathcal{F} \subseteq \mathbb{Z}_n[z]$ which computes the function $SIGN_{t,b}(z), \forall z \in \overline{D_t}$, and then a polynomial $f_{gt} \in \mathcal{F} \subseteq \mathbb{Z}_n[x, y]$, computing the function $GT_{t,b}(x, y), \forall x, y \in D_t$, as $f_{gt}(x, y) = f_{sign}(x - y)$. Since $f_{gt} \in \mathcal{F}$, it can be homomorphically evaluated by the `Eval` algorithm of the HE scheme, by replacing addition and multiplications of the polynomial with corresponding homomorphic operations (`Add` and `Mul`) whose inputs are ciphertexts in $\mathcal{C}$. In the following, we denote the algorithm $\texttt{Eval}(evk, c_1, c_2, f_{gt})$ by $HGT_{t,b}(c_1, c_2)$, which takes as input two ciphertexts with corresponding plaintexts m_1,$m_2 \in D_t$, and outputs an encryption of $GT_{t,b}(m_1, m_2)$. In particular, since $GT_{t,b}$ is defined on the interval $D_t = \{0, \ldots, t-1\}$, $t \leq \frac{q}{2}$, with q being the smallest prime factor of n, then $c_1, c_2 \in \mathcal{C}_t = \{c \in \mathcal{C} \text{ s.t. } \texttt{Dec}(sk, c) < t\}$ is a sufficient condition for $\texttt{Dec}(sk, HGT_{t,b}(c_1, c_2)) = GT_{t,b}(m_1, m_2)$. The computational complexity required to interpolate $2t - 1$ points by applying the Lagrange formula is $\mathrm{O}(t^2)$ operations in $\mathbb{Z}_n$; while the evaluation of the polynomial $f_{sign} \in \mathbb{Z}_n[z]$, whose degree is at most $2t - 2$, has a computational complexity $\mathrm{O}(t)$. From this, it is easy to note that the computational cost of the $HGT_{t,b}(\cdot, \cdot)$ algorithm is $\mathrm{O}(t)$. We note that, while there are no current algorithms to compute $HGT_{t,b}(\cdot, \cdot)$ in less than $\mathrm{O}(t)$, research efforts driven by the usefulness of a homomorphic comparison may lead to an improvement in this sense. Since our methodology relies on the computation of $HGT_{t,b}(\cdot, \cdot)$ as an atomic component, such improvements will positively affect the efficiency of our attack.

3 Attack Strategy

In the following we detail a plaintext recovery attack which takes as input a ciphertext and the publicly available evaluation key evk of the HE scheme at hand (which can be either a FHE, or a SWHE capable of computing $HGT_{t,b}$). Since a FHE scheme must allow the evaluation of arbitrary polynomials, then it must provide to the evaluator a method to obtain encryptions of known values, preferably avoiding interaction with the key owner. In case the HE scheme is an asymmetric one, such ciphertexts can be directly obtained employing the public key encryption algorithm, while for a symmetric scheme, the encryption of any value can be obtained from a single encryption of $\hat{m} = 1$ leveraging homomorphic operations (we can obtain encryptions of all powers of 2 by iteratively summing $\hat{m}$ by itself, and then compute the encryption of any integer value leveraging its binary representation). Thus, we assume, in case of a symmetric FHE scheme, that an encryption of $\hat{m} = 1$ is embodied in the evaluation key evk to allow the

computation of encryptions of known values by the evaluator. Such encryption of $\hat{m}$ can be used also by the attacker to obtain the ciphertexts required to perform the attack.

