Secure Modular Hashing

Abelino Jiménez, Bhiksha Raj
Carnegie Mellon University
Pittsburgh, PA, USA
{abelinoj, bhikshar}@andrew.cmu.edu

Jose Portelo, Isabel Trancoso
Instituto Superior Técnico
Lisboa, Portugal
{Jose.Portelo, Isabel.Trancoso}@inesc-id.pt

***Abstract*—In many situations, such as in biometric applications, there is need to encrypt and "hide" data, while simultaneously permitting restricted computations on them. We present a method to securely determine the ℓ_2 distance between two signals if they are close enough. This method relies on a locality sensitive hashing scheme based on a secure modular embedding, computed using quantized random projections, being a generalization of previous work in the area. Secure Modular Hashes (SMH) extracted from the signals preserve information about the distance between the signals, hiding other characteristic from the signals. Theoretical properties state that the described scheme provides a mechanism to threshold how much information to reveal, and is also information theoretically secure above this threshold. Finally, experimental results reveal that distances computed from SMH vectors can effectively replace the actual Euclidean distances with minimal degradation.**

I. Introduction

Alice has handed Bob two vectors x_1 and x_2 in encrypted form. She permits him to guess the distance between x_1 and x_2, *but only if they are close enough*. If they are farther than some threshold, she would prefer Bob not to know the distance between the two. Under no circumstance must Bob know the actual vectors themselves.

Situations such as these arise in several situations. The simplest situation one may visualize is when one stores a large set of data on a cloud server. Given a new query instance, they may wish to retrieve the indices of all instances that are close to the query instance, without revealing information about the remaining instances to the server. Another instance where such a requirement is of interest is in the case of secure biometrics: the server retains a template of (features derived from) the biometric. The user wishes to authenticate himself such that the server can assign a confidence, but only if the confidence is high. The server must not learn any additional information about the data. While problems such as these may be solved using partially [1] or fully homomorphic [2] encryption techniques, these are computationally unattractive solutions. The need for distance-computation in the encrypted domain can sometimes increase computational cost by several orders of magnitude [1]; alternate approaches using "garbled" circuits require high levels of parallel processing to even bring to within the realm of practical realizability [3].

A more attractive solution is to achieve this through some sort of map that hides Alice's data, and yet permits Bob to compute the distance between them. Formally, if Alice's data resides in some metric space $\mathcal{M} = (M, d_m)$, we would like to map them to a different space $\mathcal{S} = (S, d_s)$ through some map $f : \mathcal{M} \longrightarrow \mathcal{S}$ such that $d_s(f(x_1), f(x_2)) \sim d_m(x_1, x_2)$ if $d_m(x_1, x_2)$ is less than some threshold r, and $d_s(f(x_1), f(x_2))$ is not related to $d_m(x_1, x_2)$ otherwise. At the same time $f(x_1)$ should not reveal any information about x_1, or even about $f(x_2)$, *i.e.* the mapping must be *information theoretically secure* (ITS).

A solution that prompty comes to mind, given the above requirement, is Locality Sensitive Hashing (LSH) [4]. LSH schemes map the input data space into a set of "buckets" using a probabilistic map such that vectors that are close to each other in the input space map into the same bucket with very high probability, whereas those that are distant from one another have a high probability of being mapped into different buckets. Formally, a locality sensitive hashing function is a map h from $\mathcal{M}$ to a universe U such that, given any two vectors $x_1, x_2 \in \mathcal{M}$,

$$\begin{aligned} d_s(x_1, x_2) < r \quad &\Rightarrow \quad h(x_1) = h(x_2) \text{ with probability } \geq P_1 \\ d_s(x_1, x_2) > cr \quad &\Rightarrow \quad h(x_1) \neq h(x_2) \text{ with probability } \geq P_2 \end{aligned}$$

for some radius of interest r and some constant c. Ideally both P_1 and P_2 are high. Thus, vectors that are less than a distance r apart have a high probability of being hashed to the same value, while they are highly improbable to do so if they are more than cr apart.

Although LSH schemes have primarily used as a mechanism for fast nearest-neighbor matching [4], [5] they can, in fact, also be used to both hide the data, and compute distances [6], [7]. However, they do not satisfy our second requirement – they are not information-theoretically secure. In order to guarantee privacy they must be combined with homomorphic encryption schemes [8]. In [9] Boufounos *et al.* propose a band-quantization scheme that is shown to be information-theoretically secure, and has been successfully applied to nearest-neighbor problems [9]; however it reveals the length of the input vector.

