Privacy-friendly Forecasting for the Smart Grid using Homomorphic Encryption and the Group Method of Data Handling

Joppe W. Bos[1], Wouter Castryck[2,3], Ilia Iliashenko[2], Frederik Vercauteren[2,4]

[1] NXP Semiconductors
joppe.bos@nxp.com

[2] Cosic, Dept. Electrical Engineering, KU Leuven and imec,
firstname.lastname@kuleuven.be

[3] Laboratoire Paul Painlevé, Université de Lille-1

[4] Open Security Research

Abstract. While the smart grid has the potential to have a positive impact on the sustainability and efficiency of the electricity market, it also poses some serious challenges with respect to the privacy of the consumer. One of the traditional use-cases of this privacy sensitive data is the usage for forecast prediction. In this paper we show how to compute the forecast prediction such that the supplier does not learn any individual consumer usage information. This is achieved by using the Fan-Vercauteren somewhat homomorphic encryption scheme. Typical prediction algorithms are based on artificial neural networks that require the computation of an activation function which is complicated to compute homomorphically. We investigate a different approach and show that Ivakhnenko's group method of data handling is suitable for homomorphic computation.
Our results show this approach is practical: prediction for a small apartment complex of 10 households can be computed homomorphically in less than four seconds using a parallel implementation or in about half a minute using a sequential implementation. Expressed in terms of the mean average percentage error, the prediction accuracy is roughly 21%.

1 Introduction

One of the promising solutions to cope with current and future challenges of electricity supply is the *smart grid*. With the prospect of having a positive impact on the sustainability, reliability, flexibility, and efficiency many countries around the world are investing significantly in such smart grid solutions. The deployment of smart meters is already well underway. For example, in the United Kingdom the large energy suppliers were operating over $400,000$ smart gas and electricity meters, representing 0.9 percent of all the domestic meters operated by the large suppliers in 2014 [9]. This development is expected to continue and intensify:

This work was supported by the European Commission through the ICT programme under contract H2020-ICT-2014-1 644209 HEAT.

the EU third energy package has as an objective to replace at least 80 percent of electricity meters with smart meters by 2020 [15]. This change will fundamentally re-engineer the (electricity) service industry.

The replacement of the classical meters with their smart variants has advantages for both the consumer and industry. Some of the key benefits include giving consumers the information to gain control over their energy consumption, lowering the cost for managing the supply of energy across industry, and producing detailed consumption information data from these smart meters which in turn enable a wide range of services [9]. It is expected that the meters have an update rate of every 15 minutes at least [14]. When generating such a large amount of consumer data a lot of privacy sensitive information is being disclosed. There are various initiatives (e.g. [31,36]) which stress and outline the importance of having solutions for the smart grid where privacy protecting mechanisms are already built-in by design.

This work is concerned with enhancing the privacy of the smart meter readings in the setting of *forecast prediction*: energy suppliers need to forecast in order to buy energy generation contracts that cover their clients. Moreover, to ensure network capacity the network operators require longer term forecasting [23,36,10]. This forecasting is typically done by taking as input the (aggregated) data from a number of households. Based on this consumption data, together with other variables such as the date and the current temperature and weather, a forecast is computed to predict the short, medium, or long term consumption. The energy providers or network operators only need to know the desired forecast information based on their (potentially proprietary) forecasting algorithm and model. There is no need to observe the individual consumer data.

We investigate the potential of *fully homomorphic encryption* (FHE) to realize this goal. The notion of FHE was introduced in the late 1970s [33] and a concrete instantiation was found in 2009 by Gentry [19]. FHE allows an untrusted party to carry out arbitrary computation on *encrypted data* without learning anything about the content of this data. Currently, the Fan-Vercauteren (FV) FHE scheme [16] is regarded as the best choice with respect to security and practical performance. See Section 4 for a more detailed description of the FV scheme. Additively homomorphic encryption schemes [30] and other tools have been proposed to enhance the privacy in the setting of computing detailed billing in the context of the smart grid [32,29,18,25,13,24]. However, these approaches cannot be directly used in the setting of prediction algorithms since these more complex algorithms need to compute both additions and multiplications.

One popular class of algorithms which are used for prediction are based on artificial neural networks. One of the main ingredients in these forecasting algorithms is the computation of the so-called activation function, in practice it is common to use a sigmoid function where the logistic function $t \mapsto 1/(1+e^{-t})$ is a popular choice. However, computing such a sigmoid function homomorphically is far from practical. One possible way to proceed is to simply ignore the sigmoidality requirement and to proceed with a truncated Taylor series approximating this function or, more generally, to use any non-linear polynomial function which is *simple*. This was investigated by Livni et al. [26] regardless of cryptographic applications. Recent work by Xie et al. [38] and Dowlin et al. [12] suggests to apply the same approach to homomorphically encrypted data. However, by computing artificial neural networks in this fashion it becomes just an organized manner of fitting a polynomial through the given data set. In this paper we investigate an older tool for realizing this goal. Namely, we show that Ivakhnenko's group method of data handling (GMDH) which was proposed back in 1970 [22] is a perfect match for being computed homomorphically. Moreover, a recent comparison analysis between different forecasting methods [35] showed that GMDH produced significantly more accurate results compared to the other methods considered.

We show that GMDH can be implemented homomorphically using the recent fixed point approach from [11,6]. Using a five-layered network (one input layer, three hidden layers and an output node) we are able to homomorphically predict the next half-hour energy consumption for an apartment complex of 10 households. Our software implementation results indicate that this requires less than four seconds using a parallel implementation or about half a minute using a sequential implementation while the prediction accuracy expressed using the mean absolute percentage error (MAPE, see Section 3 for a definition) is only 21 percent. This shows that privacy preserving forecasting using homomorphic encryption is indeed practical.

2 The Smart Grid and Privacy Concerns

The authors of [34] define the smart grid as "*an electricity network that can cost efficiently integrate the behavior and actions of all users connected to it – generators, consumers and those that do both – in order to ensure economically efficient, sustainable power system with low losses and high levels of quality and security of supply and safety*". This paper is concerned with the cryptographic solutions to privacy concerns within the smart grid. Within this scope we assume that the meters are protected against various types of side-channel attacks such that no secret data can be retrieved from the device when it is operating (e.g. key extraction). Moreover, we assume that the smart metering device acts honestly in accordance with the implementation or protocol given to it. These assumptions avoid the usual security threats and leave us with the privacy related concerns which we aim to address.

In the early 1990s, Hart showed a non-intrusive approach where by monitoring the electric load one can observe the individual appliances turning on and off [20]. Hence, detailed smart meter readings, which are expected to be generated at least every 15 minutes in the context of the smart grid (cf. [14]), can be used to derive various privacy sensitive information about a house-hold or even an apartment complex. In order to grasp where the main privacy challenges are in smart metering it is good to understand how and when the meter readings are used in practice by the various parties involved. As identified by the survey paper [23], which in turn has collected this information from the privacy impact assessment by NIST [36] and the enumeration of data uses by the consultation of the British Department of Energy and Climate Change [10], the key usages of smart meter readings include the usage for *load monitoring and forecasting* and *smart billing.*

There has been a significant amount of work related to privacy-preserving smart billing solutions for the smart grid. One line of research allows complex non-linear tariff policies where the bill is computed and sent along with a zero-knowledge proof to ensure that the computations are correct [32,29]. Another approach is based on privacy-friendly aggregation schemes (e.g. using additively homomorphic encryption schemes such as the Paillier scheme [30]) where one can compute a function on the ciphertexts which corresponds to adding the plaintexts [18,25,13,24]. Such approaches heavily rely on the fact that only aggregation of the results is required. As soon as more complex operations need to be computed (such as a large number of multiplications) one has to look for other solutions.

