Scalable, Practical VoIP Teleconferencing with End-to-End Homomorphic Encryption

Kurt Rohloff, *Member, IEEE,* David Bruce Cousins, *Senior Member, IEEE,*
Daniel Sumorok, *Member, IEEE*

Abstract—We present an approach to scalable, secure Voice over IP (VoIP) teleconferencing on commodity mobile devices and data networks with end-to-end Homomorphic Encryption (HE). We assume an honest-but-curious threat model where an adversary, despite observing all communications between all teleconference participants and having full access to teleconferencing servers, is unable to obtain unencrypted data and subsequently listen to the conversation. Prior secure VoIP teleconferencing services have relied on 1) all teleconferencing clients to maintain point-to-point encrypted links with other clients or 2) a teleconferencing server which can access and manipulate VoIP streams unencrypted. Our approach mixes VoIP data streams at a single VoIP teleconferencing server only while encrypted and data streams are never decrypted at the teleconferencing server. Our approach is based on a previously known encryption scheme we implement. Innovation in our approach comes from a novel efficient VoIP encoding scheme to greatly reduce the circuit depth needed to support homomorphic mixing of the encrypted VoIP data, parameterization for relatively low bandwidth usage and implementation and integration of our designs into an existing open-source VoIP infrastructure. We experimentally evaluate our end-to-end encrypted VoIP teleconferencing application running on commodity iPhones, mixing at the VoIP servers on the lowest-cost Amazon AWS cloud server instances and communicating on commercial data networks with commercial cellular data and commodity 802.11n wireless access points.

- *K. Rohloff is with the Cybersecurity Research Center in the College of Computing Sciences, New Jersey Institute of Technology, Newark, NJ, 07102.*
  E-mail: rohloff@njit.edu
- *D. Cousins and D. Sumorok are with Raytheon BBN Technologies, Cambridge, MA, 02138.*
  E-mail: {dcousins,dsumorok}@bbn.com

Sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. FA8750-11-C-0098. The views expressed are those of the authors and do not necessarily reflect the official policy or position of the Department of Defense or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited.)

1 Introduction

Teleconferencing is an important aspect of modern professional life that supports long-distance commerce and collaboration. With the increased prevalence and reliance on teleconferencing technologies, there is an increased need to provide secure, scalable teleconferencing technologies. Voice-over-IP (VoIP) teleconferencing is becoming increasingly important due to the prevalence of low-cost portable computing device and easy, economical data network access. There is a perceived risk that compromised data networks or VoIP servers could be used by adversaries to steal and leak sensitive information communicated through VoIP in teleconferences.

Secure VoIP teleconferencing has been partially served through point-to-point communication links which are made secure with point-to-point encryption technologies [1], [2]. This approach does not scale. When there are more than a handful of participants in a teleconference call, $O(n^2)$ point-to-point encryption links are needed, leading to bandwidth issues which degrades the quality of user experiences. To address the scaling limitation, voice signals could be sent to a centralized teleconferencing server that mixes VoIP signals to support interactive teleconferencing and the mixed signals are returned to the participants.

Up to now, VoIP signals on a teleconference server needed to be mixed in the clear, meaning that VoIP signals need to be unencrypted at the server. VoIP teleconferencing servers are often hosted in a semi-secure environment, such as by commodity cloud providers such as Amazon AWS [1] or Microsoft Azure [2]. This creates opportunity

1. https://aws.amazon.com/
2. https://azure.microsoft.com/

for adversaries to snoop on VoIP data such that if the server is compromised, voice data could leak to adversaries. As such, teleconferencing solutions have been vulnerable to man-in-the-middle attacks of various types [3].

Taken together, there is a need for a VoIP teleconferencing capability with end-to-end encryption where VoIP signals are never accessible in the clear except by clients with decryption keys. We address these limitation to provide a teleconferencing approach where data is mixed while encrypted. This approach prevents an adversary from obtaining the communications of teleconference participants even if the adversary observes all communication and data manipulations performed by the teleconference server.

To use the cryptosystem, teleconferencing clients encode their voice samples with an additive encoding scheme, encrypt their encoded voice data under an additive homomorphic encryption scheme, send their encrypted voice samples to a VoIP teleconferencing server. The VoIP teleconferencing server performs an encrypted signal mixing operation on the encrypted VoIP signals. In our scheme, this encrypted mixing operation consists of homomorphic addition on the encrypted voice signals. The teleconference server sends the result of the encrypted mixing back to the clients. The clients then decrypt, decode and play back the result. Our scheme relies on the pre-sharing of a common private key for our additive homomorphic encryption scheme.

The encoding operation converts raw VoIP signal into a data representation which is natively encrypted by the additive homomorphic encryption scheme. Our homomorphic scheme natively encrypts the integer coefficients of a polynomial, and the encode operation converts raw floating-point VoIP signals into the integer coefficients of a polynomial. The decode operation is the inverse of the encode operation.

In modern VoIP teleconferencing schemes, the mixing operation merges VoIP signals. In particular, a mixing operation takes the encoded VoIP signals, each representing a single client speaking, and outputs a single signal that encodes all of the clients' voices. Different VoIP encoding schemes use different encoding approaches.

Innovation in our approach comes from:

- A novel efficient VoIP encoding scheme that greatly reduces the circuit depth needed to support homomorphic mixing of the encrypted VoIP data.
- A homomorphic mixing operation that solely uses homomorphic additions on encrypted data.
- An additive homomorphic encryption that is a simplification of a prior well-known scheme.
- Parameterizations to provide security, high voice quality and relatively low bandwidth usage.
- Implementation and integration of our designs into an existing open-source VoIP infrastructure.
- Experimentation of our prototype on low-cost mobile and cloud computing infrastructure and commercial data networks.

The cryptographic basis of our approach is an efficient implementation of an additive homomorphic encryption scheme. The cryptosystem builds from recent advances in practical homomorphic encryption [4]. In particular, our approach is based on recent advances in lattice-based homomorphic encryption and is a simplification of the LTV scheme [5] which is itself an NTRU-like cryptosystem [6].

Many modern VoIP communication systems rely on logarithmic encoding schemes which dedicate more bits of the encoding to the most significant bits of the encoded data. Our approach relies on an additive encoding scheme which greatly reduces the circuit depth of the VoIP mixing operation.

We integrate our voice data encoding and encryption implementations with the existing Mumble/Murmur open-source VoIP teleconferencing framework [3]. We experimentally evaluate our end-to-end encrypted VoIP teleconferencing application running on commodity iOS-based clients such as iPod Touches and iPhones, mixing on the lowest-cost Amazon AWS cloud server instances and commercial cellular data networks or commodity 802.11n wireless access points.

A high-level overview of our approach is mentioned in [7]. The underlying cryptosystem without implementation is presented in [5] and an early implementation of this cryptosystem without application to VoIP is in [4].

The paper is organized as follows. Section 2 discusses related teleconferencing approaches and applied homomorphic encryption efforts. Section 3 discusses our security model, assumptions we make of the adversary and design goals for our scalable VoIP teleconferencing capability with end-to-end encryption. Section 4 discusses the overall

3. http://www.mumble.info/

architecture of our end-to-end encrypted VoIP teleconferencing capability. In Section 5 we describe our simplification of the LTV-based homomorphic cryptosystem [5] that we build from. We introduce our custom VoIP coding scheme in Section 6 and the homomorphic mixing operation in Section 7. In Section 8 we describe integrating the cryptosystem with the Mumble open-source VoIP framework. Section 9 discusses our engineering trade-offs in parameterizing the coding scheme, the cryptosystem and the VoIP mixer. In Section 10 we discuss experimentation. We present a security analysis in Section 11. We conclude the paper with a discussion of our insights and future work in Section 12.

2 Related Work

Advances in secure VoIP technologies have focused on providing security for data in transit [2], addressing identity and key management [8], among many other security issues [9].

There has been some recent work to address these prior challenges, including using Secure Multi-Party Computation (SMC) [10] to mix encrypted VoIP signals over multiple servers, with the restriction that each client needs to trust a server. This prior SMC-based approach is experimentally demonstrated by integrated with the Mumble/Murmur software, and our team shared integration and implementation advice with the authors of [10].

The basis of our approach builds on lattice-based Homomorphic Encryption (HE) which has previously been considered inefficient for broad practical use [11]. Solutions to HE runtime challenges have been explored through several means, including by improving the theoretical efficiency of the underlying scheme [12], [13], and by developing more efficient implementations of these schemes [14]–[18]. Despite these advances in HE schemes and implementations, there are few applications of these technologies.