Comparison-Based Attack. The core idea is to perform a homomorphic binary search over the possible candidates for the value of the plaintext corresponding to the ciphertext at hand. To this end, a comparison function CMP, taking two ciphertexts as inputs and yielding an outcome in cleartext, is computed leveraging the homomorphic greater-than function $HGT_{t,b}$ (see Sect. 2). Since the result of the $HGT_{t,b}$ function is still encrypted, the m-distinguisher is employed to determine its actual (plaintext) value. The attacker can compute the comparison function CMP employing the aforementioned strategy as follows:

Definition 6 (Comparison Function). *Given the ciphertexts $c_1, c_2 \in \mathcal{C}_t$ and $\mathtt{A}_m$ the algorithm computing the m-distinguisher, where m is a fixed plaintext value, the function $CMP : \mathcal{C}_t \times \mathcal{C}_t \to \{1, 0, -1\}$ is computed as:*

$$CMP(c_1, c_2) = \begin{cases} 1 & \text{if } v_1 = 1 \wedge v_2 \neq 1, \\ 0 & \text{if } v_1 = 1 \wedge v_2 = 1, \\ -1 & \text{otherwise} \end{cases}$$

with $v_1 = \mathtt{A}_m(HGT_{t,m}(c_1, c_2), \mathtt{k_{pub}})$, $v_2 = \mathtt{A}_m(c_1 - c_2 + c_m, \mathtt{k_{pub}})$ and c_m being an encryption of m computed by the attacker.

Denoting with T_d the computational complexity of the m-distinguisher, we have that the time complexity of CMP is $T_{CMP} = \mathrm{O}(t + 2T_d)$ as its execution involves at most two computations of the m-distinguisher plus one computation of the $HGT_{t,m}$ function, which has complexity $\mathrm{O}(t)$. Leveraging the function CMP, the binary search strategy locates the value of the actual plaintext in the range D_t, which is t elements wide, with a computational cost of $\mathrm{O}(T_{CMP} \cdot \log(t)) = \mathrm{O}((t + 2T_d)\log(t))$.

Starting from the strategy which has just been described, we improve its effectiveness extending the range of the recoverable plaintexts. To this end, we split the set of recoverable plaintexts into $|D_t| = t$ sized chunks, find into which chunk the plaintext is likely to be contained, and proceed to retrieve it employing the binary search approach. We denote with D_s the set of recoverable plaintexts ($D_s = \{0, 1, \ldots, s-1\}$, $s \leq n$), and with $\mathcal{C}_s$ the set of ciphertexts obtained encrypting plaintexts in D_s, i.e.: $\mathcal{C}_s = \{c \in \mathcal{C} \text{ s.t. } \mathtt{Dec}(sk, c) < s\}$. The recoverable message space D_s is split into σ chunks containing numerically consecutive plaintexts, with $\sigma = \lceil \frac{s}{t} \rceil$: for instance, the i-th chunk contains plaintexts values $\{(i-1)t, \ldots, it-1\}$.

Algorithm 1 shows how our improved attack is performed. It iterates over all the σ chunks, testing, for each one of it, if the plaintext m_c, corresponding to the input ciphertext c, may be contained in the chunk being scanned (lines 2–9). To this end, the algorithm starts by testing if m_c may be in a chunk $\{(i-1)t, \ldots, it-1\}$ by verifying if $GT_{t,m}(m_c, (i-1)t) = m$ (lines 3–4). In case

Algorithm 1. Plaintext Recovery Attack

Input: ciphertext $c \in \mathcal{C}_s$, $\mathcal{C}_s = \{c \in \mathcal{C} \text{ s.t. } \texttt{Dec}(sk, c) < s\}$
Output: plaintext $m_c = \texttt{Dec}(sk, c)$, $m_c \in \mathbb{Z}_n$

1 **begin**
2 **for** $i \leftarrow 1$ **to** σ **do**
3 $c_{gt} \leftarrow HGT_{t,m}(c, \texttt{Enc}(pk, (i-1)t))$
4 **if** $\texttt{A}_{(m)}(c_{gt}, \texttt{k}_{\texttt{pub}}) = 1$ **then**
5 $c_{gt} \leftarrow HGT_{t,m}(c, \texttt{Enc}(pk, it-1)) + \texttt{Enc}(pk, 1)$
6 **if** $\texttt{A}_{(m)}(c_{gt}, \texttt{k}_{\texttt{pub}}) = 1$ **then**
7 $m_c \leftarrow \texttt{BinarySearch}(c - \texttt{Enc}(pk, (i-1)t))$
8 **if** $m_c \neq \perp$ **then**
9 **return** $m_c + (i-1)t$