One example where more complex operations are performed is in the setting of load monitoring and forecasting. There are many different forecasting approaches (see e.g. the survey paper [21] on this topic and the references therein). One of the popular and well-studied techniques is using artificial neural networks (see e.g. [1,17]). In the next section we describe how such neural networks operate, analyze the challenges they pose when being evaluated in the encrypted domain, and discuss how this naturally leads to considering the group method of data handling as an alternative forecasting tool.

3 Neural Networks versus The Group Method of Data Handling

Over time, artificial neural networks (ANNs) have manifested themselves among the most popular and reliable prediction tools for various purposes, including load forecasting. For our preliminary discussion, it suffices to think of an ANN as a real-valued function $f : \mathbf{R}^{n_0} \to \mathbf{R}$ that arises as the composition of a number of 'neurons' $\nu_{ij} : \mathbf{R}^{n_{i-1}} \to \mathbf{R}$, organized in layers $i = 1, \ldots, r$, as depicted in Figure 1. Each neuron is of the form

$$\nu_{ij} : \mathbf{R}^{n_{i-1}} \to \mathbf{R} : (x_1, x_2, \ldots, x_{n_{i-1}}) \mapsto g\left(\sum_{k=1}^{n_{i-1}} w_{ijk} x_k - b_{ij}\right)$$

for weights and biases $w_{ijk}, b_{ij} \in \mathbf{R}$ and some fixed sigmoidal activation function $g : \mathbf{R} \to \mathbf{R}$, such as the logistic function $t \mapsto 1/(1 + e^{-t})$. The global shape of the network is decided in advance, and the goal is to determine the weights w_{ijk} and the biases b_{ij} such that f approximates an unknown target function $\tilde{f} : \mathbf{R}^{n_0} \to \mathbf{R}$, in our case load prediction, as well as possible. This is done during a so-called supervised learning phase. One starts from a reasonable guess, after which the network's performance is assessed by feeding to it a number of input-output pairs of $\tilde{f}$, taken from a given data set, and measuring the error. During

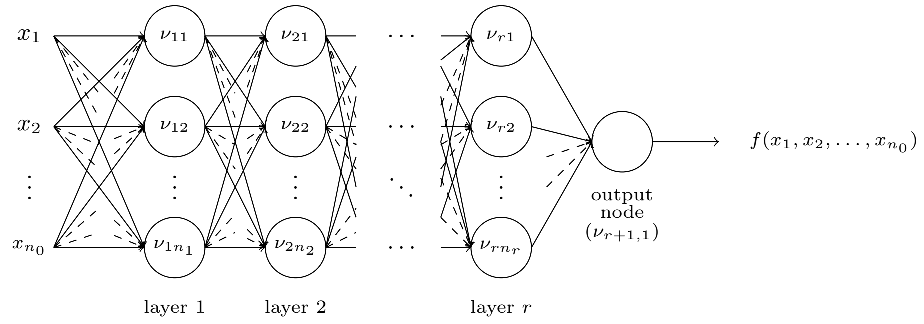

Fig. 1. Design of an Artificial Neural Network (ANN).

a process called backpropagation, which is based on the chain rule for derivation, the weights and biases are then modified repeatedly, in the hope of converging to values that minimize the error.

The backpropagation method requires the activation function g to have a nice and easy derivative, while at the same time it should be sigmoidal, i.e. its graph should have the typical step-like activation shape, allowing the ANNs to do what they were designed for: to simulate computation in (an area of) the human brain. Unfortunately, the class of such functions does not contain examples that are easy to evaluate homomorphically. A natural attempt would be to use a Taylor approximation to the logistic function or to one of its known alternatives, but such approximations become highly non-sigmoidal away from the origin.

One way out is simply to ignore the sigmoidality requirement and to proceed with this truncated Taylor series, or more generally to replace g by any simple non-linear polynomial function, the easiest choice being $t \mapsto t^2$. This has been investigated by Livni et al. [26] for reasons of computational efficiency, regardless of cryptographic applications. Recent work by Xie et al. [38] and Dowlin et al. [12] suggested to apply the same approach to homomorphically encrypted data. The resulting neural networks were named ‘crypto-nets’.

However in this way the ANN just becomes an organized way of fitting a polynomial through the given data set. There exist older and simpler prediction tools that do this. In this paper we study one of the oldest such tools, namely Ivakhnenko’s group method of data handling (GMDH) from 1970 [22]. Besides being suited for applications using homomorphic encryption, one particular feature is that its performance in the context of load forecasting enjoys a large amount of existing literature, at times even with results that are superior to ANNs. Indeed, a comparison analysis between different forecasting methods from 2008 [35] showed that GMDH produced significantly more accurate results compared to the other methods considered.

The basic version of GMDH works as follows, although many variations are possible (and seem to deserve a further analysis). The goal is to approximate

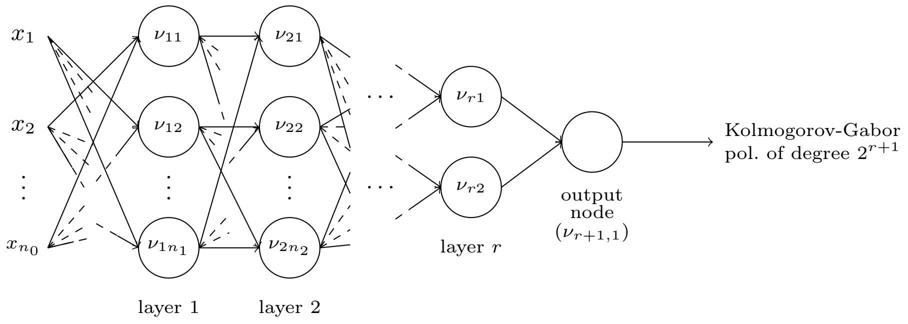

Fig. 2. Network-like illustration of the Group Method of Data Handling.

our target function $\tilde{f} : \mathbf{R}^{n_0} \to \mathbf{R}$ with a truncated Wiener series

$$a_0 + \sum_{i=1}^{n_0} a_i x_i + \sum_{i=1}^{n_0} \sum_{j=i}^{n_0} a_{ij} x_i x_j + \sum_{i=1}^{n_0} \sum_{j=i}^{n_0} \sum_{k=j}^{n_0} a_{ijk} x_i x_j x_k + \dots,$$

which is also called a Kolmogorov-Gabor polynomial. The idea is to approach this by a finite superposition of quadratic polynomials

$$\nu_{ij} : \mathbf{R}^2 \to \mathbf{R} : (x, y) \mapsto b_{ij0} + b_{ij1}x + b_{ij2}y + b_{ij3}xy + b_{ij4}x^2 + b_{ij5}y^2$$

along a diagram of the kind depicted in Figure 2. One can think of this as some sort of ANN, and indeed the diagram is sometimes called a 'polynomial neural network'. As a first main difference, however, note that the wiring is incomplete: each neuron has two inputs only.