In this paper we propose a *Secure Modular Hashing* scheme for vectors in Euclidean spaces (with the ℓ_2 metric), which satisfies all of our requirements. The proposed solution is a simple *modular* variant of the p-stable LSH from [7]. Our proposal may also be viewed as a generalization of the solution in [9] although in the process of generalization it becomes more complete. The scheme naturally provides a two-parameter mechanism to threshold how much information to reveal, and is also information theoretically secure outside this threshold.

In the following sections we first define the secure modular hashing scheme, and subsequently demonstrate several of its properties including those we state above. Finally we show,

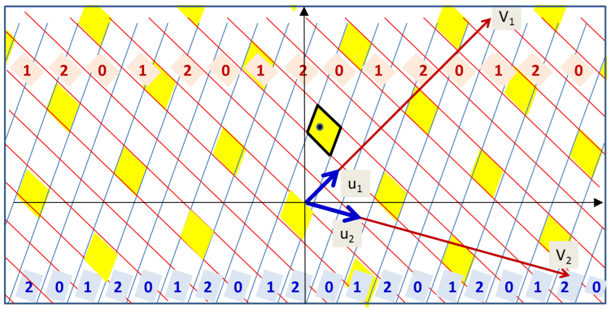


Fig. 1: Illustration of SMH using two hashes with $k = 3$. The black dot represents data vector x. The two red vectors v_1 and v_2 represent two projections φ_1 and φ_2. The two short blue vectors u_1 and u_2 represent the corresponding random biases U_1 and U_2. The SMH stripes the space perpendicularly to the projections. The width of the stripes is δ. Red lines show the stripes corresponding to v_1 and blue lines show the stripes corresponding to v_2. Stripes are indexed modulo k (3 here) by the SHM, starting from the stripes in which the biases (u_1 and u_2 respectively) lie. The red numbers show the indices corresponding to v_1 and the blue numbers show indices corresponding to v_2. The vector gets mapped into the indices of the cell within which it lies. In this example it is mapped into the cell $(1, 1)$, which is outlined in black and highlighted in yellow. Other cells which are also mapped onto the same indices $(1, 1)$ are also highlighted in yellow. These are indistinguishable from their SMH.

through a simple experiment, how it may be used in a biometric application.

II. Secure Modular Hashing

Definition: Let x be a point in $\mathbb{R}^N$. We define the *Secure Modular Hash* function as a random projection $\mathbf{Q}_k$ from $\mathbb{R}^N$ into $(\mathbb{Z}/k)^J$ (where $\mathbb{Z}/k$ represents the set of integers $\{0, 1, \cdots, k-1\}$), such that $\mathbf{Q}_k(x) = [Q_k^1(x)\ Q_k^2(x) \cdots Q_k^J(x)]^\top$, and each $Q_k^i(x)$ is given by

$$Q_k^i(x) = \lfloor \varphi_i^\top x + U_i \rfloor \ (\text{mod}\ k) \tag{1}$$

with $U_i \sim \text{unif}(0, k)$ and $\varphi_i \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{\delta^2} I_N\right)$ independent. δ is a scalar factor to control the variance of φ_i.

The SMH function of Equation 1 is similar to the p-stable LSH of [7] with two minor differences. Equation 1 would indeed be identical to a Gaussian LSH ($p = 2$) if $U \sim \text{unif}(0, 1)$ and the modulus were absent. But these differences are key. The modularity w.r.t. k makes the hash information theoretically secure. The uniform scalar $U \sim \text{unif}(0, k)$ hides information about the length of x. For $k = 2$, the proposed function is also similar to the band-quantized hashing scheme of [9], which is similar to the proposed scheme with $k = 2$ but with $U \sim \text{unif}(0, 1)$. Consequently, the latter too shares the ITS property that vectors do not reveal information about one another; however it reveals information about the length of x.

Figure 1 illustrates the geometrical interpretation of the SMH. The function has two user-modifiable parameters, the scalar resolution δ and the modulus k, and two randomly generated parameters, the set of random vectors $\Phi = \{\varphi_i,\ i = 1, \dots, J\}$ and the set of random scalars $\mathbf{U} = \{U_i,\ i = 1, \dots, J\}$. When a user Alice must transmit the vector $\mathbf{Q}_k(x)$ to server Bob, she retains $\{\Phi, \mathbf{U}\}$ as her private key. In order to compute the distance between two vectors x_1 and x_2, Bob computes the following empirical average instead

$$d_Q(x_1, x_2) = \frac{1}{J} \sum_i \mathbb{I}(Q_k^i(x_1) \neq Q_k^i(x_2)) \tag{2}$$

where $\mathbb{I}$ is the index function. The distance $d_Q(x_1, x_2)$ simply computes the fraction of the total hashes for x_1 and x_2 that do not match.

