INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Extracting Mel-Frequency and Bark-Frequency Cepstral Coefficients from Encrypted Signals

Patricia Thaine, Gerald Penn

Department of Computer Science, University of Toronto, Canada
`pthaine@cs.toronto.edu, gpenn@cs.toronto.edu`

Abstract

We describe a method for extracting Mel-Frequency and Bark-Frequency Cepstral Coefficient from an encrypted signal without having to decrypt any intermediate values. To do so, we introduce a novel approach for approximating the value of logarithms given encrypted input data. This method works over any interval for which logarithms are defined and bounded.

Extracting spectral features from encrypted signals is the first step towards achieving secure end-to-end automatic speech recognition over encrypted data. We experimentally determine the appropriate precision thresholds to support accurate WER for ASR over the TIMIT dataset.

Index Terms: automatic speech recognition, cryptography, security, privacy, homomorphic encryption, feature extraction

1. Introduction

Perhaps the most fundamental operation for speech processing, image processing, and digital signal processing in general is the Fourier transform. Being able to perform the Fourier transform in a privacy-preserving manner is key to the creation of privacy-preserving algorithms in these domains. Speech signals in particular contain some of the most sensitive information that we produce, both because of their content (what was said) and because of the way that the content was produced (who was saying it). The benefits of having privacy-preserving speech processing algorithms are three-fold. First, of course, there is preserving the privacy of users who send inputs to speech processing systems. Second, they would revolutionize the accuracy of systems which require a lot of varied and sensitive training data (e.g., automatic speech recognition systems). Third, the less information that a company stores or processes in the clear, the less likely they are to leak customer data as a result of getting hacked.

Related Work. [1] propose a related method for approximating natural logarithms in the context of image brightness/contrast filtering. Whereas their method determines what we shall call an accuracy parameter, ϕ, by reverse-engineering their Taylor series expansion of the logarithm from a given acceptable error bound, however, ours is determined experimentally from an analysis (in the clear) of word error rates (WERs), and uses a novel table-lookup strategy [2] rather than a Taylor series.

[3] proposed a secure keyword recognition algorithm which makes use of HMMs. In this protocol, Alice has a speech signal, which she splits into T frames $\mathbf{x}_i \in \mathbb{R}^d$ for $i = 1, 2, ..., T$, where d is the dimension of the feature vector used (MFCCs of $d = 39$ are used). Bob has Δ HMMs, each trained on one keyword. Though ingenious, [3]'s secure keyword spotter solves a problem in which a client sends an encrypted speech sample to a server for secure keyword search with the client's cooperation. The client's device must itself perform the feature extraction calculations and, in the protocol proposed, also calculate intermediate values (namely, logarithms) in a cooperative manner by means of oblivious transfer and secure two-party computation. An approach that would be more likely to be in demand is one in which a server itself is able to store both the encrypted speech data and a secure algorithm that can perform homomorphically encrypted computations on the encrypted data without relying on the client. This brings up the issue of secure feature extraction from an encrypted speech signal. To our knowledge, the issue of how to conduct secure feature extraction on an encrypted speech signal has thus far been ignored by the community. Instead, it has been assumed that Alice has access to the recording and is able to extract features (usually MFCCs) herself, encrypt them, and send the encrypted features to Bob.

Contributions. We describe how to extract speech features, specifically Bark-Frequency Cepstral Coefficients (BFCCs) and Mel-Frequency Cepstral Coefficients (MFCCs), from an encrypted signal. In doing so, we also describe an approach for approximating the value of a logarithm given encrypted input data, without needing to decrypt any intermediate values before obtaining the function's output. Furthermore, we study how the floating point precision of speech signals, MFCCs, and BFCCs affect the WER of 10 different speech recognition systems.