Also the learning phase is quite different from the one in conventional ANNs. Here the goal is to determine the coefficients b_{ijk} of the quadratic polynomials ν_{ij}, but also the concrete structure of the network, which is not fixed in advance. Indeed, one decides beforehand on the number of layers r and the number of neurons n_i in each layer, but the wiring between these is defined during the learning process. Recall that each node can have only two inputs, so the following constraint should be satisfied: $n_i \leq \binom{n_{i-1}}{2}$. In order to prevent exponential growth of the number of neurons, the left hand side will in general be much smaller than the right hand side. As to *which* combinations end up being chosen, one first considers all possible combinations and then removes the $\binom{n_{i-1}}{2} - n_i$ worst neurons with respect to their error performance, in the sense explained below, while at the same time determining the coefficients b_{ijk} of the surviving neurons. One then proceeds with the next layer. In particular, there is no backpropagation. The node with the smallest error performance will be assigned as an output for the whole network; this may in fact be different from what was initially foreseen to become the output neuron. One sometimes applies the rule that if at some point all nodes in layer i perform worse than the best performing node in layer $i - 1$, then the algorithm stops, and the latter node is assigned as the output.

To assess the error performance of a neuron, while at the same time determining the coefficients of the corresponding quadratic polynomial, one uses a given data set of correct input-output pairs for $\tilde{f}$. Additionally, an error (or loss) function should be set up beforehand. Throughout this paper we use the Mean Squared Error (MSE) function

$$\mathrm{MSE}((y_1^{\text{forecast}}, \ldots, y_n^{\text{forecast}}), (y_1^{\text{actual}}, \ldots, y_n^{\text{actual}})) = \frac{1}{n} \sum_{i=1}^{n} (y_i^{\text{forecast}} - y_i^{\text{actual}})^2,$$

but there are a couple of other standard choices, such as the Mean Absolute Error (MAE) and the Mean Absolute Percentage Error (MAPE):

$$\frac{1}{n} \sum_{i=1}^{n} \left| y_i^{\text{forecast}} - y_i^{\text{actual}} \right| \qquad \text{resp.} \qquad \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i^{\text{forecast}} - y_i^{\text{actual}}}{y_i^{\text{actual}}} \right|.$$

For each neuron ν_{ij} the data set is randomly split into a learning set and a test set. This is done to avoid overfitting, where the network learns too much about the inherent noise always being present in real-world data. The learning set is used to determine the coefficients b_{ijk}, by choosing them such that the error is as small as possible. In the case of MSE this can be achieved by linearization of the quadratic polynomial and applying the least squares method. The test set is then used to assess the performance of the neuron.

4 The Fan-Vercauteren SHE scheme

In this section we briefly describe a simplified version of the FV scheme [16], which we will present in its *somewhat* homomorphic encryption (SHE) form, meaning that it is suitable only for computations up to a given depth, thereby avoiding very expensive bootstrapping operations. It concerns a scale-invariant SHE scheme based on the hardness of the ring version of the learning with errors problem (RLWE) [27]. It works in the polynomial ring $R = \mathbf{Z}[X]/(f(X))$ with $f(X) = X^d + 1$ and $d = 2^n$. For an integer N we denote with R_N the reduction of R modulo N. Abusing notation, elements of R will often be identified with their unique representant in $\mathbf{Z}[X]$ of degree at most $d-1$, and similarly elements of R_N are identified with their unique representant inside

$$\left\{ \alpha_{d-1}X^{d-1} + \alpha_{d-2}X^{d-2} + \ldots + \alpha_0 \;\middle|\; \alpha_i \in (-N/2, N/2] \text{ for all } i \right\},$$

but this should cause no confusion. For an element $a \in R$ or $a \in \mathbf{Z}[X]$ we write $[a]_N$ do denote its reduction inside the above set of representants.

The plaintext space in the FV scheme is given by the ring R_t for some small integer modulus $t > 1$, while a ciphertext is given by a pair of ring elements in R_q where $q > 1$ is a much larger modulus. The key generation and the encryption operations in the FV scheme require sampling from two probability distributions defined on R, denoted χ_{key} and χ_{err}. The security of the scheme is determined by the degree d of f, the size of q, and by the probability distributions. Typically

χ_{key} and χ_{err} are coefficient-wise discrete Gaussian distributions centered around 0 and having a small standard deviation, but in practice one often samples the coefficients of the key from a uniform distribution on a narrow set like $\{-1, 0, 1\}$. The RLWE distribution on $R_q \times R_q$ is constructed as follows: first choose a fixed element $s \leftarrow \chi_{\text{key}}$, and then generate samples of the form (a, b) with $a \leftarrow R_q$ uniformly random and $b = [-(as + e)]_q$ with $e \leftarrow \chi_{\text{err}}$. (The minus sign is not standard but makes a better fit with the discussion below.) The decision RLWE problem is then to distinguish between the RLWE distribution and the uniform distribution on $R_q \times R_q$. The search RLWE problem is to retrieve s from polynomially many samples. Both problems are believed to be very hard for an appropriate choice of parameters.

By construction, for a RLWE sample (a, b) we have that $e = -[as + b]_q$ and therefore that the right-hand side has small coefficients, with overwhelming probability. Furthermore note that the sample can be easily re-randomized without knowledge of s as follows: choose $u \leftarrow \chi_{\text{key}}$ and $e_1, e_2 \leftarrow \chi_{\text{err}}$ and form the new sample as $(ua + e_1, ub + e_2)$. In the encryption scheme below, the public key will consist of a single RLWE sample, which will be re-randomized during encryption. The new RLWE sample will then be used as an additive mask to encrypt a message $m \in R_t$. Before we present the FV scheme, we first describe some subroutines that are required in the algorithm:

– $\texttt{WordDecomp}_{w,q}(a)$: This function is used to decompose a ring element $a \in R_q$ in base w by splicing each coefficient of a. For $u = \lfloor \log_w(q) \rfloor$, it returns $a_i \in R$ with coefficients in $(-w/2, w/2]$, such that $a = \sum_{i=0}^{u} a_i w^i$.
– $\texttt{PowersOf}_{w,q}(a)$: This function scales an element $a \in R_q$ by the different powers of w. It is defined as $\texttt{PowersOf}_{w,q}(a) = (aw^i)_{i=0}^{u}$.

These two functions can be used to perform a polynomial multiplication in R_q through an inner product: $\langle \texttt{WordDecomp}_{w,q}(a), \texttt{PowersOf}_{w,q}(b) \rangle = a \cdot b$. This expression has advantage in reducing the noise during homomorphic multiplications, as the first vector contains small elements only.

The FV scheme consists of an encryption scheme augmented with additional functions `Add`, `Mult`, and `ReLin` to compute homomorphically on encrypted data.