this test succeeds (line 4, case of the `if` being taken), Algorithm 1 proceeds to test also if m_c is smaller than the upper bound $it-1$ of the chunk at hand, by verifying that $GT_{t,m}(m_c, it-1) = m-1$ with an analogous approach (lines 5–6). If the tests at lines 3 – 6 succeed, then the current chunk may contain the plaintext m_c, and so Algorithm 1 attempts a plaintext recovery employing the binary search approach described in precedence over the current chunk (line 7). We note that the answer of these tests are subject to potential false positives. Indeed, if $m_c \notin \{(i-1)t, \ldots, it-1\}$, then $m_c - (i-1)t \notin \overline{D_t}$ or $m_c - (it-1) \notin \overline{D_t}$: thus, it means that the polynomial $f_{sign}(z) \in \mathbb{Z}_n[z]$, obtained by interpolating points whose x-coordinates range over $\overline{D_t}$, is evaluated on a point $z \notin \overline{D_t}$, hence yielding an outcome which is either outside the set $\{m-1, m\}$ or (by coincidence) inside it. Therefore, it may happen that $f_{gt}(m_c, (i-1)t) = f_{sign}(m_c - (i-1)t) = m$ and $f_{gt}(m_c, it-1) = f_{sign}(m_c - (it-1)) = m-1$ even if $m_c \notin \{(i-1)t, \ldots, it-1\}$. In this case, the interval $\{(i-1)t, \ldots, it-1\}$ is identified as a false positive. However, these false positive are filtered out later in the algorithm. Indeed, since the binary search is effective only under the assumption that the sought plaintext is in D_t, Algorithm 1 (line 7) exploits the homomorphic operations to subtract the value of the lower bound of the current chunk from m_c, working on its corresponding ciphertext c, to retrieve the value of $m_c \bmod t$, provided that the chunk detection was not reporting a false positive. If a result is returned (line 8), the actual value of m_c is reconstructed adding back the lower bound of the current chunk to the value retrieved by the binary search (line 9), otherwise the current chunk is a false positive.

We now consider the time complexity of Algorithm 1 as a function of the value of the plaintext to be retrieved m_c. Algorithm 1 is expected to perform $\lceil \frac{m_c}{t} \rceil$ iterations of the outer loop. Each one of the iterations, save for the last one, will fail the membership tests with very high probability (false positives are unlikely), thus resulting in a computational effort which is $\mathrm{O}(t + T_d)$ at each iteration. However, we now consider the overall worst-case complexity $T_a(m_c)$ of the improved plaintext recovery attack:

$$\mathrm{O}\left(\lceil \frac{m_c}{t} \rceil (t + T_d + T_{\texttt{BinarySearch}})\right) = \mathrm{O}\left(\log(t)(m_c + \lceil \frac{m_c}{t} \rceil T_d)\right) \tag{1}$$

Therefore, our attack has a linear time complexity, which is the main reason why it is able to practically recover only ciphertexts whose corresponding plaintext is not too big. However, by setting $t = 2^{20}$ (an upper bound imposed by the $O(t^2)$ computational cost of Lagrange interpolation), we see that, unless $T_d > 2^{23}$, recovering plaintexts as big as 2^{32} still retains a computational complexity $T_a(m_c) < 2^{40}$. Since many typical FHE scenarios involve computations on relatively small values (e.g. power consumption statistics from smart meters), we deem this plaintext recover capability effective enough to be worth considering.

To conclude the description of our attack, we now show the speed-up obtained by Algorithm 1 over an exhaustive search strategy leveraging only the m-distinguisher. This latter attack tries all plaintext values $x \in \mathbb{Z}_n$ in increasing order, with the recovered plaintext being the first x such that $\mathtt{A}_m(\mathtt{Enc}(pk, x) - c + \mathtt{Enc}(pk, m)) = 1$. Denoting the value of the recovered plaintext as m_c, with this strategy we employ the m-distinguisher m_c times, therefore the complexity of this approach is $O(m_c T_d)$. The speed-up of our attack over this simple strategy can be computed as follows:

$$\frac{m_c T_d}{T_a(m_c)} = \frac{m_c T_d}{\log(t)(m_c + \lceil \frac{m_c}{t} \rceil T_d)} = \frac{m_c t T_d}{\log(t)(m_c t + m_c T_d)} = \frac{t T_d}{\log(t)(t + T_d)}$$