2. Background

2.1. Homomorphic Encryption

Homomorphic encryption (HE) schemes allow for computations to be performed on encrypted data without needing to decrypt the data. Figure 1 illustrates a typical example of how homomorphic encryption can be used to process a user's data. For this work, we use Brakerski-Fan-Vercauteren (BFV) fully homomorphic encryption [4, 5], with improvements to encryption and homomorphic multiplication of [6]. This scheme is implemented in the PALISADE Lattice Cryptography Library[1]. But our proposal can be implemented using any HE scheme that allows for multiple addition and component-wise multiplication operations in the encrypted domain.

2.2. Notation and Scheme Overview

We will be using the same notation as in [5], but as we provide only a brief overview of the homomorphic encryption scheme in this paper, the specific optimizations introduced in [5, 6] will not be discussed. Using homomorphic encryption, we are able to perform linear (i.e., multiplication, addition, or both) and,

[1]`https://git.njit.edu/palisade/PALISADE`

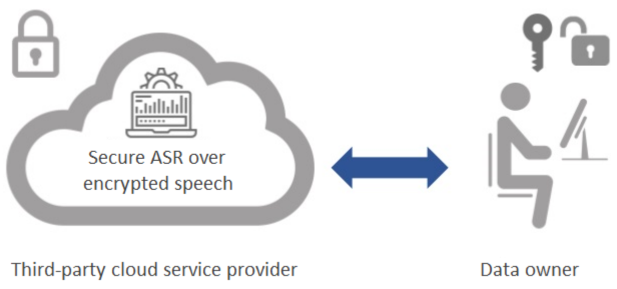
Secure ASR over encrypted speech
Third-party cloud service provider
Data owner

Figure 1: *Using homomorphic encryption, a data owner is able to encrypt their speech sequences, send them to the cloud for processing, then receive and decrypt the results without anybody else (not even the service provider) ever having the ability to decrypt the input or output data.*

depending on the encryption scheme, polynomial operations on encrypted data. We can neither divide a ciphertext, nor exponentiate using an encrypted exponent. However, we can separately keep track of a numerator and a corresponding denominator. For clarity, we shall refer to the encrypted version of values $*$ as $E(*)$, component-wise addition as $E(*) + E(*)$, and component-wise multiplication as $E(*) * E(*)$. A more detailed description of the BFV scheme follows:

Let $R = \mathbb{Z}[x]/(g(x))$ be an integer ring, where $g(x) \in \mathbb{Z}[x]$ is a monic irreducible polynomial of degree d. Bold lowercase letters denote elements of R and their coefficients will be denoted by indices (e.g., $\mathbf{a} = \sum_{i=0}^{d-1} a_i \cdot x^i$). $\mathbb{Z}_q$ denotes the set of integers $(-q/2, q/2]$, where $q > 1$ is a power of 2. The secret key is called $\mathtt{sk} = (1, \mathbf{s})$, where $\mathbf{s} \leftarrow R^2$. The public key is called $\mathtt{pk} = ([-(\mathbf{a} \cdot \mathbf{s} + \mathbf{e})]_q, \mathbf{a})$, where $\mathbf{a} \leftarrow R_q$, $\mathbf{e} \leftarrow \chi$. The plaintext space is taken as R_p for some integer modulus $p > 1$. Let $\Delta \equiv \lfloor q/p \rfloor$ and $r_p(q) \equiv q \mod p$, so that $q = \Delta \cdot p + r_p(q)$.

We use the following operations:

- $\mathtt{Encrypt}(\mathtt{pk},\mathbf{m})$: message $m \in R_p$, $\mathbf{p_0} = \mathtt{pk}[0]$, $\mathbf{p_1} = \mathtt{pk}[1]$, $\mathbf{u} \leftarrow R_2$, $\mathbf{e_1}, \mathbf{e_2} \leftarrow \chi$:
$$\mathtt{ct} = \Big([\mathbf{p_0} \cdot \mathbf{u} + \mathbf{e_1} + \boldsymbol{\Delta} \cdot \mathbf{m}]_q, [\mathbf{p_1} \cdot \mathbf{u} + \mathbf{e_2}]_q\Big)$$
- $\mathtt{Decrypt}(\mathtt{sk}, \mathtt{ct})$: $\mathbf{s} = \mathtt{sk}$, $\mathbf{c_0} = \mathtt{ct}[0]$, $\mathbf{c_1} = \mathtt{ct}[1]$.
$$\left[\left\lfloor \frac{\mathbf{p} \cdot [\mathbf{c_0} + \mathbf{c_1} \cdot \mathbf{s}]_q}{q} \right\rceil\right]_p$$
- $\mathtt{Add}(\mathtt{ct}_1, \mathtt{ct}_2)$: $([\mathtt{ct}_1[0] + \mathtt{ct}_2[0]]_q, [\mathtt{ct}_1[1] + \mathtt{ct}_2[1]]_q)$
- $\mathtt{Mul}(\mathtt{ct}_1, \mathtt{ct}_2)$: For this paper, we use component-wise multiplication, a simplified description of which is: $([\mathtt{ct}_1[0] \cdot \mathtt{ct}_2[0]]_q, [\mathtt{ct}_1[1] \cdot \mathtt{ct}_2[1]]_q)$. We omit the details for obtaining this result [5].

2.3. Setting

Correctness. An appropriate multiplicative depth at the parameters-generation step of our method for correctness to be guaranteed. Error resulting from ciphertext multiplication is reduced using the scale-invariant method [4, 5].

Security. The BFV scheme has semantic security [7], which means that "whatever an eavesdropper can compute about the cleartext given the cyphertext, he can also compute without the cyphertext" [8]. A malicious adversary will neither be able to learn anything about a user's inputs nor about the intermediate and output values of the algorithm.

To calculate the BFCCs of an encrypted signal we set the BFV scheme's parameters as follows:

- Plaintext Modulus $= 14942209$,
- $\sigma = 3.2$,
- Root Hermite Factor $= 1.006$,
- $m = 32768$,
- Ciphertext Modulus $= 2037035672919629476175426 7584669403052717435612598398430688180935286 49268397111195699118081$.

For our experiments, we chose parameters that give over a 128-bit security level the values presented in [7]. This means that it would take over 2^{128} computations to crack the decryption key. We use an Intel Core i-7-8650U CPU @ 1.90GHz and a 16GB RAM for the computations.

Integrity. The BFV scheme itself does not guarantee the integrity of the results that a service provider sends back to a user and our algorithm does not ensure that a server will run the code that they say they will.

2.4. Encoding Variables

The very first step for converting an algorithm into HE-friendly form is to make the data amenable to HE analysis. For the BFV scheme, this includes transforming floating point numbers into integers by computing an approximate rational representation for them. This can be done by multiplying the floating point numbers with a pre-specified power of 10 and then rounding or truncating to the nearest integer [9]. We use truncation. We based our encoding on the method suggested in [10]: choose the number of decimal places to be retained based on a desired level of accuracy ϕ, then multiply the data by 10^{ϕ} and round to the nearest integer. Though straightforward, there had not yet been tests of which values of ϕ would lead to decent accuracies for various tasks. These are shown in Figure 1 and Figure 2 for ASR performed on TIMIT.

3. Mel- and Bark-Frequency Cepstral Coefficients

3.1. BFCCs vs MFCCs

BFCCs and MFCCs produce approximately the same performance in speech recognition tasks [11]. We verify this independently (see Figure 2). We choose to use the Bark scale rather than the Mel scale given that it is easier to compute under the limitations presented by homomorphic encryption schemes:non-polynomial operations are very computationally expensive to perform and we want to limit the quantity required. While the Bark scale can be computed using the Traunmüller equation [12], the Mel scale requires one to calculate the logarithm of variables. If desired, the logarithm operation of the Mel scale can be approximated through the use of the method presented in Section 3.4 or by using the secure two-party computation method presented in [3]. However, these are both comparatively computationally inefficient options. We shall, from now on, only discuss MFCCs with the purpose of comparing the WERs of systems which them vs. BFCCs as inputs.