1. $\texttt{ParamsGen}(\lambda)$: For a given security parameter λ, choose a degree $d = 2^n$ and thus a polynomial $f(X) = X^d + 1$, moduli q and t and distributions χ_{err} and χ_{key}. Also choose the base w for $\texttt{WordDecomp}_{w,q}(\cdot)$. Return the system parameters $(d, q, t, \chi_{\text{err}}, \chi_{\text{key}}, w)$.
2. $\texttt{KeyGen}(d, q, t, \chi_{\text{err}}, \chi_{\text{key}}, w)$: Sample the secret key $s \leftarrow \chi_{\text{key}}$, sample $a \leftarrow R_q$ uniformly at random, and sample $e \leftarrow \chi_{\text{err}}$. Compute $b = [-(as + e)]_q$. The public key is the pair $\mathbf{pk} = (b, a)$ and the secret key is $\text{sk} = s$. The scheme uses another key $\mathbf{rlk}$ called *relinearization key* in the function `ReLin` below. Define $\ell = u+1 = \lfloor \log_w(q) \rfloor + 1$, sample a vector $\mathbf{a} \leftarrow R_q^{\ell}$ uniformly at random, sample $\mathbf{e} \leftarrow \chi_{\text{err}}^{\ell}$, and let $\mathbf{rlk} = ([\texttt{PowersOf}_{w,q}(s^2) - (\mathbf{e} + \mathbf{a} \cdot s)]_q, \mathbf{a}) \in R_q^{\ell} \times R_q^{\ell}$.
3. $\texttt{Encrypt}(\mathbf{pk}, m)$: First encode the input message $m \in R_t$ into a polynomial $\Delta m \in R_q$ with $\Delta = \lfloor q/t \rfloor$. Next sample the error polynomials $e_1, e_2 \leftarrow \chi_{\text{err}}$,

sample $u \leftarrow \chi_{\text{key}}$, and compute the two polynomials $c_0 = \Delta m + bu + e_1 \in R_q$ and $c_1 = au + e_2 \in R_q$. The ciphertext is the pair of polynomials $\mathbf{c} = (c_0, c_1)$.

4. `Decrypt`(sk, $\mathbf{c}$): First compute the polynomial $\tilde{m} = [c_0 + sc_1]_q$. Then recover the plaintext message m by a decoding the coefficients of $\tilde{m}$ by scaling down by Δ and rounding.
5. `Add`($\mathbf{c_1}, \mathbf{c_2}$): For two ciphertexts $\mathbf{c_1} = (c_{1,0}, c_{1,1})$ and $\mathbf{c_2} = (c_{2,0}, c_{2,1})$, return $\mathbf{c} = (c_{1,0} + c_{2,0}, c_{1,1} + c_{2,1}) \in R_q \times R_q$.
6. `Mult`($\mathbf{c}_1, \mathbf{c}_2, \mathbf{rlk}$): Compute $\tilde{\mathbf{c}}_{\text{mult}} = (c_0, c_1, c_2)$ where $c_0 = \lfloor \frac{t}{q} \cdot c_{1,0} \cdot c_{2,0} \rceil$, $c_1 = \lfloor \frac{t}{q} \cdot (c_{1,0} \cdot c_{2,1} + c_{1,1} \cdot c_{2,0}) \rceil$, and $c_2 = \lfloor \frac{t}{q} \cdot c_{1,1} \cdot c_{2,1} \rceil$ and apply relinearization.
7. `ReLin`($\tilde{\mathbf{c}}_{\mathbf{mult}}, \mathbf{rlk}$): Write $\mathbf{rlk} = (\mathbf{b}, \mathbf{a})$ and $\tilde{\mathbf{c}}_{\text{mult}} = (c_0, c_1, c_2)$, then compute a relinearized ciphertext as $\mathbf{c}' = (c_0', c_1')$ as $([c_0 + \langle \texttt{WordDecomp}_{w,q}(c_2), \mathbf{b} \rangle]_q, [c_1 + \langle \texttt{WordDecomp}_{w,q}(c_2), \mathbf{a} \rangle]_q)$.

Given an FV ciphertext $\mathbf{c} = (c_0, c_1)$, we can write $[c_0 + c_1 s]_q = \Delta m + e$, where e is called the noise inside the ciphertext. Every operation on ciphertexts causes the noise to increase. It is clear that when the noise gets too large, in particular if $\|e\|_\infty > \Delta/2$, correct decryption will fail, where $\|\cdot\|_\infty$ denotes the maximal absolute value of the coefficients.

From now on we assume that χ_{err} is a coefficient-wise discrete Gaussian with standard deviation σ and that χ_{key} samples the coefficients uniformly from $\{-1, 0, 1\}$. With overwhelming probability $B_{\text{err}} = 6\sigma$ and $B_{\text{key}} = 1$ are upper bounds on the absolute values of the coefficients of their respective samples. Therefore we can use $V = B_{\text{err}}(1 + 2dB_{\text{key}}) = B_{\text{err}}(1 + 2d)$ as an upper bound on the noise of the input ciphertexts. When doing arithmetic the noise is affected in the following way. Firstly, adding ciphertexts $\mathbf{c}_1$ and $\mathbf{c}_2$ corresponds to adding the noises, potentially augmented by a carryover γ satisfying $\|\gamma\|_\infty < t$, as explained in [16]. Secondly, multiplying a ciphertext $\mathbf{c}$ by an unencrypted scalar $(\Delta\alpha, 0)$ for some $\alpha \in R_t$ corresponds to multiplying the noise by α, again with some carryover γ. For use below, fix an integer $\lambda \geq 1$ and assume that the coefficients of α are in $\{-1, 0, 1\}$ with at most λ of them being non-zero. Then in a similar way one sees that $\|\gamma\|_\infty < \lfloor \lambda/2 \rfloor \cdot t$. Thirdly, multiplying two ciphertexts $\mathbf{c}_1$ and $\mathbf{c}_2$ whose noise coefficients are bounded by E results in a ciphertext whose noise coefficients are at most

$$2 \cdot E \cdot t \cdot d \cdot (d+1) + 8 \cdot t^2 \cdot d^2 + \ell \cdot B_{\text{err}} \cdot w \cdot d/2$$

in absolute value, by [16, Lem. 2 & Lem. 3].

Now assume that we wish to evaluate a GMDH network $f : R_t^{n_0} \to R_t$ having r hidden layers in a fresh component-wise encryption of an n_0-tuple $(x_1, x_2, \ldots, x_{n_0}) \in R_t^{n_0}$. For the moment just think of this as a Kolmogorov-Gabor polynomial that we evaluate in the encrypted domain along a diagram of the kind depicted in Figure 2; the purpose of this will become clear in the next section. The network parameters b_{ijk} are assumed to be small public scalars along the lines mentioned above: the coefficients are in $\{-1, 0, 1\}$ and at most λ of them are non-zero. Define $A_1 = 6 \cdot \lambda \cdot t \cdot d \cdot (d+1) + 2 \cdot \lambda$ and

$$A_2 = 3/2 \cdot \lambda \cdot \ell \cdot B_{\text{err}} \cdot w \cdot d + 24 \cdot t^2 \cdot d^2 + 5 \cdot (\lfloor \lambda/2 \rfloor + 1) \cdot t.$$

One verifies that homomorphically evaluating a node $\nu_{1j} : R_t^{n_0} \to R_t^{n_1}$ from the first layer causes the noise coefficients to grow to at most $A_1 \cdot V + A_2$. Recursively applying this formula yields the upper bound $A_1^{r+1} \cdot V + (A_1^{r+1} - 1) \cdot A_2/(A_1 - 1)$ on the absolute values of the noise coefficients that are present in the output of the entire network f.