This calculation shows that our attack improves the exhaustive search strategy by a constant factor, thus without changing its asymptotic complexity. In particular, the speed-up depends on the values of t, chosen by the attacker, and T_d, given by the target scheme. Although this improvement may seem negligible, we will show, for the FHE schemes targeted by our attack, that the magnitude of the speed-up may be significant in practice, as it largely increases the number of recoverable plaintexts.

4 Two Case Studies

In this section, we evaluate our attack against two symmetric noise free FHE schemes [23], `OctoM` and `JordanM`. Although there is an efficient 1-distinguisher for these schemes, they were claimed to be secure against ciphertext-only adversaries aiming to recover either the plaintext or the secret key [23], making them a proper target for our attack.

4.1 Target Fully Homomorphic Encryption Schemes

We report a description of the two target symmetric FHE schemes, `OctoM`[1] and `JordanM`, focusing only on the details which are relevant for our attack: the characterization of the ciphertext space and the description of homomorphic operations. We refer the reader to [23] for further details. The plaintext space

[1] We find out that, to make `OctoM` multiplicatively homomorphic, some additional constraints are needed: they will be shown in the full version of the paper.

is the ring of integers $\mathbb{Z}_n$, with a composite $n > 2$, for both schemes. The ciphertext space of `OctoM` is the set of 8×8 matrices with entries in $\mathbb{Z}_n$, i.e., $\mathbb{Z}_n^{8\times 8}$, while the one of `JordanM` is the set of 3×3 matrices with entries in $\mathbb{O}(\mathbb{Z}_n)$, where $\mathbb{O}(\mathbb{Z}_n)$ is the non commutative, non associative algebra of octonions whose support is the vector space $\mathbb{Z}_n^8$. Both schemes perform a single matrix addition to homomorphically add two ciphertexts, while to homomorphically multiply two ciphertexts C_1, C_2 the two schemes employ different procedures:

- `OctoM` **Homomorphic Multiplication.** $C_{mul} = C_2 \cdot C_1 \cdot C_{-1}$, where $\cdot$ denotes the matrix dot product and $C_{-1} \in \mathbb{Z}_n^{8\times 8}$ is an encryption of the plaintext value $n-1$, embodied in the evaluation key, evk.
- `JordanM` **Homomorphic Multiplication.** $C_{mul} = C_1 \star C_2$, where $\star$ denotes the Jordan product.

4.2 Security Analysis

As already acknowledged in [23], the target FHE schemes are linearly decryptable, that is their decryption function can be expressed as a dot product between the key and the ciphertext represented in a d-dimensional vector space defined over the ring $\mathbb{Z}_n$. For the two target FHE schemes, the ciphertext space dimension d is 64 for `OctoM`, since a ciphertext is an 8×8 matrix, while $d = 9 \cdot 8 = 72$ for `JordanM`, since the ciphertext matrix is a 3×3 matrix whose entries are elements of $\mathbb{O}(\mathbb{Z}_n)$. Linearly decryptable schemes are vulnerable to KPAs: if the attacker has approximately d plaintexts/ciphertexts pairs, then a linear system of equations can be built to recover the key and decrypt any ciphertext. In addition, an efficient 1-distinguisher for any linearly decryptable scheme is proposed in [23]. We now describe the construction of this distinguisher, since we leverage it to perform our attack. Given a ciphertext C, represented as a d dimensional vector, consider the first $d+1$ powers of C. Since the ciphertext space dimension is d, then these $d+1$ ciphertexts are linearly dependent. Therefore, by definition, there are non trivial solutions to the system of d equations with $d+1$ unknowns a_i defined as $\sum_{i=1}^{d+1} a_i C^i = 0$. Since the decryption function is linear and the encryption scheme is multiplicatively homomorphic, the following equality also holds: $\sum_{i=1}^{d+1} a_i m^i = 0$, where $m = \texttt{Dec}(sk, C)$. If $m = 1$, this equation becomes $\sum_{i=1}^{d+1} a_i 1^i = 0 \Rightarrow \sum_{i=1}^{d+1} a_i = 0$. Therefore, if the additional constraint $\sum_{i=1}^{d+1} a_i \neq 0$ is added to the system of equations $\sum_{i=1}^{d+1} a_i C^i = 0$, a solution is found if and only if $m \neq 1$. In conclusion, by looking at the solution of this system, we can determine if $m = 1$ or not. The computational complexity of this 1-distinguisher is $O(d^3)$, since solving a system of equations has cubic complexity. We remark that the system can be solved directly on ciphertexts, no knowledge about the plaintexts or the key is required. Therefore, this distinguisher is a particular case of Definition 4, since it does not employ $\texttt{k}_{\texttt{pub}}$, the publicly available portion of the cipher key. While the existence of these vulnerabilities (1-distinguisher and KPA) is acknowledged by designers of `OctoM` and `JordanM` too, their security analysis claims [23, Theorem 7] that the hardness of solving quadratic equation systems in $\mathbb{Z}_n$ (with a composite n) guarantees that