We show in the following sections that $d_Q(x_1, x_2)$ is monotonically related to $||x_1 - x_2||$ in expectation, and that $\mathbf{Q}$ itself is secure. Since the individual maps Q_i are statistically independent, it is sufficient to analyze the basic hash specified by Equation 1. To simplify notation, we will drop the superscript i in the following discussions.

III. Properties of the Secure Modular Hash

In the following discussion we use the notation $\mathbb{P}(A)$ to represent the probability of the event A.

A. The SMH is uninformative

Theorem 1. $\forall x \in \mathbb{R}^N$ and $\forall i \in \mathbb{Z}/k$

$$\mathbb{P}\left(Q_k(x) = i\right) = \frac{1}{k}$$

To prove Theorem 1, we first need the following result.

Proposition 1. Let X be a continuous random variable with any probability density function. Let U be a random variable that is uniformly distributed between 0 and k. If X and U are independent, then $(X + U)(\text{mod}\ k)$ is distributed uniformly between 0 and k.

Proof: The proof for the proposition is provided in [16]. ■

Proof of Theorem 1

To complete the proof of Theorem 1, we note the following:

$$\lfloor \varphi^\top x + U \rfloor \ (\text{mod}\ k) = \lfloor \varphi^\top x + U\ (\text{mod}\ k) \rfloor \ (\text{mod}\ k). \tag{3}$$

Combining Equation 3 with Proposition 1 provides the necessary result. ■

The consequence of Theorem 1 is that $\lfloor \varphi^\top x + U(\text{mod}\ k) \rfloor$ is uniformly distributed over $\mathbb{Z}/k$ *regardless* of the value of x itself (or even of its probability distribution, if x were a random vector). The implication is that absent knowledge of U, the hash $Q(x)$ provides no information about x itself.

This is a small but significant difference with the standard p-stable LSH, where no such condition applies. In fact, it is easy to show that in p-stable LSH the length of x is exposed. We demonstrate this below for the case of Gaussian LSH. The statement can also be shown to generalize to other p-stable LSH functions, although the proof is outside the scope of this paper.

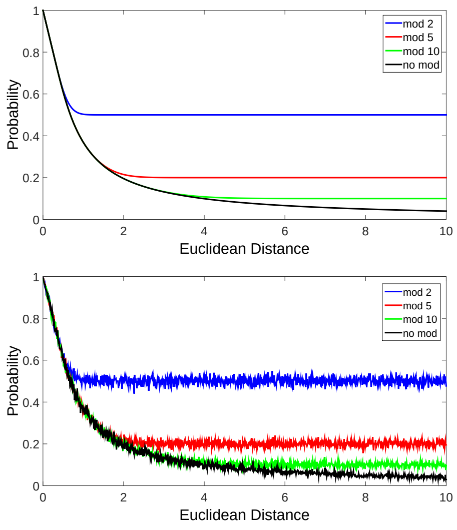


Fig. 2: Upper: Theoretical value of $\mathbb{P}\left(Q_k(x_1) = Q_k(x_2)\right)$ as a function of $\|x_1 - x_2\|$. Lower: Estimates derived from simulations.

Proposition 2. For the p-stable LSH $Q(x) = \left\lfloor \frac{\varphi^\top x + U}{\delta} \right\rfloor$, where $\varphi \sim \mathcal{N}(\mathbf{0}, I_N)$ and $U \sim \text{unif}(0, \delta)$, $\mathbb{P}(Q(\mathbf{0}) = 0) = 1$.

Proof: The statement is obvious from the definition of $Q(x)$ itself: since $U < \delta$, $\lfloor \frac{U}{\delta} \rfloor = 0$. ∎

Proposition 3. If x_1 and $x_2 \in \mathbb{R}^N$ and we define $Q(x)$ as before, then,

$$\mathbb{P}\left(Q(x_1) = Q(x_2)\right) = \\ 2\mathcal{C}\left(\frac{\delta}{\|x_1 - x_2\|}\right) - 1 + \frac{\sqrt{2}\|x_1 - x_2\|}{\sqrt{\pi}\delta}\left(e^{-\frac{\delta^2}{2\|x_1 - x_2\|^2}} - 1\right) \tag{4}$$

where $\mathcal{C}$ is the cumulative distribution of a standard normal distribution.