The parameters of the FV scheme are not only determined by the noise growth, but also by the security requirements. It is easy to see that when d and σ/q grow, amounting to larger polynomials and more noise in the ciphertexts, then RLWE becomes harder. A precise security analysis is beyond the scope of this paper, but to derive our security estimates we closely follow the work by Albrecht, Player and Scott [3] and the open source LWE-estimator implemented by Albrecht [2]. In particular, the LWE-estimator allows one to estimate the concrete hardness of the LWE problem given the dimension d, the modulus q and the standard deviation σ. Note that the actual tool takes as input the parameter $\alpha = \sqrt{2\pi}\sigma/q$, instead of σ directly.

For the design reasons explained in Section 6 we will take $r = 3$, $\lambda = 9$, while for compatibility reasons with the software library `FV-NFLlib` [7] we wish to take $w = 2^{32}$ and $\log_2 q$ an integral multiple of 62. Targetting a security level of 80 bits, we can address the restrictions coming from both the noise growth and the security considerations by using the parameter set $d = 4096$, $q \simeq 2^{186}$ and $\sigma = 102$ (corresponding to $\alpha = 256/q$). These parameters will be used throughout the remainder of the paper and allow for usage of all plaintext moduli $t \leq 396$. Note that one ciphertext takes up 186kB space.

5 Representing fixed-point numbers in plaintext space

Our final goal is to evaluate a trained GMDH network in the encrypted domain using the FV scheme. As explained in the previous section, the plaintext space is of the form R_t, which is the reduction modulo a certain integer $t > 1$ of $R = \mathbf{Z}[X]/(X^d + 1)$, where $d = 2^n$ for some $n \in \mathbf{Z}_{>0}$. Therefore an important task is to encode the input values $x_1, x_2, \ldots, x_{n_0} \in \mathbf{R}$, as well as the coefficients $b_{ijk} \in \mathbf{R}$, as elements of R_t. This should be done in such a way that real additions and multiplications agree with the corresponding operations in the ring R_t, up to a certain depth of computation. Dowlin et al. [11] proposed two ways of addressing this issue, which were revisited in a recent paper by Costache et al. [6], who showed them to be essentially equivalent, and also provided lower bounds on t and d guaranteeing that the arithmetic in $\mathbf{R}$ is indeed compatible with that in R_t to the extent desired. We briefly recall their main conclusions, adapted to our setting.

On the real number side, we use fixed-point arithmetic. We assume that the x_i's and the b_{ijk}'s are given in balanced ternary expansion to some finite precision, that is, they are of the form

$$b_{\ell_1-1}b_{\ell_1-2}\ldots b_0 \,.\, b_{-1}b_{-2}\ldots b_{-\ell_2} \qquad (1)$$

with $b_i \in \{-1, 0, 1\}$ for $i = -\ell_2, \ldots, \ell_1 - 1$. This should be read as

$$b_{\ell_1-1}3^{\ell_1-1} + b_{\ell_1-2}3^{\ell_1-2} + \ldots + b_0 3^0 + b_{-1}3^{-1} + b_{-2}3^{-2} + \ldots + b_{-\ell_2}3^{-\ell_2}.$$

As usual we say that (1) has ℓ_1 integral digits and ℓ_2 fractional digits; throughout we assume that $\ell_1 \geq 1$ and $\ell_2 \geq 0$. In order to encode (1) as an element of R_t one simply replaces the base 3 by X. This yields

$$b_{\ell_1-1}X^{\ell_1-1} + b_{\ell_1-2}X^{\ell_1-2} + \ldots + b_0 X^0 + b_{-1}X^{-1} + b_{-2}X^{-2} + \ldots + b_{-\ell_2}X^{-\ell_2}, \quad (2)$$

which one can rewrite as

$$b_{\ell_1-1}X^{\ell_1-1} + b_{\ell_1-2}X^{\ell_1-2} + \ldots + b_0 X^0 + b_{-1}X^{d-1} + b_{-2}X^{d-2} + \ldots + b_{-\ell_2}X^{d-\ell_2},$$

using the relation $X^d \equiv -1$.

To decode a given element of R_t one first considers its unique representant inside $\left\{ \alpha_{d-1}X^{d-1} + \alpha_{d-2}X^{d-2} + \ldots + \alpha_0 \;\middle|\; \alpha_i \in (-t/2, t/2] \text{ for all } i \right\}$, after which one replaces all suitably high powers X^i by $-X^{i-d}$, and one evaluates the resulting Laurent polynomial at 3. The outcome is a rational number whose denominator is a power of 3, so it can be easily rewritten in balanced ternary expansion. For simplicity we think of 'suitably high' as $i > d/2$, although to improve the bound on d in Lemma 1 below, a more careful (but easy) estimation should be made, that takes into account the lengths of the integral and fractional parts of the fixed-point numbers involved.

Clearly, the ring operations in R_t are compatible with fixed-point arithmetic on the real number side as long as they do not involve 'wrapping around' modulo t and/or modulo $X^d + 1$. (In the latter case this means that neither the terms of high degree nor the terms of low degree are allowed to cross the separation point $X^{d/2}$.) Thus t and d should be taken large enough to ensure this, for which Costache et al. elaborated concrete lower bounds. We will not explicitly rely on these bounds, but rather apply the underlying ideas to obtain a more implicit statement. For all integers $\ell \geq 0, \lambda \geq 0, r \geq -1$ we define $d_{\ell,\lambda,r} := 2^{r+1}\ell + (2^{r+1} - 1)\lambda$. Moreover for all $\ell_1 \geq 1, \lambda_1 \geq 1, \ell_2 \geq 0, \lambda_2 \geq 0, r \geq -1$ we introduce a polynomial $D_{\ell_1,\lambda_1,\ell_2,\lambda_2,r}(X) \in \mathbf{Z}[X]$, which is recursively defined by putting

$$D_{\ell_1,\lambda_1,\ell_2,\lambda_2,-1}(X) = 1 + X + X^2 + \ldots + X^{\ell_1+\ell_2-1}$$

and for $r \geq 0$ letting $D_{\ell_1,\lambda_1,\ell_2,\lambda_2,r}(X)$ be

$$X^{2d_{\ell_2,\lambda_2,r-1}} + 2X^{d_{\ell_2,\lambda_2,r-1}} D_{\ell_1,\lambda_1,\ell_2,\lambda_2,r-1}(X) + 3D_{\ell_1,\lambda_1,\ell_2,\lambda_2,r-1}(X)^2$$

multiplied with $1 + X + X^2 + \ldots + X^{\lambda_1+\lambda_2-1}$. We then define $c_{\ell_1,\lambda_1,\ell_2,\lambda_2,r} = \|D_{\ell,\lambda,r}(X)\|_\infty$ where as before $\|\cdot\|_\infty$ denotes the maximal absolute value of the coefficients. Note that $\deg D_{\ell,\lambda,r}(X) = d_{\ell_1+\ell_2-1,\lambda_1+\lambda_2-1,r}$. This all looks a bit cumbersome but the idea underlying these definitions should become apparent from the proof below.