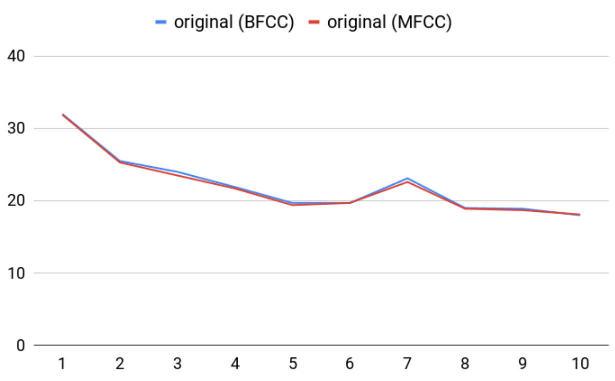

Figure 2: *Comparison of the WER of 10 types of Automatic Speech Recognition Systems trained and tested on the TIMIT dataset using BFCCs vs MFCCs (original precisions).*

3.2. Real Discrete Fourier Transform

The operations for performing the Real Discrete Fourier Transform (RDFT) can be found in [13]. We adapt them for our task:

$$\begin{aligned} E(x_1(n)) = \Big[\sum_{k=0}^{N-1} E(\text{round}(x(k) * 10^{\phi})) \\ * \text{ round}(\cos\left(\frac{2\pi nk}{N}\right) * 100\Big] \end{aligned} \tag{1}$$

$$\begin{aligned} E(x_0(m)) = \Big[\sum_{k=0}^{N-1} E(\text{round}(x(k) * 10^{\phi})) \\ * \text{ round}(\sin\left(\frac{2\pi mk}{N}\right) * 100))\Big], \end{aligned} \tag{2}$$

where N is the number of samples, $x(\cdot)$ is the vector representing the audio signal, and

$$n = 0, 1, 2, ..., N_1$$
$$N_1 = \begin{cases} \frac{N}{2}, & \text{if } N \text{ is even} \\ \frac{N-1}{2}, & \text{if } N \text{ is odd} \end{cases} \tag{3}$$
$$m = 1, 2, 3, ..., N_0$$
$$N_0 = \begin{cases} \frac{N}{2} - 1, & \text{if } N \text{ is even} \\ \frac{N-1}{2}, & \text{if } N \text{ is odd} \end{cases} \tag{4}$$
$$E(\text{RDFT}) = E(x_1(n))^2 + E(x_0(m))^2.$$

The value of $\phi = 2$ for the sine and cosine multipliers 10^{ϕ} was chosen by experimentally choosing the minimal value ϕ that had acceptable accuracy relative to the actual unencrypted RDFT values (not shown here because of space considerations). Note that sines and cosines themselves, and thus multiplication by 10^{ϕ}, can be calculated in plaintext prior to encryption.

Using the parameters listed in Section 2.3, it takes 46523.5 ms to calculate the RDFTs of 100 frames, given a sample rate of 16, a frame length of 25, and a shift length of 10.

3.3. Bark Scale

We use [12]'s Bark scale equation, adapted for our task, with an empirically tested ϕ (see Figure 5).

$$\begin{aligned} &\text{Critical band rate (bark)} = \\ &\frac{26.81 * f * 10^2 - ((0.53 * 10^2) * (1960 + f))}{(1960 + f) * 10^2} \end{aligned} \tag{5}$$

Computing the Bark scale necessitates keeping track of both a numerator and a denominator, since we cannot perform division in the encrypted domain. We will dispense with the divisor next.

Using the parameters in Section 2.3, it takes 315.26 ms to run the Bark scale on the frequencies from the previous step.