Proof: The proof is given in [16]. ∎

The implication of the above statement is that the probability that two points have the same hash depends on the distance between the points: $\mathbb{P}\left(Q(x_1) = Q(x_2)\right) = F(\|x_1 - x_2\|)$. Moreover, $F()$ is monotonic in its argument. Setting $x_2 = \mathbf{0}$, and since $Q(\mathbf{0}) = 0$ from Proposition 2, we get $\mathbb{P}\left(Q(x_1) = 0\right) = F(\|x_1\|)$. In other words, given J independent Gaussian LSH hashes of x, we can estimate $\|x\|$ simply by counting the fraction of hashes that are 0.

B. The relationship between $d_Q(x_1, x_2)$ and the distance $\|x_1 - x_2\|$

To relation of the distance $d_Q(x_1, x_2)$ to the actual distance between the vectors can be intuited from Figure 1. If two vectors are very close together, the odds that they will fall within the a cell of the same color are high. On the other hand, if they are very far apart, the odds of this happening are no better than chance. We formalize this in the following theorem.

Theorem 2. If x_1 and $x_2 \in \mathbb{R}^N$, then

$$\mathbb{P}\left(Q_k(x_1) = Q_k(x_2)\right) = \\ \frac{1}{k}\left(1 + 2\sum_{i=1}^{\infty} \text{sinc}^2\left(\frac{i}{k}\right) e^{-2\left(\frac{\pi\|x_1 - x_2\| i}{\delta k}\right)^2}\right) \tag{5}$$

Proof: The proof is presented in [16]. ∎

The importance of this theorem is that the probability that the hashes of two vectors x_1 and x_2 take the same value depends on the distance $\|x_1 - x_2\|$. However, the true significance only appears when we actually plot the probability. Figure 2 plots $\mathbb{P}\left(Q_k(x_1) = Q_k(x_2)\right)$ as a function of $\|x_1 - x_2\|$ for various values of k. The upper plot shows the theoretical values given by Theorem 2. The lower shows empirical estimates derived from a simulation over 1000 runs using $\delta = 1$. The figures show that although the probability that $Q_k(x_1) = Q_k(x_2)$ does depend on $\|x_1 - x_2\|$, it eventually goes to $\frac{1}{k}$. Moreover, it does so exponentially fast.

We formalize this in the following.

Lemma 1. $\forall x_1, x_2 \in \mathbb{R}^N$

$$\frac{1}{k} \leq \mathbb{P}(Q_k(x_1) = Q_k(x_2)) \leq \frac{1}{k}\left(1 + \frac{k^2}{3} e^{-2\left(\frac{\pi\|x_1 - x_2\|}{\delta k}\right)^2}\right)$$

Proof: The proof is presented in [16]. ∎

Corollary. If $\|x_1 - x_2\| \to \infty$ then $\mathbb{P}\left(Q_k(x_1) = Q_k(x_2)\right) \to \frac{1}{k}$

Proof: This is directly obtained using the Squeeze theorem. ∎

The key term in Lemma 1 is the upper bound, which falls to $\frac{1}{k}$ exponentially in $\|x_1 - x_2\|$. We can now state the following.

Theorem 3. The expected value of $d_Q(x_1, x_2)$ is given by

$$\mathbb{E}[d_Q(x_1, x_2)] = \\ 1 - \frac{1}{k}\left(1 + 2\sum_{i=1}^{\infty} \text{sinc}^2\left(\frac{i}{k}\right) e^{-2\left(\frac{\pi\|x_1 - x_2\| i}{\delta k}\right)^2}\right) \tag{6}$$

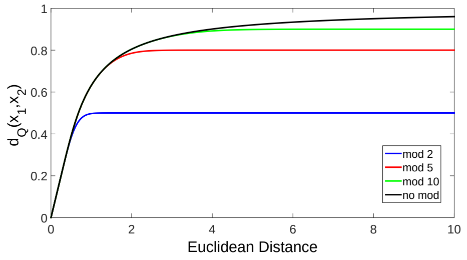
Proof:

The proof is now straightforward. We note that $d_Q(x_1, x_2)$ as defined by Equation 2 is simply an unbiased estimator of $\mathbb{P}(Q_k(x_1) \neq Q_k(x_2)) = 1 - \mathbb{P}(Q_k(x_1) = Q_k(x_2))$. Using the value for $\mathbb{P}(Q_k(x_1) = Q_k(x_2))$ from Theorem 2, the above formula results. ∎

Figure 3 shows the expected value of d_Q and values derived through simulation as a function of $\|x_1 - x_2\|$. The curves are mirror images of the figures in Figure 2.

From Figure 3, it is apparent that $d_Q()$ is a reliable predictor for $\|x_1 - x_2\|$ for small values of this distance. At larger values, however, it rapidly saturates to $1 - 1/k$, at which point they become uniformative. Moreover, the distance increases to $1 - 1/k$ exponentially quickly with $\|x_1 - x_2\|$.