Lemma 1. *Suppose that the input values $x_1, x_2, \ldots, x_{n_0}$ resp. the coefficients b_{ijk} are given by balanced ternary expansions of at most ℓ_1 resp. λ_1 integral* *digits and ℓ_2 resp. λ_2 fractional digits. Let x_{out} be the evaluation of our GMDH network at the x_i's, obtained by using fixed-point arithmetic. Let $\phi(X) \in R_t$ be the evaluation of our GMDH network at the encodings of the x_i's (using the encodings of the b_{ijk}'s as coefficients), obtained by using the respective ring operations in R_t. If*

$$t \geq 2 \cdot c_{\ell_1,\lambda_1,\ell_2,\lambda_2,r} \quad \text{and} \quad d \geq 2 \cdot \max\{d_{\ell_1+\ell_2-1,\lambda_1+\lambda_2-1,r}, d_{\ell_2,\lambda_2,r} + 1\}$$

then $\phi(X)$ decodes to x_{out}.

Proof. Consider the evaluation of our GMDH network when carried out in $\mathbf{Z}[X, X^{-1}]$, using encodings of the form (2). We claim that the outcome is of the form $X^{-m}g(X)$ with $m \leq d_{\ell_2,\lambda_2,r}$ and $g(X) \in \mathbf{Z}[X]$ of degree at most $d_{\ell_1+\ell_2-1,\lambda_1+\lambda_2-1,r}$ and having coefficients bounded (in absolute value) by $c_{\ell_1,\lambda_1,\ell_2,\lambda_2,r}$. This claim clearly implies the lemma.

The key observation is that if one replaces all inputs by $X^{-\ell_2} + X^{-\ell_2+1} + X^{-\ell_2+2} + \ldots + X^{\ell_1-1}$ while replacing all encoded b_{ijk}'s by $X^{-\lambda_2} + X^{-\lambda_2+1} + X^{-\lambda_2+2} + \ldots + X^{\lambda_1-1}$ then these quantities can only increase, by the triangle inequality for the absolute value. By induction on r, it is easy to show that the corresponding evaluation is precisely $X^{-d_{\ell_2,\lambda_2,r}} \cdot D_{\ell_1,\lambda_1,\ell_2,\lambda_2,r}(X)$, from which the claim follows. ■

These bounds are easy to compute in practice, using a computer algebra package. For example with $\ell_1 = 4$, $\ell_2 = 1$, $\lambda_1 = 1$, $\lambda_2 = 8$ and $r = 3$, along the Magma script below we obtain the bounds

$$t \geq 93659577705415581454099599864654 \approx 2^{106.207} \quad \text{and} \quad d \geq 368. \tag{3}$$

This concrete choice of parameters will reoccur later in the paper.

```
ell1 := 4; lambda1 := 1; ell2 := 1; lambda2 := 8; r := 3;
R<X> := PolynomialRing(Integers());
D := &+[X^i : i in [0..ell1 + ell2 - 1]];
beta := &+[X^i : i in [0..lambda1 + lambda2 - 1]];
depth := -1;
repeat
    depth +:= 1;
    d := 2^depth*ell2 + (2^depth-1)*lambda2;
    D := beta*(X^(2*d) + 2*X^d*D + 3*D^2);
until depth eq r;
print "Bound on t is", 2*Maximum(Coefficients(D));
print "Bound on d is", 2*Maximum(Degree(D), 2*d + lambda2 + 1);
```

One sees that the obtained bound on t is very large, which is problematic for a direct application of the FV scheme: remember from the previous section that we need $t \leq 396$. To address this issue we follow an idea mentioned in [4, §5.5], namely to decompose the plaintext space using the Chinese Remainder Theorem

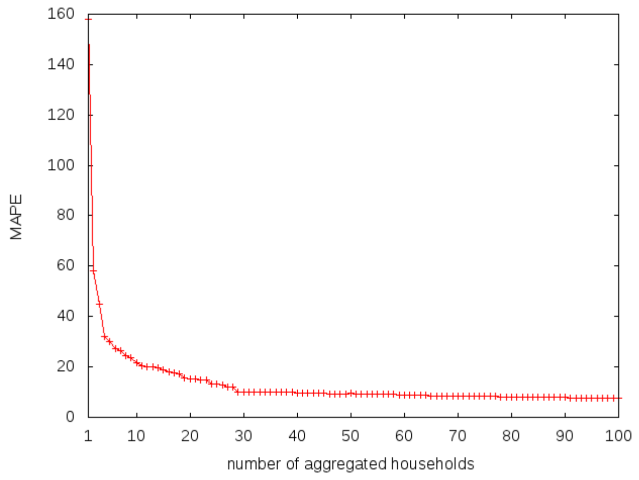

Fig. 3. The MAPE when forecasting the power consumption for the next half hour when using a varying number of aggregated households.

(CRT). That is, if one lets t be a large enough product of small mutually coprime numbers $t_1, t_2, \ldots, t_m$ then we have the well-known ring isomorphism

$$R_t \to R_{t_1} \times R_{t_2} \times \ldots \times R_{t_m} : g(X) \mapsto (g(X) \bmod t_1, \ldots, g(X) \bmod t_m).$$

Instead of evaluating our GMDH network directly in R_t, we can work in each of the R_{t_i}'s separately, simply by reducing things modulo t_i. The outcomes can then be combined very efficiently in order to end up in R_t again. As a consequence it suffices to carry out the FV scheme using the much smaller plaintext spaces R_{t_i}, although one needs to do it for each i separately. For the above example, the 13 mutually coprime numbers 269, 271, 277, 281, 283, 285, 286, 287, 289, 293, 307, 311 and 313 multiply together to

$$t = 95059483533087812461171515276210 \approx 2^{106.229},$$

which indeed satisfies the bound from (3). Thus it suffices to work with R_{269}, R_{271}, …, R_{313}.

6 Prediction Approach for the Smart Grid

6.1 Prediction Model: Apartment Complexes

It is known that it is intrinsically difficult to make accurate short-term predictions based on data from one household when using an artificial neural network [37], and the same volatile behaviour is to be expected when following a GMDH approach. In order to confirm this we designed and trained for each value of $n = 1, \ldots, 100$ a GMDH network that predicts the energy consumption during the next half hour for n aggregated households. This was done along the design criteria (and using the data set) described in Section 6.2 below. The observed prediction qualities, expressed in terms of the mean average percentage error (MAPE), are given in Figure 3. One sees that the results for one household are particularly bad, showing a MAPE of over 158 percent. However, the results start to improve significantly when using aggregated measurements of 10 households: here the MAPE is slightly above 20 percent, while it drops to 7 percent for $n = 100$. These observations are well in line with the ones for ANNs [37]. Due to this volatile nature we decided to aim for *aggregated* prediction, albeit for a *low* number of households, at a scale where the security issues mentioned in Section 2 remain at hand. More precisely, we chose $n = 10$. This use-case matches small apartment complexes in rural areas and the aggregation could be performed locally inside the apartment.