Besides the runtime challenges of HE designs, there is limited experience with data structures and representations of homomorphic encrypted data [19]. Modern lattice-based HE provides a very different computation model as compared to the more well understood RAM compute models. The porting of familiar data structures and algorithms (such as for VoIP mixing) for efficient use with HE is an ongoing challenge, especially for highly efficient encrypted execution of these algorithms over the encrypted input data. We expand beyond single-bit-per-ciphertext encodings by placing entire VoIP data frames into each ciphertext. These designs are in some sense much simpler than existing current industry practice, such as the mu-law encoders [20] which are common in modern VoIP systems.

3 Threat Model and Functional Goals

To frame the functional and security goals of our end-to-end encrypted VoIP teleconferencing concept, we present our high-level design considerations:

1) **Encryption Work Factor**: We should provide an encryption work factor adequate to protect teleconferences. For our lattice-based scheme, current security estimates indicate that a root Hermite factor $\delta < 1.007$ generally provides 80 bits of security and an adequate work factor [21].
2) **Server Compromise**: The confidentiality of the data should be preserved even if the server is fully compromised by an honest-but-curious adversary who observes all internal server operations.
3) **Latency**: There should usable end-to-end latency of less than 150 ms, which is considered acceptable in practice [22].
4) **Sound Quality**: There should be acceptable sound quality, preferably with full-duplex so users could listen while simultaneously speaking.
5) **Scalability**: The teleconferencing capability should scale to support more than two participants without noticeable degradation in sound quality or latency.
6) **Bandwidth Usage**: The end-to-end encrypted VoIP teleconferencing capability should ideally require less than an order of magnitude ciphertext expansion over unencrypted or even current AES-encrypted approaches.
7) **Wide Geographic Area**: The capability should operate over a wide geographic area, ideally trans-continental if not intercontinental, without an unacceptable degradation in sound quality or latency.

4 End-to-End Encrypted VoIP Teleconferencing Architecture

We now describe the high-level architecture of our end-to-end encrypted VoIP teleconferencing capa-

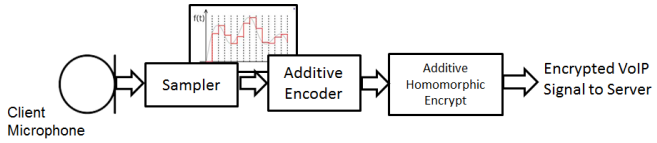

Fig. 1: VoIP Encoding

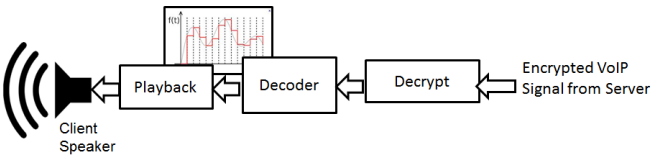

Fig. 2: VoIP Decoding

bility. Multiple clients, possibly from multiple access points, connect over a commodity data network, such as the public Internet, to a VoIP server. Each client samples a stream of voice data, encodes the voice data stream and encrypts it. The client sends its stream of encrypted VoIP data to the VoIP server. The VoIP server receives the encrypted VoIP streams from multiple clients. The VoIP server performs *homomorphic mixing* operations on the encrypted voice streams and sends the encrypted results back to the clients. The mixed encrypted VoIP data received by each client is then decrypted by the clients, decoded and played back to the clients' users.

Figure 2 shows how the returning stream of encrypted VoIP data is decrypted into plaintext, and the plaintext is decoded into a stream of digitized audio samples. The audio samples are then played back to the user through the devices' speakers.

A common approach to merge digital audio signals in a teleconference session at the server is to add the encoded samples of unencrypted (plaintext) digital audio signals from the client. The function which merges the encoded, but unencrypted, sampled digital audio signals at the teleconference server is called the Mix function. Although this design relies on nontrivial encoding and decoding processes, the Mix function can be a simple, low-depth operation at the teleconference server. This suggests an approach to encrypted VoIP teleconferencing to design a homomorphic encrypted version of the plaintext Mix function. We call this homomorphic encryption version of the Mix function as the EvalMix function, and the design of Mix suggests that EvalMix could be built by solely homomorphically adding encrypted audio signals. Hence, both an additive homomorphic encryption scheme and an additive encoding scheme would be needed for this to approach to work. (We say that an encoding scheme $\mathsf{Encode}()$ is additive if $\mathsf{Encode}(a+b) = \mathsf{Encode}(a) + \mathsf{Encode}(b)$.)

A primary challenge is then to design a homomorphic evaluation function EvalMix such that for two voice samples v_1 and v_2 and their respective encodings $\mathsf{Enc}(pk, v_1)$ and $\mathsf{Enc}(pk, v_2)$, we know that the result of mixing the signals, $\mathsf{Mix}(v_1, v_2)$ is equal to $\mathsf{Dec}(sk, \mathsf{EvalMix}(\mathsf{Enc}(pk, v_1), \mathsf{Enc}(pk, v_2)))$ for an encoding of the digital audio stream into plaintext. We should be able to parameterize the Enc and EvalMix operations so that they are sufficiently secure and resource efficient to make end-to-end encrypted VoIP teleconferencing practical. If $\{\mathsf{Enc}(pk, v_1), \mathsf{Enc}(pk, v_2), \mathsf{Enc}(pk, v_3)\}$ are encrypted audio blocks received by the server at a specific time from three clients, then the server should return to client 1 a ciphertext c_1' that when decrypted equals $\mathsf{Mix}(v_2, v_3)$.

5 Additive Homomorphic Lattice-Based Cryptosystem

Our design builds on a recent efficient FHE design [5] and an implementation [4]. Our approach is a lattice-based cryptographic scheme. Mathematical preliminaries for lattice-based cryptography can be found in [23], and a general survey on the design of lattice-based schemes can be found in [24].

Our HE cryptosystem provides the core public key encryption primitives of Key Generation ($\mathsf{KeyGen} \rightarrow (pk, sk)$), Encryption ($\mathsf{Enc}(pk, \mu) \rightarrow c$) and Decryption ($\mathsf{Dec}(sk, c) \rightarrow \mu$) where pk is a public key, sk is a secret key, μ is a plaintext and c is a ciphertext. Our scheme is defined to be an additive homomorphic encryption scheme because it provides an Evaluation Addition operation ($\mathsf{EvalAdd}(c_1, c_2) \rightarrow c_3$) where $\mathsf{Dec}(\mathsf{Enc}(a+b)) = \mathsf{Dec}(\mathsf{EvalAdd}(\mathsf{Enc}(a), \mathsf{Enc}(b)))$ if the scheme is properly parameterized to satisfy its correctness constraints.

Our primary simplification of the scheme in [4], [5] is that we remove the ability to support Evaluation Multiplication (EvalMult). Because we simplify the scheme in this way, the correctness and security constraints for the general scheme in [4], [5] hold for our simplified scheme, and we are able to choose smaller concrete parameters while maintaining security and correctness.

5.1 Plaintext and Ciphertext Representation

Concretely, we represent our plaintext and ciphertext as vectors of unsigned integers. We use power-

of-2 cyclotomics, meaning that plaintext and ciphertext vectors represent the coefficients of integer mod p polynomials of degree 2^x [23]. Almost all of the operations we need to support on the plaintext and ciphertext are linear transforms, and by restricting ourselves to power-of-2 dimensions, we greatly reduce the implementation complexity required in our cryptosystem. Reduced implementation complexity translates directly into more efficient implementation. Current HE implementations designed for non-power-of-2 cyclotomics can support more general capabilities, but we intentionally choose to focus on a simpler scheme for improved runtime and reduced ciphertext expansion for limited-depth applications.

For n a power of 2, we define the ring $R = \mathbb{Z}[x]/(x^n+1)$ (i.e., integer polynomials modulo x^n+1) where, for any positive integer q, define the ciphertext space $R_q = R/qR$ (i.e., integer polynomials modulo x^n+1, with mod-q coefficients). Additional details on mathematical preliminaries can be found in [23]. The plaintext space is R_p for some integer $p \geq 2$, meaning that plaintext are length-n vectors of integers modulus p. Typically p is much smaller than 2^{64} and we typically choose p to be between 2 and 2^{12}. Except for special applications, such as for our secure VoIP application, we often need on the order of several hundred bits to represent q to support deep computations at a reasonable level of security.