We now attempt to quantify this behavior. We note that $d_Q(x_1, x_2)$ simply computes the average number of mismatches of a collection of hashes. Let $I_k(r) = \mathbb{I}(Q_k(x_1) =$



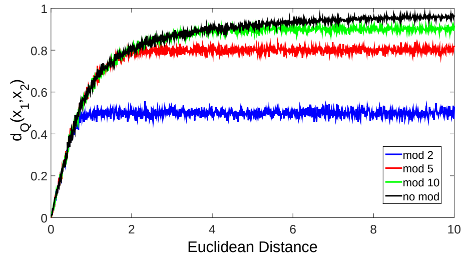


Fig. 3: Upper: Theoretical value of $d_Q(x_1, x_2)$ as a function of $\|x_1 - x_2\|$. Lower: Estimates derived from simulations.

$Q_k(x_2))|_{\|x_1-x_2\|=r}$ be an indicator function that indicates if the hashes for the two vectors are identical when they are a distance r apart in Euclidean space. The SMH distance $d_Q(x_1, x_2)$ merely estimates $1 - \mathbb{E}[I_k(r)]$. Consequently, the distance $d_Q(x_1, x_2)$ between x_1 and x_2 is indistinguishable from the d_Q value obtained when the two vectors are infinitely distant from one another if the distribution $\mathbb{P}(I_k(r))$ cannot be distingished from $\mathbb{P}(I_k(\infty))$. In the theorem below we quantify the divergence between the two distributions and shows that it falls exponentially fast towards 0 with increasing r as well. We will briefly revert to the superscript notation to indicate individual hashes in a collection.

Theorem 4. Let x_1 and x_2 be any two vectors in $\mathbb{R}^N$ such that $\|x_1 - x_2\| = r$. Let $\{Q_k^i(x),\ i = 1, \ldots, J\}$ be a set of of J independent randomly drawn SMH functions. Let $\mathcal{I}_k^J(r) = \{\mathbb{I}(Q_k^i(x_1) = Q_k^i(x_2)),\ i = 1, \ldots, J|\ \|x_1 - x_2\| = r\}$ be a collection of J indicator functions which indicate if corresponding hashes of x_1 and x_2 match. Then, there exists $r_0 \in \mathbb{R}$ such that for all $r > r_0$ the Kullback Leibler divergence between the probability distribution $\mathbb{P}(\mathcal{I}_k^J(\infty))$ and $\mathbb{P}(\mathcal{I}_k^J(r))$ is bounded by

$$D_{KL}(\mathbb{P}(\mathcal{I}_k^J(\infty))\|\mathbb{P}(\mathcal{I}_k^J(r))) \le \frac{2J(k-1)}{3} e^{-2\left(\frac{\pi r}{\delta k}\right)^2}$$

Proof: The proof is given in [16]. ■

Specifically, we show in the proof that

$$r_0 = \frac{\delta k}{\pi}\sqrt{\frac{1}{2}\log\left(\frac{k^2}{3(k-1)}\right)}$$

is a sufficient lower bound on r_0. Figure 4 plots the actual KL divergence as a function of $\|x_1 - x_2\|$ for $\delta = 1$. We observe that this divergence quickly falls to zero. In fact, for $J = 1$ and $k = 2$, the divergence is $O(10^{-16})$ for $r = 5\delta$.

In effect, for any finite number of hashes, the SMH distance d_Q for two vectors that are r apart becomes indistinguishable

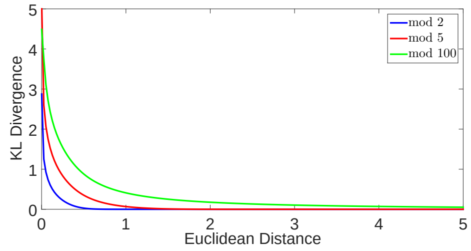


Fig. 4: KL divergence between $\mathbb{P}(\mathcal{I}_k^J(\infty))$ and $\mathbb{P}(\mathcal{I}_k^J(r))$ as a function of r for all $r > r_0$. The divergence was computed using Equation 5. The first 500,000 terms in the series were used to obtain the probabilities.

from that for infinitely distant vectors for all but a small range of values for r. The divergence also increases with the number of hashes J; however this increase is linear, as opposed to the exponential fall in the divergence with increasing r. Increasing J merely expands the radius r within which d_Q carries information about r by a small amount.