no information about plaintexts can be inferred from ciphertexts. The proof of this claim is based on two reductions: first, the problem of finding the secret key is reduced to the problem of solving a system of multivariate quadratic equations in $\mathbb{Z}_n$, then the problem of recovering a plaintext is reduced to the problem of solving a univariate quadratic equation in $\mathbb{Z}_n$. These reductions state that solving quadratic equations in $\mathbb{Z}_n$ is sufficient to break the cryptosystems, but they do not state that recovering the secret key or a plaintext is as hard as solving quadratic equations in $\mathbb{Z}_n$. Thus, there is no contradiction between the existence of our attack and the hardness of solving quadratic equations in $\mathbb{Z}_n$ (which is as hard as factoring n).

4.3 Breaking Target FHE Schemes with Our Attack

As already discussed in Sect. 3, since a FHE scheme must allow the evaluation of arbitrary polynomials, then it must provide to the evaluator a method to obtain encryptions of known values, preferably avoiding interaction with the key owner. Since no method to provide this capability was proposed for the considered FHE schemes, we assume that an encryption of 1 is embodied in the evaluation key to provide this capability to users of the FHE scheme. It is worth noting that an encryption of 1 is not necessary for the `OctoM` scheme, since it can be computed by squaring the encryption of -1 already provided in the evaluation key. In the instantiation of our attack against `JordanM`, there is a relevant issue related to the fact that, as outlined in Definition 4, the m-distinguisher may have a wrong outcome on a negligible portion of the ciphertexts. However, this portion is not negligible for several ciphertexts being used in our attack. The problem arises because of two random values which are employed to randomize the encryption. We denote these two values for a ciphertext C by r_C, s_C. In particular, we find out two relevant facts about these values[2]: first, if $s_C = 1 \vee r_C = 1$, then the 1-distinguisher will classify the ciphertext as an encryption of 1 independently from the plaintext value m; secondly, the evaluation correctness property (see Definition 3) holds not only for the message m, but for r_C, s_C too. The latter fact basically means that, given two ciphertexts C_1, C_2, encrypted with random values respectively r_{C_1}, s_{C_1} and r_{C_2}, s_{C_2}, these two properties, related to homomorphic operations `Add` and `Mul` of `JordanM` scheme, hold:

$$C = \texttt{Add}(C_1, C_2) \rightarrow r_C = r_{C_1} + r_{C_2} \wedge s_C = s_{C_1} + s_{C_2}$$
$$C = \texttt{Mul}(C_1, C_2) \rightarrow r_C = r_{C_1} \cdot r_{C_2} \wedge s_C = s_{C_1} \cdot s_{C_2}$$

