# ATTACKING A PRIVACY PRESERVING MUSIC MATCHING ALGORITHM

José Portêlo[1], Bhiksha Raj[2], Isabel Trancoso[1]

[1] IST / INESC-ID Lisboa, Portugal
[2] Language Technologies Institute, Carnegie Mellon University, Pittsburgh PA, USA

**ABSTRACT**

Secure multi-party computation based techniques are often used to perform audio database search tasks, such as music matching, with privacy. However, in spite of the security of individual components of the matching schemes, the overall scheme may still not be secure. This paper explains how such flaws may occur, using a privacy preserving music matching problem as a template, and provides a solution, and analyzes the resulting tradeoff between privacy and computational complexity. Although the paper focus on a music matching application, the principles can be easily adapted to perform other tasks, such as speaker verification and keyword spotting.

***Index Terms***— Music matching, Audio matching, Privacy, Secure multi-party computation.

## 1. INTRODUCTION

As access to on-line data and services increases, ensuring the privacy and security of the data and users has become increasingly important. A large number of techniques have hence been been developed to protect the data. The majority of these methods are intended to project the data from imposters and eavesdroppers, and do so through encryption schemes that permit only authorized parties to decrypt the data. However, an alternate scenario, where two parties must perform *joint* computations, but desire not to expose their data to one another, has also become increasingly important. For instance, a user "Alice" may desire to query the database belonging to an entity "Bob", but not wish Bob to know what her query was, or the responses she obtained. Bob, in turn, may wish not to let Alice obtain anything besides the responses to her query.

Although such privacy concerns are typically expressed in the context of text and numeric data, a significant recent trend also applies them to multimedia and audio data, where parties may desire to perform mining, recognition or other pattern matching operations collaboratively, but are not trustful of one another. We address one such problem in this paper, that of private music matching. In our scenario (as we explain in Section 3), Alice wishes to obtain metadata about a music snippet she possesses from Bob's catalog, but desires not to permit Bob to know what her song was. Bob in turn desires to protect his catalog from Alice.

In order to deal with such problems a variety of *secure multi-party computation* (SMC) protocols (briefly explained in Section 2) have been proposed in the literature [3]. These protocols employ methods such as *homomorphic* public-key encryption, *oblivious transfer* [8] and randomization, that enable Alice and Bob to perform a variety of "primitive" operations collaboratively, without divulging information. For instance, Alice may possess a vector $\mathbf{a}$, and Bob a vector $\mathbf{b}$. An SMC primitive would permit Alice and Bob to obtain randomly split additive shares of the result for the dot product $\mathbf{a} \cdot \mathbf{b}$ without revealing $\mathbf{a}$ to Bob or $\mathbf{b}$ to Alice. This is achieved through careful, iterated exchange of encrypted partial results in a manner that guarantees that Alice and Bob remain ignorant of each others' data. A variety of SMC protocols have been proposed that permit Alice and Bob to perform such operations as dot products, determining the largest component of the component-wise sums of their data, identifying the largest value, etc. "securely", *i.e.* without revealing their data to one another.

Based on these primitives, larger schemes can be built up that enable two or more users to perform a variety of tasks, such as database searching, classification, prediction, etc. securely, *i.e.* without revealing private information to one another. Relevant to this paper, *audio*-related tasks, such as keyword spotting, speaker identification, etc. can also be performed. It must be noted that since these are composed from operations that require iterated exchange, they tend to be *much more* expensive than the corresponding "insecure" counterparts. Since the larger schemes are composed from provably secure primitives that do not individually reveal information, the larger operations are generally considered to be secure themselves.

However, this is not always the case. Purported proofs of privacy notwithstanding, simply combining a number of secure primitives does not always result in a secure algorithm. We specifically address the problem of secure music matching (which we describe in Section 3), which uses a combination of secure dot product to compute matching scores and a secure max value primitive to identify the best matching song. The algorithm has previously been described in [1] and [2], and can retrieve metadata for Alice without permitting Bob to know her music clip or the metadata returned. However, as we explain in Section 4, in spite of the fact that individual computations are secure, the fact that Alice's clip is likely to match a segment of one of Bob's songs exactly enables him to precompute all such potential matches *a priori* locally, and this may be enough for a motivated Bob to uncover the identity of Alice's clip in polynomial time.

We propose a solution, described in Sections 5 and 6, that employs a variant of *oblivious transfer* [8], which protects Alice's information from Bob by embedding it in random data. We show how this proposed solution prevents Bob from being certain of Alice's data, and explain the probability of exposure. The solution, of course, comes at the cost of greatly increased computation as we explain; security is never free and comes at a computational price.