Alternately viewed, for any specified ϵ it is possible to find a distance r_ϵ such that the KL divergence between the distribution of SMH distances between vectors that are r_ϵ apart and those between vectors that are infinitely far apart is less than ϵ.

C. Information theoretic security

The above sections show that the SMH maps vectors such that distances can be exposed in a limited manner. We now show that it is also information theoretically secure: knowing the hash for one vector provides no information about the hashes of other vectors that are distant from it.

Theorem 5. The mutual information between $Q_k(x_1)$ and $Q_k(x_2)$ is bounded by $\frac{k^2}{3}e^{-2\left(\frac{\pi\|x_1-x_2\|}{\delta k}\right)^2}$.

Proof: The proof is provided in [16]. ■

The bound given by Theorem 5 is rather weak. Figure 5 shows the theoretical bound given by Theorem 5, and the actual mutual information values obtained through simulations. We note that the simulations show greatly lowered Mutual Information compared to the bound.

Nevertheless, even the weak bounds from Theorem 5 fall exponentially to 0 as $\|x_1 - x_2\| \to \infty$. In fact, for $k = 2$, for distances greater than 5δ, the upper bound on the mutual information falls to below 10^{-50}. To set this in perspective, more than 10^{50} hashes would be requried in order to glean 1 bit of information about a vector y from that of vector x. For larger k the bound is weaker, but for $k = 10$, at a distance of 10δ, the bound is lower than 10^{-11}.

IV. Applying the Secure Modular Hash

The SMH provides us with a mechanism to enable a remote server to compute distances in a controlled manner. Specifically, a user Alice can encrypt and transmit her data to enable a remote server Bob to perform specific computations that require limited distance computation without knowing

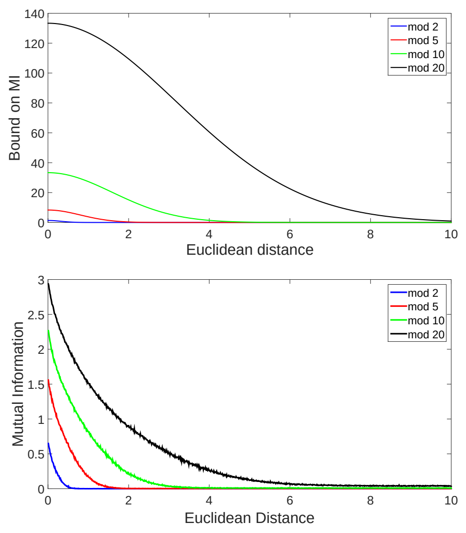


Fig. 5: Upper: Theoretical upper bound on mutual information. Lower: Estimates of mutual information from simulation.

more than the distance values themselves, and even so, only if they are below a threshold. Alice can also control the threshold below which she is willing to reveal information by manipulating k, δ, and J (the number of hashes).

We present here an example scenario from voice authentication. Voice authentication systems require a user to register with the system, which then retains the models of the user. Subsequently, during operation, the user provides the system with additional samples, which the system uses to authenticate them.

The setup is fraught with all of the problems that biometric authentication systems suffer – if the models possessed by the system are somehow compromised, they could be used for nefarious purposes, such as tracking the speaker across systems, recovering the speaker's voice from public repositories, etc. To guard against such mishap, the models in the system's possession must not be usable except by the target user; the system must be unable to infer any information about the speaker either from the model it possesses, or from the data received from the user over multiple interactions. It must nevertheless be able to successfully authenticate the user, while rejecting imposters.

We propose the following framework, which is identical to that proposed in [10]. User Alice generates samples of registration data $x_n,\ n = 1, \dots, N$. She converts each instance x_n to an SMH vector $\mathbf{Q}_k(x_n) = [Q_k^1(x_n)\ Q_k^2(x_n) \cdots Q_k^J(x_n)]^\top$. She uses the same projections $\varphi_i,\ i = 1, \dots, J$, and the biases $U_i,\ i = 1, \dots, J$ to convert all registration vectors x_n. She then transmits $\mathbf{Q}_k(x_n),\ n = 1, \dots, N$, to the system, and retains $\varphi_i, U_i, i = 1, \dots, J$, as her private keys.

System Bob employs an authentication algorithm that depends on the computation of distances between the vectors, rather than on the actual values of the vectors themselves. Such a classifier may be based on nearest neighbors, or, alternately, on methods that compute functions of the distances between vectors. In this paper we use a simple support vector machine with an RBF kernel that has the form $\exp\left(\gamma\|x - y\|^2\right)$ as Bob's classifier.