As a consequence, a ciphertext C_{gt} obtained through homomorphic evaluation of $GT_{t,1}$ function has only four possible assignments to its random values $r_{C_{gt}}, s_{C_{gt}}$, which are $\{(0,0),(0,1),(1,0),(1,1)\}$, since the image of $GT_{t,1}$ is $\{0,1\}$. Therefore, while for a generic ciphertext C, $Pr(r_C = 1 \vee s_C = 1)$ is negligible, and thus this issue is not relevant for the reliability of the 1-distinguisher in general, for

[2] Proofs are omitted for the sake of brevity. They will be included in the full version of the paper as long as a fully detailed description of `JordanM` and `OctoM` cryptosystems.

a generic ciphertext C_{gt} obtained through homomorphic evaluation of $GT_{t,1}$ the same probability is 0.75, hence the outcome of the 1-distinguisher, when its input is a ciphertext C_{gt} as in our attack, is likely to be erroneous.

To overcome this issue, we devise a ciphertext refreshing procedure, which employs the available encryption of 1 to compute a new ciphertext C' as $C+C-\texttt{Enc}(pk,1)*C$, having the same plaintext m, but random values $r_{C'}, s_{C'}$ which are highly likely to be different from 1, since they depend on the random values chosen for the encryption of 1. Therefore, in our attack we employ a slightly tailored version of the distinguisher, whose output is equal to $\texttt{A}_1(C)\cdot\texttt{A}_1(C')$. By using this distinguisher, we can perform our attack on both the target FHE schemes to recover plaintexts. Then, after d plaintexts have been recovered, we can perform the KPA and recover the key, breaking the schemes.

We can now estimate the computational complexity of our attack for the target FHE schemes. For linearly decryptable schemes, T_d, the computational complexity of the 1-distinguisher is $\mathrm{O}(d^3)$, with $d = 64$ for `OctoM` and $d = 72$ for `JordanM`, which means that $T_d = \mathrm{O}(2^{19})$ for both schemes. However, the distinguisher is always invoked twice in our attack to increase its reliability, therefore the computational complexity we are going to use in place of T_d, in the formulae derived in Sect. 3 to estimate the computational effort of our attack, is $T'_d = 2T_d = \mathrm{O}(2^{20})$. Given this estimation, we can see that it is practical to recover plaintext values as big as 2^{32}, which is expected to be enough for a significant number of ciphertexts in FHE applications. The computational effort required to recover a plaintext value $m_c = 2^{32}$, can be computed as follows (see Eq. 1 in Sect. 3), replacing T_d with $T'_d = 2^{20}$ and setting $t = 2^{20}$:

$$T_a(2^{32}) \leq 2^{32}\log(2^{20}) + \left\lceil \frac{2^{32}}{2^{20}} \right\rceil 2^{20}\log(2^{20}) = 2^{32}\cdot 20 + 2^{32}\cdot 20 \leq 2^{38}$$

Conversely, recovering a plaintext as big as $m_c = 2^{32}$ via an exhaustive search strategy has a computational cost of $\mathrm{O}(m_c T_d) = 2^{32}\cdot 2^{19} = 2^{51}$ (note that with this strategy we do not need to invoke twice the 1-distinguisher, thus we can use T_d instead of T'_d). Indeed, the speed-up of our attack is: $\frac{tT_d}{\log(t)(t+T'_d)} \geq \frac{2^{39}}{2^5\cdot 2^{21}} = 2^{13}$. This result shows that the improvement of our attack is not negligible: considering a computational effort fixed a-priori, the number of plaintexts recoverable by our attack is 2^{13} times bigger than the number of plaintexts recoverable by the exhaustive search strategy (when $t = 2^{20}$). For instance, with a computational cost bounded by 2^{38}, our attack can recover plaintexts up to 2^{32}, while the exhaustive search can recover plaintexts up to 2^{19}. We successfully implemented the `OctoM` and `JordanM` cryptosystems as well as the described plaintext recovery attack in Python 2.7, with the intent to verify the effectiveness of the proposed attack. In practice, the security level of the target schemes affects the computational effort to perform the homomorphic operations as well as the modular arithmetic operations needed to evaluate a m-distinguisher. Therefore, such a dependency from the security level and/or the parameter sizes of the cryptoscheme is included in the computational complexity formulae of both our attack and the exhaustive search as the same multiplicative factor (which has

been omitted in the previous treatment). Hence, independently from the security margin, when the plaintext values are bounded (e.g. less than 2^{32}) our method largely improves the practicality of their derivation employing only ciphertext material.