5.2 LTV-based Scheme with LSB Encoding

Based on the mathematical preliminaries, we now provide a description of our simplified LTV scheme [5]. Concrete implementation and parameter discussions are given after the mathematical sketches.

In lattice-based encryption schemes, plaintext, ciphertext and other ring elements can be represented in either evaluation or Chinese Remainder Transform (CRT) representation [23]. These representations are equivalent. A ring element can be converted from an evaluation representation to a CRT representation by application of a special Number Theoretic Transform called the Chinese Remainder Transform (CRT) [23]. More explicitly, we can convert a ciphertext from its evaluation representation c_e to its CRT representation c_{crt} computing $c_{crt} = \mathsf{CRT}_q(c_e)$ for appropriately chosen parameterizations of the CRT as in [14]. The inverse CRT (CRT^{-1}) converts ring Elements from the CRT representation back to the evaluation representation. In this case, c_e can be recovered from c_{crt} by applying the inverse CRT transform, $c_e = \mathsf{CRT}^{-1}{}_q(c_{crt})$.

Our simplified LTV encryption scheme is agnostic with respect to the representation of the ciphertext. We describe the operation of this scheme assuming a CRT representation of ciphertext unless otherwise stated. It is advantageous to keep ciphertext in CRT representation because additions and multiplications on ciphertext are performed element-wise in this representation. Otherwise, in evaluation representation, ciphertext multiplication requires an operation similar to a convolution operation to perform.

Our cryptosystem operations are described as follows:

- KeyGen: choose a "short" $f \in R$ such that $f = 1 \bmod p$ and f is invertible modulo q. Concretely, f is a vector of n integers and f is invertible modulo q if and only if each of its mod-q CRT evaluations is nonzero. The "short" elements f can be chosen from discrete Gaussians. E.g., we can let $f = p \cdot f' + 1$ (where 1 is a length-n with all entries set to 1) for some Gaussian-distributed f'. the Gaussian distribution is zero-centered with a distribution parameter denoted as r. We output $sk = f$. The secret key, as defined above, is most efficiently represented in its evaluation representation. Similarly, we choose a "short" $g \in R$, possibly from discrete Gaussians. The g is sampled in it evaluation representation, but then converted to its CRT representation. The CRT representations of f_1^{-1} (modulo q) is the mod-q inverse of f_1. We output $pk = h = g \cdot f^{-1} \bmod q$ in its double-CRT representation.
- Enc($pk = h, \mu \in R_p$): choose a "short" $r \in R$ and a "short" $m \in R$ such that $m = \mu \bmod p$. The vectors r and m are sampled in their evaluation representations, but then converted to their CRT representations. We output $c = p \cdot r \cdot h + m \bmod q$. Concretely, m can be chosen as $m = p \cdot m' + \mu$ for a Gaussian-distributed m' sampled in its evaluation representation, then converted to its CRT representation.
- Dec($sk = f, c \in R_q$): compute $\bar{b} = f \cdot c \bmod q$, and lift it to the integer polynomial $b \in R$ with coefficients in $[-q/2, q/2)$. Output $\mu = b \bmod p$. The key multiplication operation can be performed in either the evaluation or CRT representation.

The scheme supports the additive homomorphic

operation:

$$\mathsf{EvalAdd}(c_0, c_1) = c_0 + c_1 \bmod q$$

The EvalAdd operation can be performed using both evaluation or CRT representations, but our implementation always uses a CRT representation so the scheme supports expected forward compatibility with additional encrypted operations supported on the ciphertext.

5.3 Security of Scheme

The proofs of security in [5] are directly applicable to the simpler additive homomorphic scheme we discuss here. In particular, Lemma 3.6 in [5] demonstrates the security of this scheme, and we do not provide a detailed security proofs. In Section 9 we leverage further parameter selection and noise management optimizations from [21], [25]–[27] which do not affect the security proofs given in [5].

6 VoIP Signal Mixer and Additive Encoding

We consider the sampled and digitized representation of VoIP audio signals as being a sequence of y-bit integers $[v_1, v_2, \ldots]$. Define $v^i = [v_1^i, v_2^i, \ldots, v_m^i]$ to represent a block of samples, $\mathsf{Encode}([v_1^i, \ldots, v_m^i]) = [z_1^i, \ldots, z_n^i] = z^i$ where z^i is a mod p plaintext and $\mathsf{Enc}([z_1^i, \ldots, z_n^i]) = [c_1^i, \ldots, c_n^i] = c^i$ where c^i is a mod q ciphertext.

We define the Mix function as the function which takes unencrypted encoded sampled audio signals as input and outputs a single audio signal in which all input signals are overlayed on one another and can be heard. The most common (and simplest) mixing functions in deployed VoIP systems is a sample-adder, and we use this as our model plaintext mixing operation which we define here.

Definition 1. Mix*: We define the* Mix *operation as follows where* $j = \lfloor t/2 \rfloor$*:*

$$\mathsf{Mix}(z^1, \ldots, z^t) = \begin{cases} z^1, & \text{if } t = 1 \\ \mathsf{Mix}(c^1, \ldots, z^j) + \\ \quad \mathsf{Mix}(z^{j+1}, \ldots, z^t), & \text{otherwise} \end{cases}$$

Our goal is to show that for our Encode and Decode functions, when used in conjunction with the Mix, the following property holds for proper parameterizations and arbitrary samples $v^1, v^2, \ldots, v^t$ where t is a variable that represents the number of active speakers:

$$\mathsf{Decode}\left(\mathsf{Mix}\left(\mathsf{Encode}(v^1), \ldots, \mathsf{Encode}(v^t)\right)\right) = v^1 + v^2 + \cdots + v^t \quad (1)$$

When designing the Encode and Decode operations, wrap-around in the plaintext signal should be avoided when the Mix function is applied to encoded (but not encrypted) plaintext data. Specifically, for the plaintext samples $z_j^1, z_j^2, \ldots$, if $\sum_i z_j^i > p$ for some j, then the encoded plaintext data output by the Mix operation wraps around p. If this wrap-around occurs on the plaintext data, there will be distortion in the resulting mixed audio signal, even if there is no decryption. If this distortion occurs in the plaintext data, then the distortion will also be present in encrypted output signal resulting from applying the EvalMix operation to the encrypted input signal due to the property of Equation 2.

To guarantee there is no distortion due to plaintext wrap-around, we design Encode and Decode to guarantee that Equation 1 always holds when used with our defined EvalMix operation under the assumption that there are no more than t teleconference participants speaking simultaneously. Note that t is a variable which can be changed as the VoIP capability is used, as long as clients agree on a t value to guarantee a common encoding and decoding operation. We design the Encode and Decode to operate with any arbitrarily chosen t. Our vision is that the parameter t will be set for every session. We experimentally evaluate scenarios below for various values of t and where that are more than t active speakers on a session to investigate the effect this variable has on the usability of the scheme.

Definition 2. Encode*: We assume without loss of generality that the input is sampled mod* 2^y*. We define the* Encode *operation as* $[z_1, z_2, \ldots, z_n] := \mathsf{Encode}([v_1, v_2, \ldots, v_m])$ *where*

$$z_i = \sum_{j \in \{kn+i | 0 \le c, kn+i \le m\}} v_j * 2^{k(y+s)} \mod p$$

Note that we use $(y + s)$-bit representations of the data samples in the encoding, even though we assume the samples from the audio source have y-bit representations. This is to add extra padding bits so that when we add encoded samples of up to $t \le 2^s$ speakers, we have extra most-significant bits to avoid $\sum_i z_j^i > p$ for some j. Generally we set t to be a power of 2, so that for a scenario where we support up to 4 active speakers in a session, we set $t = 4$, so $s = 2$.