Bob only receives the SMH vectors $\mathbf{Q}_k(x)$, and does not ever "see" the actual vectors x themselves. Consequently, the distance $\|x - y\|$ required by the classifier is replaced by the function $g\left(d_Q(x, y)\right)$, where the function $g()$ derives an estimate of the Euclidean distance between x and y from $d_Q(x, y)$. For small values of k, the identity function $g(x) = x$ suffices. Note that by the nature of the SMH $g\left(d_Q(x, y)\right)$ will only provide a meaningful proxy for $d(x, y)$ if x and y are sufficiently close. Otherwise, the distance estimate obtained will be meaningless. For the RBF kernel based SVM, we employ the pseudo kernel $\exp\left(\gamma g\left(d_Q(x, y)\right)^2\right)$. Note that this is not strictly a Kernel; however this may easily be replaced by the true Kernel variant proposed in [12], which is also a radial function computed from Hamming distances.

For authentication, Alice records data x and computes $\mathbf{Q}_k(x)$ using her private keys $\varphi_i, U_i,\ i = 1, \dots, J$, and transmits it to Bob. Bob computes the necessary distances from $\mathbf{Q}_k(x)$ within his classifier to determine whether to authenticate Alice or not.

Since Alice retains her private keys, the uninformativeness guarantees of the SMH ensure that Bob cannot invert any given $\mathbf{Q}_k(x)$ to recover the vector x. Moreover, the information theoretic guarantees ensure that even if Bob were to retain and analyze all of Alice's data, he could make no generalizations beyond the limited range within which Alice's data must lie in order for her to be authenticated.

V. Experimental Evaluation

We note from the previous sections that the SMH provides a *limited* mechanism to compute distances. However, the distance proxy $g\left(d_Q(x, y)\right)$ is stochastic; consequently it remains to be tested if it is viable as a divergence measure for performing authentication.

As a proof of concept, we ran speaker verification experiments on the YOHO Speaker Verification corpus [13], comprising short utterances by 138 speakers. Each utterance contains a spoken sequence of three two-digit numbers. The corpus is divided into an enrollment set and a verification set. The enrollment set, which is used for training, includes 96 utterances from each speaker, totaling 14.54 hours of audio. The verification set used for testing contains 40 utterances from each speaker, totaling 6.24 hours of audio. To develop a speaker verification system, we require both recordings from target speakers and a large number of recordings from imposters. In our experiments, however, we did not explicitly record imposters; instead, for each of the 138 speakers in the corpus, the remaining 137 were used as imposters. This enables us to compare results with previous work on secure speaker verification techniques using LSH-based techniques [6].

Audio recordings comprise sequences of samples of varying length. These must be converted to fixed-length "feature" vectors to obtain the data vector x. To effect this conversion, we employ a representation known as *Gaussian mean supervectors* [14]. Each recording is first parameterized into a sequence of "Mel-frequency cepstral coefficient" (MFCC) vectors [15]. Each vector in the sequence is augmented by

#Gaussians	4	8	16	32	64	128
EER	3.55	1.52	0.60	0.33	0.18	0.20

TABLE I: Speaker verification EER (%) using supervectors.

δ	13.5	14.0	14.5	15.0	15.5
mpc=4	2.27	1.79	1.52	1.40	1.25
mpc=8	1.32	1.00	0.84	0.89	0.80
mpc=16	0.76	0.69	0.65	0.60	0.51

TABLE II: Verification EER (%), using SMH with $k = 2$.

the first- and second-order differences between its immediate neighbors, resulting in a sequence of 39-dimensional feature vectors. The distribution of the feature vectors in each recording is modeled by a Gaussian mixture model, learned using a maximum *a posteriori* estimator. The means of the Gaussians are then concatenated to result in the desired supervector.

Table I shows baseline results obtained using the technique from [14]. The procedure works directly on the supervectors and does not attempt to hide them. Results are reported in terms of equal error rate (EER). The resolution of the supervector representation depends on the number of Gaussians in the Gaussian mixture used to model the distributions of the MFCC vectors in the recordings. The performance improves as the number of Gaussians increases. The table shows results obtained with increasing numbers of Gaussians. On this corpus, the results saturate at 64 Gaussians.

Our objective in setting up the baseline was to optimize the feature representation (x) and the classifier itself, in order to ensure that potential performance degradations due to replacing the true Euclidean distance by $d_Q(x, y)$ in the classifier are not masked by suboptimalities in the classifier itself. Having established the numbers, we run subsequent experiments for evaluating the SMH using supervector representations derived from mixtures of 32 Gaussians.