3.4. Logarithm

To most accurately calculate logarithms in the encrypted domain, we will use a lookup-table-based method requiring $\log_2(n) + 2$ multiplications, where n is the size of the lookup table. We adapt this method from [2]. The idea is that, given an encrypted value $E(x)$, a vector of values $\mathbf{v} = \langle v_1, v_2, ..., v_n, 0\rangle$ and a corresponding lookup table vector, $\mathbf{l} = \langle \log_{10}(v_1), \log_{10}(v_2), ..., \log_{10}(v_n), 0\rangle$, it is possible to calculate which $E(v_i)$ is equal to $E(x)$ without ever compromising the security of $E(x)$. We can thus extract $E(\log_{10}(x))$. A thorough description of this method is beyond the scope of this paper. This method works on any interval for which the logarithm is defined and range-bounded.

With it, we can calculate the logarithm of both the numerator and the denominator resulting from the previous step. Since division corresponds to subtraction in the log domain, we can perform encrypted subtraction to reduce the representation to a single number, with negligible cost.

With the parameters we have selected, the size of one full vector that's most tractable for the logarithm calculation is at most $256 + 1$. We can, of course, use multiple vectors of this size in order to expand the precision of the logarithms that we are calculating. This should be selected according to the expected range of values as well as the output precision best suited for a given task. Using the parameters listed in Section 2.3, it takes 143005 ms (resp. 346 s) to calculate the logarithm of a value when using a lookup table of size $128 + 1$ (resp. $256 + 1$). Although this is far from the efficiency are we used to when performing calculations in the unencrypted domain, these computations are highly parallelizable. The tables themselves can be created in a compilation step with a method that has linear-time online complexity, but tables can be very large — the experiments in Section 4 requested logarithms for roughly 9×10^6 values distributed over the interval $(2 \times 10^{-1}, 2.13 \times 10^{11})$. Efficient representations of these tables will be an important topic for future research.

3.5. Discrete Cosine Transform

To calculate the Discrete Cosine Transform, we simply use Equation 1, but with $E(x(k))$ now being the log of the Bark-Scaled RDFT of the encrypted signal.

4. Numerical Analysis

Since we are only able to perform operations on integers when using the BFV homomorphic encryption scheme, we conduct three experiments to determine how much the word error rate of automatic speech recognition systems trained and tested on the TIMIT database is affected by (1) signal inputs at various precisions when using MFCCs (see Figure 3), (2) MFCCs at various precisions (see Figure 4) and (3) using BFCCs with a Bark scale at various precisions (see Figure 5). We use Kaldi[2] for these experiments. The systems in turn are:

1. mono: a HMM-GMM monophone model with deltas;
2. tri1: Deltas + Delta-Deltas Training & Decoding (T&D);

[2]http://kaldi-asr.org

3. tri2: linear discriminant analysis (LDA) + maximum likelihood linear transform (MLLT) T&D;
4. tri3: LDA + MLLT + speaker adaptive training (SAT) T&D;
5. SGMM2: Subspace-Gaussian Mixture Model (SGMM2) T&D;
6. Maximum Mutual Information (MMI) + SGMM2 T&D;
7. tri4_nnet: DNN[3] Hybrid T&D;
8. dnn4_pretrain-dbn_dnn: DNN[4] Hybrid T&D;
9. dnn4_pretrain-dbn_dnn_smbr: DNN[4] Hybrid T&D, with state-level minimum Bayes risk (sMBR) training; and
10. combine2: DNN[3] and SGMM system combination.