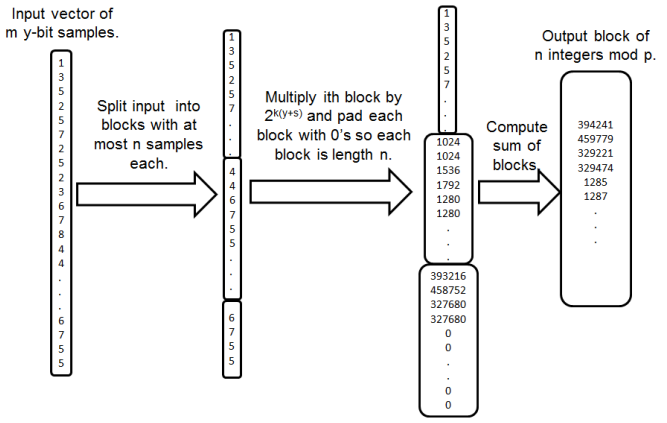

Fig. 3: Encrypted VoIP Encoding

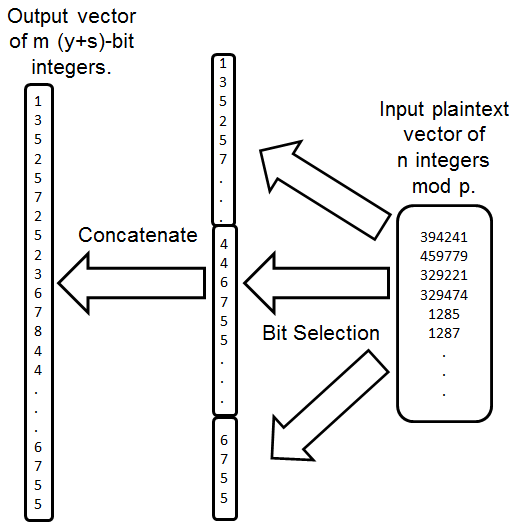

Fig. 4: Encrypted VoIP Decoding

A graphical representation of this scheme can be seen in Figure 3, with the following concrete execution steps:

- We start by splitting the length m VoIP digitized audio sample input $[v_1, v_2, \ldots, v_m]$ into blocks $[v_1, v_2, \ldots, v_n], [v_{n+1}, v_{n+2}, \ldots, v_{2n}], \ldots, [v_{1+kn}, \ldots, v_m]$. Each block has n samples except for the last block which is of length $m \mod n$ if $m \mod n > 0$, and the last block is of length n otherwise. This means that unless m is a multiple of n, the last block of $< n$ samples is the leftover samples which are not fit into the first length-n samples.
- For the ith block, we multiply all samples in the block by $2^{k(y+s)}$ so that in a binary representation, the $k(y+s)$ least significant bits are all zero and the y most significant bits is a left-shift of the original sample, with two padding bits.
- We add all of the samples for each index in the blocks.

Note that we can encode $(y+s)$ integer inputs to the Encode function the way it is formulated. In a binary representation of the encoded data element z_i, the bits in locations $[k(y+s)+1 : k(y+s)+y]$ represents the $(kn+i)$th sample. For the sake of notational simplicity, we say that $z_j^i[a,b]$ represents the ath through bth bits in the j entry of plaintext encoding vector z_i where the least significant bit is the 1st bit. We insert additional binary 00 padding between the binary representation of encoding output so that $z_j^i[k(y+s)+y+1 : k(y+s)+y+s] = [00]$ for all i, j and k. We do this so for four encoded samples $z_i^1, z_i^2, z_i^3, \ldots, z_i^t$, we know that for the summation $z_i' = z_i^1 + z_i^2 + z_i^3 + \cdots + z_i^t$, which is output by the mixing operation, has bits in locations $[k(y+s)+1 : k(y+s)+y+s]$ to hold the potential overflow resulting from summations of the $(kn+i)$th samples. We call these two padding bits which catches the overflow as overflow bits. As such, our encoding scheme induces the constraint that the plaintext modulus p must satisfy the condition that $p \geq 2^{b(y+s)}$ where $b = \lceil m/n \rceil$.

The Decode operation behaves as the inverse of the Encode operation. The Decode operation returns the bits in locations $[k(y+s)+1 : (k+1)(y+s)]$ of z_i as the $(kn+i)$th returned sample. Note that the output of the Decode operation is $\mod 2^{y+s}$, and this requires 2 additional bits. We do this so the output of the decoding can hold the summation of the encoding input without wrapping around.

Definition 3. Decode*: We define the* Decode *operation as follows:* $[v_1, v_2, \ldots, v_m] :=$ Decode$([z_1, z_2, \ldots, z_n])$ *where*

$$v_{kn+i} = z_i[k(y+s)+1 : (k+1)(y+s)].$$

Figure 4 shows how our decoding process could be implemented. On the right hand side of this figure we take the input vector. We make copies of this block and perform bit selection. Similar to the encoding operation, these operations are all highly efficient as they only involve bit selection, which is extremely efficient to implement.

We are now ready to show that Equation 1 holds with the following Theorems 1 and 2. Theorems 1 shows that the encoding scheme is additive and Theorem 2 shows that the composition of the decoding and encoding schemes is additive. We use these properties to prove that the encrypted VoIP system, when used with a homomorphic encryption equivalent of the plaintext Mix operation, does not distort the mixed encrypted signals.

Theorem 1. *Given* $v^1, \ldots, v^t$,

$$\begin{aligned}&\mathsf{Encode}(v^1) + \cdots + \mathsf{Encode}(v^t) \mod p \\ &\quad = \mathsf{Encode}(v^1 + \cdots + v^t \mod 2^{y+s}).\end{aligned}$$

Proof. By definition for $v^h = [v_1^h, v_2^h, \ldots, v_m^h]$, $z^h = [z_1^h, z_2^h, \ldots, z_n^h]$,

$$z_i^h = \sum_{j \in \{kn+i | 0 \le c, kn+i \le m\}} v_j^k * 2^{k(y+s)} \mod p$$

Define $A = \{kn+i | 0 \le c, kn+i \le m\}$ By adding these two equations,

$$\begin{aligned}&\sum_{h \in \{1,\ldots,t\}} z_i^h \mod p \\ &= \sum_{h \in \{1,\ldots,t\}} \left(\sum_{j \in A} v_j^h * 2^{k(y+s)} \right) \mod p \\ &= \sum_{j \in A} \left(\left(\sum_{h \in \{1,\ldots,t\}} v_j^h \right) * 2^{k(y+s)} \right) \mod p\end{aligned}$$

Hence,

$$\begin{aligned}&\mathsf{Encode}(v^1) + \cdots + \mathsf{Encode}(v^t) \mod p \\ &\quad = \mathsf{Encode}(v^1 + \cdots + v^t \mod 2^{y+s}).\end{aligned}$$

□

Theorem 2. *If* $t \le 2^s$, $\mathsf{Decode}(\mathsf{Encode}(v^1) + \cdots + \mathsf{Encode}(v^t) \mod p) = (v^1 + \cdots + v^t \mod 2^{y+s})$.

Proof. From Theorem 1 that $\mathsf{Encode}(v^1) + \cdots + \mathsf{Encode}(v^t) \mod p = \mathsf{Encode}(v^1 + \cdots + v^t \mod 2^{y+s})$ so,

$$\begin{aligned}&\mathsf{Decode}(\mathsf{Encode}(v^1) + \cdots + \mathsf{Encode}(v^t) \mod p) \\ &\quad = \mathsf{Decode}(\mathsf{Encode}(v^1 + \cdots + v^t \mod 2^{y+s}))\end{aligned}$$

If we assign $z' = \mathsf{Encode}(v^1 + \cdots + v^t \mod 2^{y+s})$ and $v' = \mathsf{Decode}(\mathsf{Encode}(v^1 + \cdots + v^t \mod 2^{y+s}))$, then from the definition of the Decode operation, we know that $v'_{kn+i} = z'_i[k(y+s)+1 : (k+1)(y+s)] = \left(\sum_{l \in \{1,\ldots,t\}} v_{kn+i}^l\right) \mod 2^{y+s}$. □

7 Low-Depth Homomorphic Encryption VoIP Signal Mixer

As above, define $v^i = [v_1^i, v_2^i, \ldots, v_m^i]$ to represent a block of samples, $\mathsf{Encode}([v_1^i, \ldots, v_m^i]) = [z_1^i, \ldots, z_n^i] = z^i$ where z^i is a mod p plaintext and $\mathsf{Enc}(pk, [z_1^i, \ldots, z_n^i]) = [c_1^i, \ldots, c_n^i] = c^i$ where c^i is a mod q ciphertext.

We abuse notation and assume without loss of generality that all encryption operations are done with a common public key pk, so we denote $\mathsf{Enc}(pk, z^i)$ as $\mathsf{Enc}(z^i)$. We take a similar notational shortcut with the decryption operation and an assumed common secret key sk such that $\mathsf{Dec}(sk, c^i)$ is denoted for notational simplicity as $\mathsf{Dec}(c^i)$.