A. Experiments using Secure Modular Hashes

Table II shows verification accuracies obtained using SMH hashes at the system/classifier with $k = 2$. The SMH has two parameters: the variance parameter δ and the number of measurements J. The value of J by itself is not a meaningful number, as different values of L (dimensionality of the supervector) require different values of J; hence we report our results as a function of *measurements per coefficient* (*mpc*), computed as J/L.

Both increasing δ and *mpc* improves the classification performance. The results with increasing δ are initially counter-intuitive. From the illustration of Figure 1 we may infer that increasing δ increases the area of the cells within which distance between points cannot be predicted. However, this is less counter intuitive when we consider that we actually compute statistical averages over several measurements, and increasing δ also increases the distance between the cells. This is also inferrable, both from the actual theoretical relations described in the paper, and the fact that performance increases with *mpc*. At *mpc*=16, the performance stabilizes rather quickly and thereafter is largely independent of δ. Notice that we not only greatly improve on the 11.86% EER reported in [6], but also produce an almost negligible increase in the classification error when compared with the non-secured version in Table I.

VI. Conclusion

We have proposed Secure Modular Hashes as cryptographic alternatives that enable limited release of distances between vectors, *i.e.* only if they are close to each other. They are fast, imposing trivial computational overhead – of the order of milliseconds over a transaction, in our experiments. Such schemes are useful in scenarios such as biometrics where limited release of distance information may be an acceptable alternative to the computational expense of traditional encryption. Our experiments reveal that distances computed from SMH vectors can effectively substitute for actual Euclidean distances with minimal degradation.

There are still many open questions. Can the keys be released as well? May it be possible to have multi-party versions, where φ is made public and each user only retains $\mathbf{U}$ as their key? What role does k play in such scenarios? All these remain topics of active and future research.

VII. Appendix

The appendix with the proofs is available at [16].

References

[1] M. Pathak and B. Raj, "Privacy-Preserving Speaker Verification and Identification Using Gaussian Mixture Models." IEEE Transactions on Audio, Speech and Language Processing, Vol 21:2, pp. 397-406, 2013.

[2] M. Naehrig, K. Lauter and V. Vaikuntanathan, "Can homomorphic encryption be practical?." Proceedings of the 3rd ACM workshop on Cloud computing security workshop, 2011.

[3] T. Frederiksen and J. Nielsen, "Fast and maliciously secure two-party computation using the GPU." ACNS 2013, Vol. Springer LNCS 7954, pp. 339-356, 2013

[4] P. Indyk and R. Motwani. "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality." Proceedings of 30th Symposium on Theory of Computing, 1998.

[5] A. Andoni and P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions." Communications of the ACM, Vol. 51(1), pp. 117122, 2008.

[6] M. A. Pathak and B. Raj, "Privacy Preserving Speaker Verification as Password Matching." Proceedings of ICASSP, 2012.

[7] M. Datar, N. Immorlica, P. Indyk and V.S. Mirrokni, "Locality-Sensitive Hashing Scheme Based on p-Stable Distributions." Proceedings of the Symposium on Computational Geometry, 2004.

[8] S. Rane and P. T. Boufounos, "Privacy-Preserving Nearest Neighbor Methods: Comparing Signals Without Revealing Them." IEEE Signal Processing Magazine, Vol. 30(2), pp. 18-28, 2013.

[9] P. T. Boufounos and S. Rane, "Secure Binary Embeddings for Privacy Preserving Nearest Neighbors," Proceedings of WIFS, 2011.

[10] J. Portelo, A. Abad, B. Raj and I. Trancoso, "Secure binary embeddings of front-end factor analysis for privacy preserving speaker verification." Proceedings of InterSpeech, 2013.

[11] J.-P. Vert, K. Koji Tsuda, and Bernhard Schölkopf, "A primer on kernel methods." Kernel Methods in Computational Biology, 2004.

[12] F. Wang, W.-L. Zhao, C.-W. Ngo and B. Merialdo, "A Hamming Embedding Kernel with Informative Bag-of-Visual-Words for Video Semantic Indexing," ACM Transactions on Multimedia Computing, Communications and Applications, Vol. 10:3, pp. 1-20, 2014.

[13] J. P. Campbell, "Testing with the YOHO CD-ROM Voice Verification Corpus", in *Proc. ICASSP*, Detroit, Michigan, USA, May 1995.

[14] W. M. Campbell, D. E. Sturim, D. A. Reynolds, "Support vector machines using GMM supervectors for speaker verification." IEEE Signal Processing Letters, Vol. 13:5, pp. 308–311, 2006.

[15] B. Davis, and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences." IEEE Trans. ASSP, Vol. 28(4), pp. 357–366, 1980.

[16] `http://mlsp.cs.cmu.edu/public/wifs2015appendix.pdf`