The cryptographic setting we have in mind is that these aggregated measurements are homomorphically encrypted and sent to an untrusted third party. This third party can be seen as just another service user on the smart grid network. He/she has received the concrete parameters (such as the coefficients b_{ijk}) of a trained GMDH network from the party who wants to know the consumption prediction (e.g. the electricity supplier or the network operator). Using both the encrypted inputs $x_1, x_2, \ldots, x_{n_0}$ and the network parameters the untrusted third party can compute the encrypted forecast and forward this to the final party, who is able to decrypt using the cryptographic key corresponding to the one installed in the smart meter.

6.2 Design of the Network

As explained in Section 3 the exact layout of our GMDH network is determined during a learning phase, for which we need access to some real smart meter data. We used the data that was collected through the Irish smart metering electricity customer behaviour trials [5] which ran in 2009 and 2010 with over 5,000 Irish homes and businesses participating. The data consists of electricity consumed during 30 minute intervals (in kW). Per household there are $25,728$ electricity measurements for a total of 536 days. We use the measurements of the first year as training data and the remaining half year to validate and measure how good the network is performing.

An important balancing act is to find a network layout that minimizes the number of layers (and therefore the multiplicative depth of the prediction algorithm) while at the same time preserving a reasonable prediction accuracy, preferably comparable to [37]. Through some trial and error the simplest GMDH network we found to meet these requirements consists of $r = 3$ hidden layers with $n_1 = 8$, $n_2 = 4$ and $n_3 = 2$ nodes, respectively. As input layer a set of $n_0 = 51$ nodes is used, where 48 nodes represent the half hour measurements that were made during the previous 24 hours. The remaining 3 inputs correspond to the temperature, the month, and the day of the week. The single output node $\nu_{4,1}$ then returns the predicted electricity consumption for the next half hour.

Let $\tilde{f} : \mathbf{R}^{51} \to \mathbf{R}$ denote the function that we want to approximate, for which a set of m input-output pairs

$$((x_{i1}, x_{i2}, \ldots, x_{in_0}), y_i^{\text{actual}})_{i=1,\ldots,m}, \qquad \text{with } y_i^{\text{actual}} = \tilde{f}(x_{i1}, x_{i2}, \ldots, x_{in_0}),$$

is given through the Irish data set. As explained in Section 3 these are used to inductively determine the coefficients b_{ijk}, while at the same time selecting the best performing nodes. Assuming that layer $i - 1$ was dealt with, for node ν_{ij} this is done by minimizing the quantity

$$\text{MSE}\left((f_{ij}(x_{11}, \ldots, x_{1n_0}), \ldots, f_{ij}(x_{m1}, \ldots, x_{mn_0})) , (y_1^{\text{actual}}, \ldots, y_m^{\text{actual}}) \right),$$

where $f_{ij} : \mathbf{R}^{51} \to \mathbf{R}$ denotes the function obtained from the network by temporarily considering ν_{ij} as an output node. The minimization can be done using standard linear regression. The useful feature of this approach is that one can apply L2-regularization and kill two birds with one stone. On the one hand regularization helps to avoid the *overfitting* problem, while on the other hand it allows to control the magnitude of the b_{ijk}'s. In this way one can achieve that ν_{ij} is a quadratic polynomial function with small coefficients and a reasonable MSE. We would like to point out that while we use MSE in the learning phase, the quality of the eventually resulting GMDH network is measured in terms of MAPE, in order to allow for a meaningful comparison with the forecasting results reported upon in the scientific literature.

As outlined in Section 5 we carry out fixed-point arithmetic using balanced ternary expansions, rather than binary expansions. To represent the input values $x_1, x_2, \ldots, x_{n_0}$ we use 1 fractional digit and, since the maximal data value is 27.265, at most 4 integral digits. The coefficients b_{ijk} are represented using 1 integral and 8 fractional digits. With these choices we attain basically the same average MAPE around 21 percent as in the floating point setting: a further increase of the precision does not give any significant improvement, although it gradually makes the fixed-point MSE converge to the floating-point one.

6.3 Benchmark Results

In order to assess the practical performance and verify the correctness of our selected parameters we implemented the privacy-preserving homomorphic forecasting approach as introduced in this paper. Our implementation (which will be made publicly avaiable soon) uses the FV-NFLlib software library [7] which implements the FV homomorphic encryption scheme which in turn uses the NFLlib software library (as described in [28] and released at [8]) for computing polynomial arithmetic. Our presented benchmark figures are obtained when running the implementation on an average laptop equipped with an Intel Core i5-3427U CPU (running at 1.80GHz).

Let us recall and summarize the exact forecasting setting and the parameters we selected for the implementation. It is our goal to predict the energy consumption for the next half hour of an apartment complex of 10 households while not revealing any energy consumption information to the party computing

Table 1. The time (in ms) to compute the various basic (homomorphic) operations for our selected parameters.

op	enc	dec	key gen	add	mul	scalar mul
ms	2.1	5.8	77	0.1	33	29

on this data using the GMDH approach as outlined in Section 3. Inherent to this approach we expect a MAPE which is slightly over 20 percent (see Section 6.1). In order to work efficiently with real numbers we use the fixed-point representation with the parameters as outlined in Section 5, using the CRT approach for decomposing plaintext space. We use the FV scheme for the homomorphic computation with the parameters as presented in Section 4. Hence, we target a security level of 80 bits and use the ring $R_{2^{186}} = \mathbf{Z}_{2^{186}}[X]/(X^{2^{12}}+1)$ along with a standard deviation of 102. This means a ciphertext size of 186kB. Recall that the coefficients b_{ijk} are not being encrypted, which limits the noise growth when carrying out scalar multiplications.

As outlined in Section 6.2 the layout of our network consists of an input layer of 51 nodes, three hidden layers of 8, 4 and 2 nodes respectively and a single output node. Remember that when building a new layer the learning algorithm excludes nodes corresponding to node pairs from the previous layer. So not all nodes of the resulting GMDH network affect on the final output and thus can be ignored during evaluation. Each node performs 8 multiplications out of which 5 are by polynomial coefficients and 5 additions. Since there are at most 15 nodes being evaluated this means computing 120 multiplications (out of which 75 by polynomial coefficients) and 75 additions. Table 1 summarizes the performance cost (expressed in milliseconds) for the various basic building blocks used in our homomorphic prediction algorithm. As can be seen from this table, and this is confirmed by running the entire forecasting algorithm in practice, the average computation of the prediction over 100 aggregated datasets is around 2.5 seconds depending on the node wiring. However, as explained in Section 5, this process has to be repeated 13 times for the CRT approach. In practice, the entire forecasting can be computed in half a minute. Due to the embarrassingly parallel nature of the CRT approach, a parallel implementation can compute this in less than 4 seconds or 2.5 seconds on average.

7 Conclusions and Future Work

We have shown that Ivakhnenko's group method of data handling from the 1970s is very suitable for homomorphic computation. This seems to be a better method with respect to the applicability to implement prediction homomorphically compared to the related artificial neural network based approaches in this cryptographic setting. We have studied this prediction approach in the setting of enhancing the privacy of the consumer for forecasting in the smart grid. Our privacy-preserving implementation of this approach to homomorphically forecast for 10 households shows is that this can be computed in less than four seconds for parallel and in half a minute for a sequential implementation.