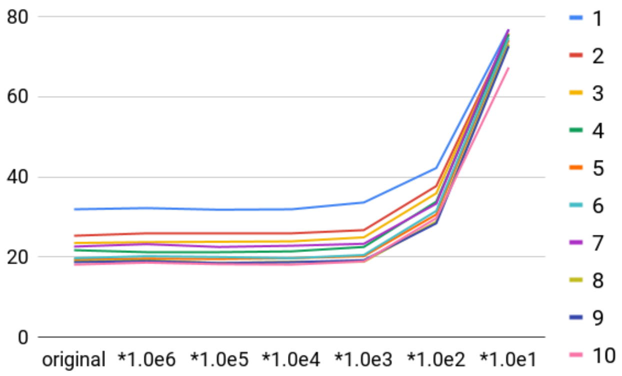

Figure 3: *WER of 10 types of Automatic Speech Recognition Systems trained and tested on the TIMIT dataset with input signals approximated at various precisions ($*1.0e\phi$) and converted into MFCCs.* ***(1)*** *mono;* ***(2)*** *tri1;* ***(3)*** *tri2;* ***(4)*** *tri3;* ***(5)*** *SGMM2;* ***(6)*** *MMI + SGMM2;* ***(7)*** *tri4_nnet;* ***(8)*** *dnn4_pretrain-dbn_dnn;* ***(9)*** *dnn4_pretrain-dbn_dnn_smbr;* ***(10)*** *combine2.*

Figure 3 shows us that $\phi = 3$ is the most sensible value to choose for the precision of the input variables. Figure 4 shows us that using the original float representation and truncating the value of the MFCC features to an integer ($\phi = 0$) both give very similar WERs for MFCCs.

5. Conclusion and Future Work

We have described a method for extracting Bark-Frequency Cepstral Coefficients from an encrypted signal without having to decrypt any intermediate values. In doing so, we have also experimented with a novel approach for approximating the value of logarithms, given encrypted input data, without needing to decrypt any intermediate values before obtaining the function's output. We have tested how realistic our approach would be if the resulting coefficients were to be used as inputs to ten different kinds of automatic speech recognition systems.

Note that, given the RDFT, it would be easy to calculate jitter, which measures whether a gesture is sustained over a short period of time and can be used to identify certain pathological voices [14]. This can be done using the equation:

$$\text{Jitter(absolute)} = \frac{1}{N-1}\sum_{i=1}^{N-1} |T_i - T_{i+1}|,$$

[3]http://kaldi-asr.org/doc/dnn2.html
[4]http://kaldi-asr.org/doc/dnn1.html

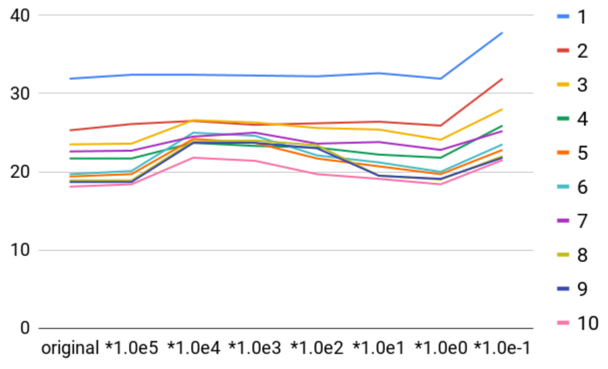

Figure 4: *WER of 10 types of Automatic Speech Recognition Systems trained and tested on the TIMIT dataset with MFCCs approximated at various precisions ($*1.0e\phi$).* ***(1)*** *mono;* ***(2)*** *tri1;* ***(3)*** *tri2;* ***(4)*** *tri3;* ***(5)*** *SGMM2;* ***(6)*** *MMI + SGMM2;* ***(7)*** *tri4_nnet;* ***(8)*** *dnn4_pretrain-dbn_dnn;* ***(9)*** *dnn4_pretrain-dbn_dnn_smbr;* ***(10)*** *combine2.*