5 Conclusions

We present a new type of plaintext recovery attack based on the capability of homomorphically evaluating the comparison between two encrypted integers and assuming the existence of an efficient algorithm to determine if a generic ciphertext is an encryption of a fixed value m. Although the computational cost of our attack is linear in the value of the plaintext being recovered, it significantly improves the number of recoverable plaintexts w.r.t. an exhaustive search strategy, which, in turn, might mean recovering a vast portion of ciphertexts in a FHE application scenario.

Acknowledgements. This work was supported in part by the EU Commission grant: "M2DC" (H2020 RIA) Grant agreement no. 688201.

References

1. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
2. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Stam, M. (ed.) IMACC 2013. LNCS, vol. 8308, pp. 45–64. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45239-0_4
3. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_50
4. Brakerski, Z.: When homomorphism becomes a liability. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 143–161. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_9
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS 2012, pp. 309–325. ACM (2012). https://doi.org/10.1145/2090236.2090262
6. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_29
7. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM J. Comput. **43**(2), 831–871 (2014). https://doi.org/10.1137/120868669
8. Çetin, G.S., Doröz, Y., Sunar, B., Martin, W.J.: An investigation of complex operations with word-size homomorphic encryption. ePrint Archive (1195) (2015). https://eprint.iacr.org/2015/1195.pdf

9. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 3–33. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_1
10. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC 2009, pp. 169–178. ACM (2009). https://doi.org/10.1145/1536414.1536440
11. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval and Johansson [19], pp. 465–482. https://doi.org/10.1007/978-3-642-29011-4_28
12. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5
13. Li, J., Wang, L.: Noise-free symmetric fully homomorphic encryption based on non-commutative rings. IACR ePrint Archive, Report 2015/641 (2015). https://eprint.iacr.org/2015/641
14. Kipnis, A., Hibshoosh, E.: Efficient methods for practical fully homomorphic symmetric-key encryption, randomization and verification. IACR ePrint Archive 2012, 637 (2012). http://eprint.iacr.org/2012/637
15. Liu, D.: Practical fully homomorphic encryption without noise reduction. IACR ePrint Archive 2015, 468 (2015). http://eprint.iacr.org/2015/468
16. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval and Johansson [19], pp. 700–718. https://doi.org/10.1007/978-3-642-29011-4_41
17. Narumanchi, H., Goyal, D., Emmadi, N., Gauravaram, P.: Performance analysis of sorting of FHE data: integer-wise comparison vs bit-wise comparison. In: AINA 2017, pp. 902–908. IEEE CS (2017). https://doi.org/10.1109/AINA.2017.85
18. Nuida, K.: A simple framework for noise-free construction of fully homomorphic encryption from a special class of non-commutative groups. IACR ePrint Archive 2014, 97 (2014). http://eprint.iacr.org/2014/097
19. Pointcheval, D., Johansson, T. (eds.): EUROCRYPT 2012. LNCS, vol. 7237. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4
20. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On Data Banks and Privacy Homomorphisms. Foundations of Secure Computation. Academia Press, Ghent (1978)
21. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Cryptogr. **71**(1), 57–81 (2014). https://doi.org/10.1007/s10623-012-9720-4
22. Tsaban, B., Lifshitz, N.: Cryptanalysis of the MORE symmetric key fully homomorphic encryption scheme. J. Math. Cryptol. **9**(2), 75–78 (2015). https://doi.org/10.1515/jmc-2014-0013
23. Wang, Y., Malluhi, Q.M.: Privacy preserving computation in cloud using noise-free fully homomorphic encryption (FHE) schemes. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9878, pp. 301–323. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45744-4_15
24. Yao, A.C.: Protocols for secure computations (extended abstract). In: FOCS 1982, pp. 160–164. IEEE CS (1982). https://doi.org/10.1109/SFCS.1982.38