We would like to point out that this approach has applications beyond the scope of just the smart grid. Other areas which need reliable prediction algorithms but work with privacy sensitive data can directly benefit as well. Examples include computing on financial data or biometric data.

References

1. A. Ahmad, M. Hassan, M. Abdullah, H. Rahman, F. Hussin, H. Abdullah, and R. Saidur. A review on applications of ANN and SVM for building electrical energy consumption forecasting. *Renewable and Sustainable Energy Reviews*, 33:102 – 109, 2014.
2. M. Albrecht. Complexity estimates for solving LWE. `https://bitbucket.org/malb/lwe-estimator/raw/HEAD/estimator.py`, 2000–2004.
3. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *J. Mathematical Cryptology*, 9(3):169–203, 2015.
4. J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In M. Stam, editor, *Cryptography and Coding 2013*, volume 8308 of *LNCS*, pages 45–64. Springer, 2013.
5. Commission for Energy Regulation. Electricity smart metering customer behaviour trials (CBT) findings report. Technical Report CER11080a, 2011. `http://www.cer.ie/docs/000340/cer11080(a)(i).pdf`.
6. A. Costache, N. P. Smart, S. Vivek, and A. Waller. Fixed point arithmetic in SHE schemes. In *Selected Areas in Cryptography – SAC 2016*, LNCS. Springer, 2016.
7. CryptoExperts. FV-NFLlib. `https://github.com/CryptoExperts/FV-NFLlib`, 2016.
8. CryptoExperts, INP ENSEEIHT, and Quarkslab. NFLlib. `https://github.com/quarkslab/NFLlib`, 2016.
9. Department of Energy & Climate Change. Smart metering implementation programme. Technical Report Third Annual Report on the Roll-out of Smart Meters, 2014. `https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/384190/smip_smart_metering_annual_report_2014.pdf`.
10. Department of Energy and Climate Change. Smart metering implementation programme – data access and privacy. `https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/43043/4933-data-access-privacy-con-doc-smart-meter.pdf`.
11. N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Manual for using homomorphic encryption for bioinformatics. Technical report, Technical report MSR-TR-2015-87, Microsoft Research, 2015.
12. N. Dowlin, R. Gilad-Bachrach, K. Laine, K. E. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In M. Balcan and K. Q. Weinberger, editors, *International Conference on Machine Learning*, volume 48, pages 201–210. JMLR.org, 2016.
13. Z. Erkin and G. Tsudik. Private computation of spatial and temporal power consumption with smart meters. In F. Bao, P. Samarati, and J. Zhou, editors, *ACNS*, volume 7341 of *LNCS*, pages 561–577. Springer, 2012.

14. European Commission. Commission recommendation of 9 March 2012 on preparations for the roll-out of smart metering systems. Official Journal of the European Union `http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32012H0148`, March 2012.
15. European Commission. Benchmarking smart metering deployment in the EU-27 with a focus on electricity. Technical Report 365, June 2014. `http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52014DC0356&from=EN`.
16. J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.
17. B. g. Koo, S. W. Lee, W. Kim, and J. H. Park. Comparative study of short-term electric load forecasting. In *Conference on Intelligent Systems, Modelling and Simulation*, pages 463–467, Jan 2014.
18. F. D. Garcia and B. Jacobs. Privacy-friendly energy-metering via homomorphic encryption. In J. Cuéllar, J. Lopez, G. Barthe, and A. Pretschner, editors, *STM*, volume 6710 of *LNCS*, pages 226–238. Springer, 2011.
19. C. Gentry. Fully homomorphic encryption using ideal lattices. In *ACM Symposium on Theory of Computing – STOC 2009*, STOC '09, pages 169–178. ACM, 2009.
20. G. W. Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, 1992.
21. L. Hernandez, C. Baladron, J. M. Aguiar, B. Carro, A. J. Sanchez-Esguevillas, J. Lloret, and J. Massana. A survey on electric power demand forecasting: Future trends in smart grids, microgrids and smart buildings. *IEEE Communications Surveys Tutorials*, 16(3):1460–1495, 2014.
22. A. Ivakhnenko. Heuristic self-organization in problems of engineering cybernetics. *Automatica*, 6(2):207 – 219, 1970.
23. M. Jawurek, F. Kerschbaum, and G. Danezis. Privacy technologies for smart grids - a survey of options. Technical Report MSR-TR-2012-119, November 2012. `http://research.microsoft.com/apps/pubs/default.aspx?id=178055`.
24. K. Kursawe, G. Danezis, and M. Kohlweiss. Privacy-friendly aggregation for the smart-grid. In S. Fischer-Hübner and N. Hopper, editors, *Privacy Enhancing Technologies – PETS*, volume 6794 of *LNCS*, pages 175–191. Springer, 2011.
25. F. Li, B. Luo, and P. Liu. Secure information aggregation for smart grids using homomorphic encryption. In *Smart Grid Comm.*, pages 327–332. IEEE, 2010.
26. R. Livni, S. Shalev-Shwartz, and O. Shamir. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems*, pages 855–863, 2014.
27. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May 2010.
28. C. A. Melchor, J. Barrier, S. Guelton, A. Guinet, M. Killijian, and T. Lepoint. NFLlib: NTT-based fast lattice library. In K. Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 341–356. Springer, 2016.
29. A. Molina-Markham, P. J. Shenoy, K. Fu, E. Cecchet, and D. E. Irwin. Private memoirs of a smart meter. In A. G. Ruzzelli, editor, *Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 61–66. ACM, 2010.
30. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.
31. Recommendation to the European Commission. Essential regulatory requirements and recommendations for data handling, data safety, and consumer protection.

Technical Report version 1.0, 2011. `https://ec.europa.eu/energy/sites/ener/files/documents/Recommendations%20regulatory%20requirements%20v1.pdf`.

32. A. Rial and G. Danezis. Privacy-preserving smart metering. In H. Reimer, N. Pohlmann, and W. Schneider, editors, *Securing Electronic Business Processes, Highlights of the Information Security Solutions Europe 2012 Conference*, pages 105–115. Springer, 2012.
33. R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.
34. Smart Grid Coordination Group. Smart grid information security. `http://ec.europa.eu/energy/sites/ener/files/documents/xpert_group1_security.pdf`, November 2012.
35. D. Srinivasan. Energy demand prediction using GMDH networks. *Neurocomputing*, 72(1):625–629, 2008.
36. The Smart Grid Interoperability Panel – Smart Grid Cybersecurity Committee. Guidelines for smart grid cybersecurity: Volume 1 - smart grid cybersecurity strategy, architecture, and high-level requirements. Technical Report NISTIR 7628 Revision 1, September 2014. `http://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.7628r1.pdf`.
37. A. Veit, C. Goebel, R. Tidke, C. Doblander, and H.-A. Jacobsen. Household electricity demand forecasting: benchmarking state-of-the-art methods. In *Conference on future energy systems*, pages 233–234. ACM, 2014.
38. P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. E. Lauter, and M. Naehrig. Crypto-nets: Neural networks over encrypted data. *CoRR*, abs/1412.6181, 2014.