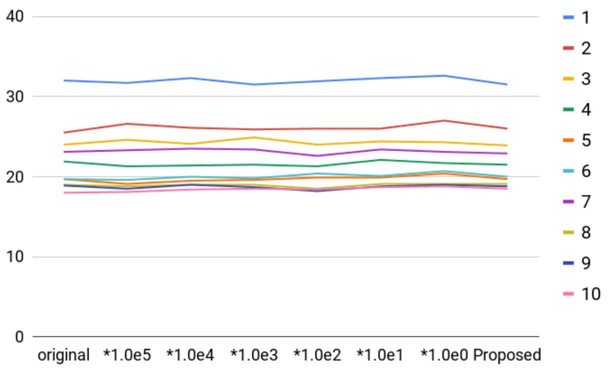

Figure 5: *WER of 10 types of Automatic Speech Recognition Systems trained and tested on the TIMIT dataset using BFCCs, with only the Bark scale approximated at various precisions ($*1.0e\phi$). The "proposed" label refers to the approximation from Equation 5.* ***(1)*** *mono;* ***(2)*** *tri1;* ***(3)*** *tri2;* ***(4)*** *tri3;* ***(5)*** *SGMM2;* ***(6)*** *MMI + SGMM2;* ***(7)*** *tri4_nnet;* ***(8)*** *dnn4_pretrain-dbn_dnn;* ***(9)*** *dnn4_pretrain-dbn_dnn_smbr;* ***(10)*** *combine2.*

where T_i are the extracted F_0 period lengths and N is the number of extracted F_0 periods [15].

Shimmer measures whether the loudness of a sound is sustained over a short period of time. Just like jitter, it can be used to identify pathological voices. We can use the equation:

$$\text{Shimmer(dB)} = \frac{1}{N-1}\sum_{i=1}^{N-1} |20\log(A_{i+1}/A_i)|.$$

These can be useful for privacy-preserving healthcare applications or privacy-preserving speaker recognition, which we intend to address more thoroughly in future work.

6. Acknowledgements

We are grateful to Sean Robertson for his feedback on the experiments. This work was supported by an RBC Graduate Fellowship and the BRAIN Ontario Research Fund.

7. References

[1] T. Shortell and A. Shokoufandeh, "Secure signal processing using fully homomorphic encryption," in *Advanced Concepts for Intelligent Vision Systems*. Springer, 2015, pp. 93–104.

[2] P. Thaine, S. Gorbunov, and G. Penn, "Efficient evaluation of activation functions over encrypted data," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019.

[3] M. A. Pathak, S. Rane, W. Sun, and B. Raj, "Privacy preserving probabilistic inference with hidden markov models." in *ICASSP*, 2011, pp. 5868–5871.

[4] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Advances in cryptology–crypto 2012*. Springer, 2012, pp. 868–886.

[5] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption." *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012.

[6] S. Halevi, Y. Polyakov, and V. Shoup, "An Improved RNS Variant of the BFV Homomorphic Encryption Scheme," http://eprint.iacr.org/2018/117, Intl. Assoc. Cryptologic Research, Tech. Rep., 2018.

[7] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, J. Hoffstein, K. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan, "Homomorphic encryption security standard," HomomorphicEncryption.org, Cambridge MA, Tech. Rep., March 2018.

[8] G. Shafi and S. Micali, "Probabilistic encryption," *Journal of computer and system sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[9] T. Graepel, K. Lauter, and M. Naehrig, "Ml confidential: Machine learning on encrypted data," in *International Conference on Information Security and Cryptology*. Springer, 2012, pp. 1–21.

[10] L. J. Aslett, P. M. Esperança, and C. C. Holmes, "Encrypted statistical machine learning: new privacy preserving methods," *arXiv preprint arXiv:1508.06845*, 2015.

[11] B. J. Shannon and K. K. Paliwal, "A comparative study of filter bank spacing for speech recognition," in *Microelectronic engineering research conference*, vol. 41, 2003.

[12] H. Traunmüller, "Analytical expressions for the tonotopic sensory scale," *The Journal of the Acoustical Society of America*, vol. 88, no. 1, pp. 97–100, 1990.

[13] O. Ersoy, "Real discrete fourier transform," *IEEE transactions on acoustics, speech, and signal processing*, vol. 33, no. 4, pp. 880–882, 1985.

[14] P. Perrot and G. Chollet, "Helping the forensic research institute of the french gendarmerie to identify a suspect in the presence of voice disguise or voice forgery," in *Forensic Speaker Recognition*. Springer, 2012, pp. 469–503.

[15] M. Farrús, J. Hernando, and P. Ejarque, "Jitter and shimmer measurements for speaker recognition," in *Eighth Annual Conference of the International Speech Communication Association*, 2007.