To support end-to-end encrypted VoIP teleconferencing, we use our encoding function Encode, decoding function Decode and mixing function Mix with our encryption function Enc and decryption function Dec to define an operation EvalMix that is the homomorphic encrypted equivalent of the plaintext Mix operation. Our goal is to design the EvalMix operation such that the following property holds for proper parameterizations and arbitrary samples $v^1, v^2, \ldots, v^t$ where t is a variable that represents the number of active speakers:

$$\begin{aligned}&\mathsf{Decode}\left(\mathsf{Dec}\left(\mathsf{EvalMix}\left(c^i, \ldots, c^t\right)\right)\right) \\ &\quad = \mathsf{Decode}\left(\mathsf{Mix}(v^1, v^2, \ldots, v^t)\right)\end{aligned} \tag{2}$$

In this equation, $c^i = \mathsf{Enc}\left(\mathsf{Encode}\left(v^i\right)\right)$ represents the encryption of the encoding of the samples v^i. The EvalMix function performs an encrypted homomorphic mixing of these ciphertext, which when decoded and then decrypted, the result is the same as if the sampled data had been mixed without encoding and encryption.

Definition 4. EvalMix*: We define our homomorphic mixing function* EvalMix *as a generalization of our previously defined* EvalAdd *operation where* $j = \lfloor t/2 \rfloor$.

$$\mathsf{EvalMix}(c^1, \ldots, c^t) = \begin{cases} c^1, & \text{if } t = 1 \\ \mathsf{EvalAdd}(\mathsf{EvalMix}(c^1, \ldots, c^j), & \\ \quad \mathsf{EvalMix}(c^{j+1}, \ldots, c^t)), & \text{otherwise} \end{cases}$$

Note that our EvalMix operation can be optimized for parallel execution by evaluating the recursive branches on different threads.

We show in Theorem 4 that Equation 2 holds for our EvalMix operation. To show this proof, we first need to show that if encrypted signals are input to the EvalMix function, then the decrypted output is the summation of the plaintext signals. We then use this property in the proof of Theorem 4 to show if encoded VoIP signals are fed as input to the EvalMix operation, then the decrypted and decoded output is a mix of the VoIP signals.

Theorem 3. *For the input plaintext* $z^1, \ldots, z^t$ *and plaintext modulus* p *assuming proper parameterization*

to enable correct decryption after repeated application of the EvalAdd *operation,*

$$\begin{aligned}&\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(z^1),\ldots,\mathsf{Enc}(z^t)))\\&\quad=(z^1+\cdots+z^t \mod p).\end{aligned}$$

Proof. We demonstrate this with a generalized proof by induction on t.

For the base case, $t = 1$, $\mathsf{EvalMix}(c^1) = c^1 \mod q$ by definition. By the correctness of decryption of our encryption scheme, $\mathsf{Dec}(\mathsf{Enc}(z^1)) = z^1 \mod p$, so $\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(z^1))) = (\mathsf{Dec}(\mathsf{Enc}(z^1)) \mod p) = z^1$.

Assume that for $t' < t$,

$$\begin{aligned}&\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(z^1),\ldots,\mathsf{Enc}(z^{t'}))\\&\quad=(z^1+\cdots+z^{t'} \mod p).\end{aligned}$$

From the definition of EvalMix, we know that

$$\begin{aligned}&\mathsf{EvalMix}(c^1,\ldots,c^t)\\&=\mathsf{EvalAdd}(\mathsf{EvalMix}(c^1,\ldots,c^j),\mathsf{EvalMix}(c^{j+1},\ldots,c^t)).\end{aligned}$$

By substitution,

$$\begin{aligned}&\mathsf{EvalMix}(\mathsf{Enc}(z^1),\ldots,\mathsf{Enc}(z^t))\\&=\mathsf{EvalAdd}(\mathsf{EvalMix}(\mathsf{Enc}(z^1),\ldots,\mathsf{Enc}(z^j)),\\&\qquad\mathsf{EvalMix}(\mathsf{Enc}(z^{j+1}),\ldots,\mathsf{Enc}(z^t))).\end{aligned}$$

Therefore,

$$\begin{aligned}&\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(z^1),\ldots,\mathsf{Enc}(z^t))\\&=\mathsf{Dec}(\mathsf{EvalAdd}(\mathsf{EvalMix}(\mathsf{Enc}(z^1),\ldots,\mathsf{Enc}(z^j)),\\&\qquad\mathsf{EvalMix}(\mathsf{Enc}(z^{j+1}),\ldots,\mathsf{Enc}(z^t)))).\end{aligned}$$

From the correctness of the EvalAdd operation,

$$\begin{aligned}&\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(z^1),\ldots,\mathsf{Enc}(z^t))\\&=\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(z^1),\ldots,\mathsf{Enc}(z^j)))\\&\qquad+\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(z^{j+1}),\ldots,\mathsf{Enc}(z^t)))\\&\qquad\mod p.\end{aligned}$$

From the induction hypothesis,

$$\begin{aligned}&\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(z^1),\ldots,\mathsf{Enc}(z^t))\\&\quad=(z^1+\cdots+z^j+z^{j+1}+\cdots+z^t \mod p).\end{aligned}$$

□

Theorem 4. *For $t \le 2^s$,*

$$\begin{aligned}&\mathsf{Decode}(\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(\mathsf{Encode}(v^1)),\\&\qquad\ldots,\mathsf{Enc}(\mathsf{Encode}(v^t)))))\\&\quad=\mathsf{Mix}(v^1,v^2,\ldots,v^t)\end{aligned}\tag{3}$$

Proof. We know from Theorem 3 that

$$\begin{aligned}&\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(z^1),\ldots,\mathsf{Enc}(z^t)))\\&\quad=z^1+\cdots+z^t \mod p.\end{aligned}$$

By substitution,

$$\begin{aligned}&\mathsf{Decode}(\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(\mathsf{Encode}(v^1)),\\&\qquad\ldots,\mathsf{Enc}(\mathsf{Encode}(v^t)))))\\&\quad=\mathsf{Encode}(v^1)+\cdots+\mathsf{Encode}(v^t) \mod p).\end{aligned}$$

We know from Theorem 1 that $(\mathsf{Encode}(v^1)+\cdots+\mathsf{Encode}(v^t) \mod p) = \mathsf{Encode}(v^1+\cdots+v^t \mod 2^{y+s})$ so,

$$\begin{aligned}&\mathsf{Decode}(\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(\mathsf{Encode}(v^1)),\\&\qquad\ldots,\mathsf{Enc}(\mathsf{Encode}(v^t)))))\\&\quad=\mathsf{Encode}(v^1+\cdots+v^t \mod 2^{y+s}).\end{aligned}$$

We know from Theorem 2 that for $t \le 2^s$, $\mathsf{Decode}(\mathsf{Encode}(v^1)+\cdots+\mathsf{Encode}(v^t) \mod p) = (v^1+\cdots+v^t \mod 2^{y+s})$, so,

$$\begin{aligned}&\mathsf{Decode}(\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(\mathsf{Encode}(v^1)),\\&\qquad\ldots,\mathsf{Enc}(\mathsf{Encode}(v^t)))))\\&\quad=(v^1+\cdots+v^t \mod 2^{y+s}).\end{aligned}$$

By definition, $(v^1+\cdots+v^t \mod 2^{y+s}) = \mathsf{Mix}(v^1,v^2,\ldots,v^t)$ so if $t \le 2^s$,

$$\begin{aligned}&\mathsf{Decode}(\mathsf{Dec}(\mathsf{EvalMix}(\mathsf{Enc}(\mathsf{Encode}(v^1)),\\&\qquad\ldots,\mathsf{Enc}(\mathsf{Encode}(v^t)))))\\&\quad=\mathsf{Mix}(v^1,v^2,\ldots,v^t)\end{aligned}\quad.$$

□

Using Theorem 4 we can now provide an encrypted VoIP mixing capability so that a teleconference participant can listen to multiple mixed audio signals from other participants on a single channel. An important feature of our scheme is that it provides full-duplex communication, meaning that participants can transmit while they receive mixed signals.