We note that although the proposed solution is presented in the context of music matching, the general principle holds for any audio matching problems, such as in recognition or mining, where one party with a query intends to find a match in another's database. In fact, our solution generalizes to a variety of scenarios where the security of the privacy preserving number comparison operations is compromised by the availability of prior information. If the entity holding the database can simulate the scores computed during the matching process by comparing portions of his catalog to other portions, the privacy of the querying party can be breached. Our proposed solution will apply in all these cases, as we explain in our

conclusions.

## 2. PRIVACY PRESERVING COMPUTATION

Consider Alice and Bob have private data $\mathbf{a} = \{a_i\}$ and $\mathbf{b} = \{b_i\}$ respectively. They wish to compute a joint function $f(\mathbf{a}, \mathbf{b})$, but without exposing their individual inputs to the other party. Privacy preserving computation techniques enable such operations. Two key tools in such computation that are central to this paper are *homomorphic encryption* and *secure multi-party computation*.

### 2.1. Homomorphic public-key cryptosystem

A public-key cryptosystem consists of a triple of probabilistic polynomial time algorithms for key generation (*KG*), encryption (*EN*) and decryption (*DE*). The key generation algorithm *KG* generates a private key $sk$ and a pubic key $pk$. The encryption algorithm *EN* algorithm encrypts any plaintext message $m$ to a cipher text $c$ using $pk$ as $c = EN(m, pk)$. The decryption algorithm *DE* recovers the original input $m$ from $c$ using the private key $sk$.

A *homomorphic* cryptosystem additionally permits specific algebraic operations to be performed indirectly in the plaintext by manipulating the ciphertext. Specifically, we consider the Paillier homomorphic cryptosystem [5], which as the following properties:

$$EN(a, pk) \times EN(b, pk) = EN(a + b, pk) \quad (1)$$

$$(EN(a, pk))^b = EN(a \times b, pk) \quad (2)$$

### 2.2. Secure Multi-party Computation (SMC)

SMC techniques enable Alice and Bob to jointly compute a function $f(\mathbf{a}, \mathbf{b})$ as if it was computed confidentially by a trusted third party, such that neither party learns anything about the other's data more than what can be inferred from the result. This is achieved through iterated exchange of encrypted or otherwise obfuscated partial results in elaborate protocols that employ homomorphic encryption, *oblivious transfer* [8] and randomization.

Typically, they define a number of "primitive" operations that can be performed "securely" (i.e. without revealing information). At the end of each primitive operation Alice and Bob end up with *random additive shares* $c_a$ and $c_b$, such that the actual outcome of the computation is given by $c_a + c_b$. Some typical primitives are:

- *Secure Inner Product (SIP)*: SIP [4] computes $c_a$ and $c_b$ such that $c_a + c_b = \mathbf{a} \cdot \mathbf{b}$.
- *Secure Max (SMAX) and Secure Max Value (SVAL)*: SMAX [4] computes $c_a$ and $c_b$ such that $c_a + c_b = \arg\max_i a_i + b_i$. SVAL [4] obtains $c_a + c_b = \max_i a_i + b_i$.
- *Permute* [6]: Here Alice and Bob end up with arrays $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$, such that $\bar{\mathbf{a}} + \bar{\mathbf{b}} = \pi(\mathbf{a}) + \pi(\mathbf{b})$, where $\pi()$ is some permutation.

Other such primitives can be defined. Larger computations, such as regression, classification, or, in our case, matching and retrieval, can be obtained through judicious combination of these primitives.

## 3. PRIVACY PRESERVING MUSIC MATCHING

Alice possesses a snippet of a song. She desires to find out what the song is. Bob possesses a song catalog and can help Alice by matching her snippet to songs in his catalog. However Alice does not desire (for whatever reason) to let Bob know what her snippet is. The secure music matching algorithm, described in [1] and [2] addresses this problem. Briefly, the secure music matching algorithm proceeds as follows:

- Step 1 - Cross-correlating the music signals: Alice and Bob use SIP to compute the cross-correlation between Alice's clip and all the recordings in Bob's catalog. The results are distributed among them as random additive shares.
- Step 2 - Obtaining the cross-correlation peaks: Alice and Bob employ SVAL to obtain additive shares of an array with the peak cross-correlation values of each of Bob's songs with Alice's clip.
- Step 3 - Finding the most likely song index: Alice and Bob perform an SMAX protocol over the additive shares of cross-correlation peaks to find the index in Bob's database which corresponds to Alice's song.
- Step 4 - Obtaining the desired information: Alice retrieves the required information from Bob's database using oblivious transfer.

The protocol, if adhered to, should enable Alice to retrieve the metadata for her song from Bob's catalog, while Bob himself never learns what song Alice's snippet contained.