From a practical usability perspective, a teleconference participant would not want their voice fed back to them through the mixing operation. Even with a small return lag as low as 10 ms, the feedback signal can become disorienting to speakers, even if such a lag from other participants is generally unnoticeable during normal conversation. We therefore want to support multiple mixing operations simultaneously, so that client voice streams $(v^1,\ldots,v^t)$, the ith client receives the mixed signal $\sum_{j\in\{1,\ldots,t\}\setminus\{i\}}(v^i)$. The EvalAdd operations in the

EvalMix function are all highly efficient as they only involve splitting vectors, multiplication by two and bitwise concatenation, which are all extremely efficient to implement. This summation for multiple recipients is computationally efficient, relying only on summation operations, and can also be performed in a parallel manner. We discuss the engineering tradeoffs associated with concrete parameter selection in greater depth in Section 9.

8 End-to-End Encrypted VoIP Teleconferencing Prototype

We implemented the additive homomorphic cryptosystem portion of the secure VoIP teleconferencing prototype in the Mathworks Matlab environment because it provides an interpreted computation environment with native support for vector and matrix manipulation for rapid prototyping. We found the Matlab syntax to be a natural fit for the primitive lattice operations needed for our LTV-based cryptosystem design. We used the Matlab coder toolkit [28] to generate an ANSI C version of our implementation. For early experimentation we compiled the generated ANSI C using gcc to run as an executable in a Linux environment for parameter tuning.

Rather than construct a VoIP capability from the ground up, we constructed an end-to-end encrypted VoIP teleconferencing capability by integrating our HE library and the Encode/Decode functionality with an existing open-source VoIP teleconferencing library. We selected the Mumble VoIP library for this integration because Mumble is mature, offers high sound quality and runs on a variety of platforms. We focused on iOS clients because these clients were the most mature open source versions of the Mumble clients.

By integrating with the Mumble library, our end-to-end encrypted VoIP library has the same user interface, usage model and deployment model as the standard Mumble capability. An image of the Graphical User Interface (GUI) of the modified client running on an iPod Touch can be seen in Figure 5 where the client is running in push-to-talk mode. Although we did not make this app publicly available, the modified Mumble software can also be deployed through an app store model, or as binaries which can be loaded onto iOS devices with development tools.

Client integration was relatively straight forward with several notable exceptions to reduce packet drops and improve sound quality:

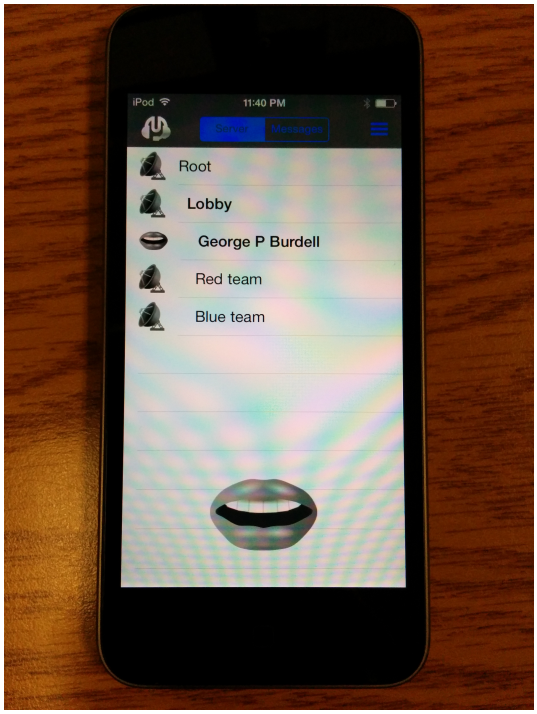

Fig. 5: The Push-To-Talk Client GUI

1) The client application generated voice packets that contained 480 samples at 48 kHz, or 10 ms worth of sound. The sound driver on the client, however, generated slightly larger packets. As a result, the period of the sound packets was slightly larger than 10 ms, and every so often two sound packets were generated back to back. The original server set a 10 ms timer and accepted one packet every 10 ms. We added a small queue at the server so we did not drop packets when we received two packets in less than 10 ms.
2) We generated new frame numbers at the server as opposed to re-using the client frame numbers. The clients correlated the frame numbers with time. This modification reduces time jitter.
3) The encryption and decryption operations for our applications were processor intensive, and were run in batches of several audio packets at once. We moved the encryption and decryption operations to a low priority thread and had a higher priority thread accept and queue new audio packets (both from the network, and from the microphone). This helped prevent a situation where audio packets are dropped because the client was busy decrypting or encrypt-

ing.

4) We improved the efficiency noise sample generation by using an approximate discrete Gaussian operation that mapped 32-bit uniform distributed samples from the client pseudo-random generator to multiple discrete Gaussian samples using a look-up table.

Our modifications, by improving sound quality and efficiency and reducing packet drops, enables increases in audio sampling rates and and lowered encryption lag. As a result, the client is capable of generating 16 bit audio samples at a rate of 48 kHz, which is a standard configuration of the Mumble VoIP client. This throughput provides a sound quality substantially better than PSTN.

9 Parameterization and Adaptation for End-to-End Encrypted VoIP Teleconferencing

We make our initial high-level end-to-end encrypted VoIP teleconferencing capability goals more concrete by choosing parameters for the encoding/decoding and cryptosystem operations so that:
- The VoIP signal data is sampled at a high enough rate that adequately high voice quality is encoded into plaintext.
- The end-to-end VoIP transmission cannot have and end-to-end lag of more than the classical 150ms limit on end-to-end lag [22].
- We target a throughput of not too much larger than would be needed for AES-encrypted operations, meaning a ciphertext expansion of less than 10, or a total bandwidth usage of less than 6.4Mbps which is supportable by most commodity wireless data transmission protocols [29].
- The plaintext is securely encrypted into VoIP ciphertext with sufficiently large work factor preventing unauthorized decryption. For our encryption scheme, this means $\delta \leq 1.007$ for 80 bits of security [21].
- The homomorphic mixing of multiple encrypted VoIP feeds can be successfully decrypted and then decoded. We discuss the case for at most $t = 4$ active speakers at any instance in time can be supported during a session, but this parameter generalizes.
- All these operations need to be run efficiently on commodity hardware, such as iOS and Linux devices with 64 bit processors. This mean that the largest ciphertext moduli has to be less than 2^{64}.

To achieve these more concrete goals in the context of our introduced cryptosystem and encoding and decoding functionality, the parameters we adjust are:

- q, the ciphertext modulus.
- p, the plaintext modulus.
- n, the ring dimension.
- y, the VoIP sample bit depth.
- ϕ, the VoIP sample rate.
- m, the number of samples per VoIP block.
- s, the number of bits of padding in the encoding scheme.

We consequently set our optimization constraints as:

- $q < 2^{64}$, the maximum ciphertext modulus for optimal execution on 64-bit processors.
- $p > 2^{\lceil m/n \rceil (y+2)}$, to guarantee lossless encoding from the definition of the Encode operation in Section 9 for 4 actively speaking clients.
- $q > 4pr\sqrt{n}w$, to guarantee correct decryption where $r = 3$ is a discrete Gaussian noise parameter and $w = 4.5$ is an assurance parameter for the encryption system we use from [4], [5].
- $n > (\log q)/(4 \log \delta)$, to guarantee security [26], [30].
- $s \geq \log_2(t)$, to guarantee that the bits of 0 padding between samples in the encoding and our correctness proofs hold when the maximum number of speakers are talking.

With these optimization constraints we want to:
- Minimize Encode, Enc, EvalMix, Dec and Decode runtime. We find in [4] that the runtimes of the encryption operations are linear in n. Naive algorithms for the runtime of Encode and Decode are also linear in n. We thus attempted to minimize n while satisfying all other constraints.
- Minimize $(qn)/(2^{y+2}m)$, the ciphertext expansion.
- Guarantee throughput $\phi 2^y$ does not cause sufficient packet drops to negatively affect voice quality. We essentially fix ϕ and evaluate voice quality experimentally.

Given the large number of variables, the highly nonlinear optimization constraints for those that we can analyze (such as ciphertext expansion) and the prevalence of experimentally evaluated conditions, such as packet drop rates, voice quality and software runtime, we performed initial experimentation using plaintext operations to find initial data sampling rates and encoding parameters. From a pragmatic standpoint for the VoIP teleconferencing application, we observed experimentally that there

is little practical sense to accommodate more than 4 active speakers. It is very difficult for any one listener to follow a conversation with so many simultaneously speaking participants. We identified experimentally that plaintext mixing with 40ms blocks of 48KHz VoIP data of 5-bit depth provided good sound quality adequate for normal conversation. The native app provides sampled data in groupings of 10ms of data, but we merge 4 blocks of 10ms of data together for an m of 1920 samples per audio block.

We use a ring dimension $n = 1024$ with 1920 samples per block, and we have a resulting plaintext modulus of $p = 2^{14} = 16384$. We then set q to be 1236950597633, which requires 41 bits to represent. These parameters results in a root Hermite factor upper-limit of $\delta = 1.00696$ which is currently believed to be adequately secure and provide at least 80 bits of security [21]. From the audio input (5 bits of 1920 samples) and ciphertext output (41 bits of 1024 samples), we have a resulting ciphertext expansion of roughly 4.37.

With these parameter settings we observed that when running on an iPhone 5s, the encoding and encryption operation took a mean time of 9.2ms and decryption and decoding took 4.6ms. The summation on the VoIP server took 0.5ms. Transport of encrypted VoIP traffic from Cambridge MA to the Northern Virginia Amazon AWS servers took an average of 15ms. This results in a total mean latency of less than 150ms for VoIP traffic. Taken together, these parameters meet our performance goals, although experimental use indicates that latency is on the high side and users who speak quickly would need to speak more slowly than normal to have a conversation, although not so slow as to be unnatural.

10 End-to-End Encrypted VoIP Teleconferencing Experimentation

We experimentally evaluated the performance of the VoIP service deployed in each of the Amazon AWS data centers across the world. We then installed the client software in iPod Touch and iPhone 5s clients. We connected each client to each of the servers through various connection types in the metro area of Boston, a major city in the New England region of the United States. These connections included 802.11n wireless enterprise gateway connected to a high-speed enterprise Internet connection, 4G LTE, 3G and 2G connections over the T-mobile commercial wireless service and an AT&T DSL connection in a rural area in the state of Connecticut.

We measured the upload and download throughput of the connections, the drop rate of VoIP packets routed through the various server locations and the subjective quality of the VoIP teleconference session as defined by the experimenters. The upload and download throughput was measured by Ookla throughput measurement app [31] on the client devices. VoIP drop rates were measured experimentally by instrumenting the VoIP servers to measure drop rates. Voice quality was measured in comparison to PSTN voice quality where "Excellent" means the VoIP conversation was better than PSTN, "Good" means the VoIP conversation was comparable PSTN, "Poor" means the VoIP conversation was worse than PSTN but still usable for communication, and "Unusable" means the connection was useless for communication.

All of the experiments were run over a 2 hour period on a weekday evening using two iPod Touch clients with servers deployed on the Amazon AWS t1.micro instances [32]. The clients were each on independent connections to the Internet at all times, so there was low likelihood of one client contributing substantially to congestion or packet drops for the other client.

Table 1 shows the upload and download throughput observed by each of the clients for each of the connections. Note that the rural DSL service provided better throughput than the 2G connection and better download throughput than the 3G connection.

TABLE 1: Experimentally Measured Data Throughput in Mb/s for Connection Types

Connection Type	Upload Rate Mb/s	Download Rate Mb/s
Enterprise 802.11n	38.22	36.53
4G LTE	35.82	17
3G	6.31	0.43
2G	0.2	0.16
Rural DSL	2.55	0.47

We observed that all of the last-mile wireless connections supported acceptable VoIP teleconference capabilities except for the 2G connections. Over all of the acceptable connections, the lowest upload or download throughput observation was on the 3G download: 0.43Mb/s Because the VoIP download and upload data flows are symmetric, this implies at least a 0.43Mb/s upload and download throughput connection is required to support VoIP teleconferencing using our prototype.

Table 2 shows the packet drop rates observed at each of the servers at the various Amazon AWS locations for the various client connection types. Note that distance between the client and server had only a minor impact on drop rates, while the connection type had a very large impact on drop rates. This implies that the connection could be a bottleneck for the VoIP service.

The subjective VoIP teleconference quality observed through each of the servers at the various Amazon AWS locations for the various client connection types had no observed impact on voice quality. Conversely, we found that connection type had a very large impact on voice quality. Enterprise 802.11n, 4G LTE, 3G all provided adequate voice quality, while 2G and Rural DSL connections provided poor to unusable voice quality.

In addition to our tests of connection-server pairings, we also tested the impact of having more than $t = 2^s$ clients actively speaking at any one time.. For this experiment we established a session with 3 iPod Touch and 4 iPhone 5s clients at various locations on the eastern United States seaboard to a single VoIP server in the Amazon AWS Northern Virginia data center. With these 7 client connections running simultaneously with 4 users speaking simultaneously we were able to hold meaningful conversation among the 4 simultaneous speakers and no voice distortion was observed by the 3 non-speaking client users. With 6 users speaking simultaneously and 1 silent listening client, we experimentally observed that no clipping due to mixing unless the 6 clients are talking so loudly as to nearly be screaming. However, when more than 4 participants attempt to speak simultaneously, the conversation was ineffective not because of limitations in the homomorphic mixing, but due to the inability of participants to mentally track the simultaneous speaking of more than 4 people. As such, limiting the number of speakers to 4 is a reasonable choice for practical teleconferencing use cases.

11 Security Analysis

Although innovation in our prototype is intended to address honest-but-curious adversaries who can observe all server behavior, our design can be further augmented to provide broader security features. For instance, if a client is compromised, the adversary at the compromised client, even with full observation of network traffic from all other clients, will be unable to determine from which other clients specific VoIP signals are coming from if there is at least two other clients. This feature arises from the inability to map mixed signals to their sources based on ciphertext observations. This non-identification feature comes from the feature of the lattice encryption system in that encryptions of two different signals are indistinguishable, and these two encrypted signals are indistinguishable from the encryption of white noise because of the noise added to ciphertext during encryption.

Recent publications have introduced the concept of subfield lattice attacks [33], [34]. Although these attacks are relevant for some variants of the NTRU cryptosystems that rely on the DSPR [5], our parameter settings do not meet the conditions of these attacks.

A possible effective attack on the end-to-end encrypted VoIP teleconferencing activity would be for a compromised client to inject noise signals into the teleconferencing mixer, thus reducing the ability of the client participants to converse with one another. This attack is a vulnerability of general teleconferencing systems. Any participant can inject noise into the conversation. A feasible longer-term approach to reduce the effectiveness of this threat would be to perform homomorphic filtering against noise injection, and possibly provide some functionality for the clients to tell the mixer which other participants they want to listen to. Over the short term, the application of authentication techniques to identify likely adversaries could be effective in addressing these issues.

Another issue is one of key distribution. The keys in our cryptosystem are relatively small: 41*1024 bits which is a little more than 5MB. These keys could easily be distributed through encrypted flat files using existing public key infrastructures, or even using Diffie-Hellman type techniques to generate shared secret keys. Furthermore, our assumption of prior key coordination can be removed by generalizing the scheme to rely on multi-key features of the LTV scheme [5].

12 Discussion and Ongoing Activities

In this paper we present and discuss a scalable approach to VoIP teleconferencing that provides end-to-end encryption. Our assessment is that we met our design goals for practical and secure teleconferencing as discussed in Section 3.

A further aspect of our layered architecture vision is an ability to mix-and-match a computing substrate at the server for increased scalability

TABLE 2: Packet Drop Rates For Various Server Locations and Client Internet Connection Types.

Server Location	Client Location	Enterprise 802.11n	4G LTE	3G	2G	Rural DSL
N. Virginia	Connecticut	0%	10%	10%	66%	33%
Oregon	Connecticut	0%	2%	3%	71%	35%
N. California	Connecticut	0%	7%	8%	67%	34%
Ireland	Connecticut	0%	7%	7%	73%	38%
Singapore	Connecticut	5%	2%	2%	68%	39%
Tokyo	Connecticut	1%	3%	4%	69%	37%
Sydney	Connecticut	5%	3%	3%	67%	34%
Sao Paulo	Connecticut	0.30%	4%	6%	76%	34%

and throughput. Although we only utilize limited-depth Additive Homomorphic Encryption capabilities, our encryption system implementation is a scaled-down version of a Fully Homomorphic Encryption (FHE) system [4]. As such, our teleconference implementation can be generalized to support FHE operations with software modifications and changes in parameter selection to maintain security. The engineering trade-off is that as deeper circuits are computed at the server, end-to-end latency increases.

Prior general implementation efforts [4] shows EvalMult runtimes are currently too large for comfortable interactive conversations if these operations are to be performed at the VoIP server. However, if runtime performance of EvalMult were to improve by only an order of magnitude or two, the latency to execute deeper circuits than the homomorphic mixing would become feasible. This provides a design path for deeper operations on the encrypted data. An initial feasible application would be encrypted noise rejection at the VoIP server by implementing a low pass filter. Low-pass filters, such as weighted moving averages, can be supported in relatively shallow depth circuits with a few multiply operations. This more general design would enable protection against some potentially practical attacks that could be made by an adversary such as noise injection attacks where an adversary inserts noise into a VoIP teleconferencing session to reduce the ability of participants to hear one another.

Even deeper operations, such as voice recognition operations run on encrypted data, would require circuits so deep that current HE implementations would need to use an effective bootstrapping operation to maintain security. Bootstrapping in current FHE implementations takes multiple orders of magnitude longer to perform than EvalMult operations, and multiple breakthroughs in the implementation of bootstrapping, or even completely new and more efficient FHE schemes, will be needed for even deeper circuits to be practical to satisfy VoIP latency constraints. Bootstrapping is likely to be first implemented practically in custom high-performance hardware, such as in a custom FPGA- or ASIC-based co-processor [35]–[37]. When used in conjunction of a variation of a not previously implemented bootstrapping scheme [38], our design offers the possibility for a much more general VoIP teleconferencing capability that incorporates signal detection and noise filtering operations on the encrypted VoIP channels.

Our end-to-end encrypted VoIP capability provides a basis for integration with other hardware-accelerated co-processors due to our use of a layered software services stack to provide high-level homomorphic encryption operations supported by lower-level lattice-based primitive implementations. Part of our longer-term vision is to provide software interfaces in our design for our highly optimized implementations of the basic FHE operations (KeyGen, Enc, EvalAdd, EvalMult, Dec) for both general and specific applications.

ACKNOWLEDGEMENT

The authors wish to acknowledge the support of Chris Peikert in designing and implementing the cryptosystem, advice on the VoIP integration from David Archer, suggestions related to VoIP mixing from abhi shelat and suggestions on parameter security from Yuriy Polyakov.

REFERENCES

[1] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The secure real-time transport protocol (SRTP)," 2004.

[2] P. Zimmermann, A. Johnston, and J. Callas, "ZRTP: Media path key agreement for secure RTP," *draft-zimmermann-avt-zrtp-04 (work in progress)*, 2007.

[3] R. Zhang, X. Wang, R. Farley, X. Yang, and X. Jiang, "On the feasibility of launching the man-in-the-middle attacks on VoIP from remote attackers," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ser. ASIACCS '09. New York, NY, USA: ACM, 2009, pp. 61–69. [Online]. Available: http://doi.acm.org/10.1145/1533057.1533069

[4] K. Rohloff and D. B. Cousins, "A scalable implementation of fully homomorphic encryption built on NTRU," in *Proceedings of the 2nd Workshop on Applied Homomorphic Cryptography (WAHC)*, 2014.

[5] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *STOC*, 2012, pp. 1219–1234.

[6] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Algorithmic Number Theory*, ser. Lecture Notes in Computer Science, J. P. Buhler, Ed. Springer Berlin Heidelberg, 1998, vol. 1423, pp. 267–288. [Online]. Available: http://dx.doi.org/10.1007/BFb0054868

[7] D. W. Archer and K. Rohloff, "Computing with data privacy: Steps toward realization," *IEEE Security & Privacy*, no. 1, pp. 22–29, 2015.

[8] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Advances in CryptologyCRYPTO93*. Springer, 1994, pp. 232–249.

[9] A. D. Keromytis, "Voice-over-IP security: Research and practice," *Security & Privacy, IEEE*, vol. 8, no. 2, pp. 76–78, 2010.

[10] J. Launchbury, D. Archer, T. DuBuisson, and E. Mertens, "Application-scale secure multiparty computation," in *Programming Languages and Systems*, ser. Lecture Notes in Computer Science, Z. Shao, Ed. Springer Berlin Heidelberg, 2014, vol. 8410, pp. 8–26. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-54833-8_2

[11] C. Gentry and S. Halevi, "Implementing Gentry's fully homomorphic encryption scheme," in *Advances in Cryptology, EUROCRYPT 2011*, ser. Lecture Notes in Computer Science, K. Paterson, Ed. Springer Berlin / Heidelberg, 2011, vol. 6632, pp. 129–148.

[12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, p. 13, 2014.

[13] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 75–92.

[14] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Advances in Cryptology–CRYPTO 2012*. Springer, 2012, pp. 850–867.

[15] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, ser. CCSW '11. New York, NY, USA: ACM, 2011, pp. 113–124. [Online]. Available: http://doi.acm.org/10.1145/2046660.2046682

[16] Y. Doröz, Y. Hu, and B. Sunar, "Homomorphic AES evaluation using NTRU." *IACR Cryptology ePrint Archive*, vol. 2014, p. 39, 2014.

[17] C. Gentry and S. Halevi, "HElib," https://github.com/shaih/HElib, 2014.

[18] L. Ducas and D. Micciancio, "FHEW: Bootstrapping homomorphic encryption in less than a second," in *Advances in Cryptology–EUROCRYPT 2015*. Springer, 2015, pp. 617–640.

[19] D. Wu and J. Haven, "Using homomorphic encryption for large scale statistical analysis," 2012.

[20] L. Liu, M. Fukumoto, and S. Saiki, "An improved mu-law proportionate NLMS algorithm," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 3797–3800.

[21] T. Lepoint and M. Naehrig, *A Comparison of the Homomorphic Encryption Schemes FV and YASHE*. Cham: Springer International Publishing, 2014, pp. 318–335. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-06734-6_20

[22] S. Na and S. Yoo, "Allowable propagation delay for VoIP calls of acceptable quality," in *Advanced Internet Services and Applications*. Springer, 2002, pp. 47–55.

[23] V. Lyubashevsky, C. Peikert, and O. Regev, "A toolkit for ring-lwe cryptography." in *EUROCRYPT*, vol. 7881. Springer, 2013, pp. 35–54.

[24] C. Peikert, "A decade of lattice cryptography," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 4, pp. 283–424, 2016.

[25] N. Gama and P. Q. Nguyen, "Predicting lattice reduction," in *EUROCRYPT*, 2008, pp. 31–51.

[26] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," in *CT-RSA*, 2011, pp. 319–339.

[27] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better lattice security estimates," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, vol. 7073. Springer, 2011, pp. 1–20.

[28] MATLAB, *R2012b*. Natick, Massachusetts: The MathWorks Inc., 2012.

[29] E. Perahia and R. Stacey, *Next Generation Wireless LANS: 802.11 n and 802.11 ac*. Cambridge University Press, 2013.

[30] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post Quantum Cryptography*. Springer, February 2009, pp. 147–191.

[31] *Ookla*, 2014. [Online]. Available: https://www.ookla.com/

[32] *Amazon AWS Instance Types*, 2014. [Online]. Available: http://aws.amazon.com/ec2/instance-types/

[33] M. Albrecht, S. Bai, and L. Ducas, "A subfield lattice attack on overstretched NTRU assumptions," *IACR Cryptology ePrint Archive*, vol. 2016, p. 127, 2016.

[34] J. H. Cheon, J. Jeong, and C. Lee, "An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low level encoding of zero," *IACR Cryptology ePrint Archive*, vol. 2016, p. 139, 2016.

[35] X. Cao, C. Moore, M. O'Neill, E. O'Sullivan, and N. Hanley, "Accelerating fully homomorphic encryption over the integers with super-size hardware multiplier and modular reduction." *IACR Cryptology ePrint Archive*, vol. 2013, p. 616, 2013.

[36] D. B. Cousins, J. Golusky, K. Rohloff, and D. Sumorok, "An fpga co-processor implementation of homomorphic encryption," in *High Performance Extreme Computing Conference (HPEC), 2014 IEEE*. IEEE, 2014, pp. 1–6.

[37] E. Öztürk, Y. Doröz, B. Sunar, and E. Savaş, "Accelerating somewhat homomorphic evaluation using FPGAs," Cryptology ePrint Archive, Report 2015/294, Tech. Rep., 2015.

[38] J. Alperin-Sheriff and C. Peikert, "Practical bootstrapping in quasilinear time," in *Advances in Cryptology CRYPTO 2013*, ser. Lecture Notes in Computer Science, R. Canetti and J. Garay, Eds. Springer Berlin Heidelberg, 2013, vol. 8042, pp. 1–20. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40041-4_1