The key step of the process is Step 3, which derives the *index* of the song corresponding to Alice's snippet from Bob's database without revealing it to Bob. At the end of Step 2 Alice and Bob obtain random additive shares $\mathbf{a} = \{a_i\}$ and $\mathbf{b} = \{b_i\}$ of the peak correlation of Alice's snippet with each of Bob's songs: $a_i + b_i$ represents the peak correlation of the $i^{\text{th}}$ song in the catalog. The SMAX protocol must find the index $i$ for which $a_i + b_i$ is maximum, but to do so without revealing the maximum index to Bob it first employs a *permute* protocol.

The permute protocol [6] produces permuted additive shares $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ for Alice and Bob such that $\bar{\mathbf{a}} + \bar{\mathbf{b}} = \pi(\mathbf{a} + \mathbf{b})$ using a permutation $\pi()$ known only to Alice. This is done as follows. Bob generates a private/ public key pair $(sk, pk)$, encrypts $\mathbf{b}$ and sends it to Alice along with the public key $pk$. She generates a random number $s_i$ for each $b_i$ from Bob and adds it homomorphically to the encrypted $b_i$ values as $EN(b_i + s_i, pk) = EN(b_i, pk) \times EN(s_i, pk)$ to obtain $EN(\mathbf{b}', pk)$, where $\mathbf{b}' = \{b_i + s_i\}$. She also computes $\mathbf{a}' = \{a_i - s_i\}$. She then permutes $EN(\mathbf{b}', pk)$ and $\mathbf{a}'$ using her permutation $\pi()$ to obtain $EN(\bar{\mathbf{b}}, pk) = \pi(EN(\mathbf{b}', pk))$ and $\bar{\mathbf{a}} = \pi(\mathbf{a}')$. She forwards $EN(\bar{\mathbf{b}}, pk)$ to Bob who decrypts it to get $\bar{\mathbf{b}}$, while she retains $\bar{\mathbf{a}}$, satisfying the requirements of the protocol.

For SMAX, Alice then generates a random number $r$ which she adds to each of the entries of $\bar{\mathbf{a}}$ and then sends $\bar{\mathbf{a}} + r$ to Bob. Bob can add this to $\bar{\mathbf{b}}$ to obtain $\bar{z} = \bar{\mathbf{a}} + \bar{\mathbf{b}} + r = \pi(\mathbf{a} + \mathbf{b} + r)$. He finds the index of the largest entry in $\bar{z}$ and returns it to Alice, who unpermutes it to find the true index of her desired song. In the process Bob provably learns nothing since the permutation $\pi()$ hides the true value of the index, and the random number $r$ hides the true value of the peak correlation.

## 4. ATTACKING THE PRIVACY OF THE ALGORITHM

In [1] it has been assumed that since each of the component primitives are provably secure, the overall algorithm must be secure as well. Alice and Bob only exchange encrypted data, and rely on secure two party computations to achieve their goals. This should suffice for them to have the guarantee that their privacy is preserved. However, this is a fallacious assumption as we now demonstrate. The problem resides neither in the homomorphic cryptosystem nor in the security properties guaranteed by any of the secure two party computations, but in the fact that the secure computation chosen to perform Step 3 is not adequate for solving the problem at hand.

The problem arises due to two factors: 1. the same operations are repeated on all songs in Bob's catalog, and 2. Bob can validly assume that in the case of a match, Alice's snippet will be very similar to a corresponding segment from one of his songs. The correlation between any other segment and the matching segment will not be significantly different from its correlation to Alice's snippet.

In Step 2, Alice and Bob obtain additive shares $a_i$ and $b_i$ of the peak correlation $z_i = a_i + b_i$ of the $i^{\text{th}}$ song in Bob's data. After Step 3, Bob possesses *permuted* values $\{\tilde{z}_i\} = \{\pi(z_i)\} + r$; the permutation $\pi()$ and noise $r$ are intended to hide the data from Bob. However, Bob can compute differences over all pairs of $\tilde{z}$'s, i.e., $\tilde{z}_2 - \tilde{z}_1 = \pi(z_2) - \pi(z_1)$, $\tilde{z}_3 - \tilde{z}_2 = \pi(z_3) - \pi(z_2)$, ..., thus removing the random number $r$. The complexity of this process is $O(K^2)$. Next Bob computes $z_{uvw}$, which is the peak cross-correlation between the segment $u$ of song $v$ and the song $w$, for all $u, v$, and $w$ with computational complexity $O(\eta^2 K^2)$, where $\eta$ is the average number of segments of the same length as Alice's snippet in a song. He then tries iteratively to find $u_1$, $v_1$, $w_1$ and $u_2$, $v_2$, $w_2$ such that $z_{u_1 v_1 w_1} - z_{u_2 v_2 w_2} \approx \pi(z_i) - \pi(z_j)$, for all $i$, $j$, with complexity $O(\eta K^2)$. Once Bob completes this, he can reverse the permutation and find the index of Alice's song in his database. In practice, he can potentially speed this up by several orders of magnitude using branch-and-bound algorithms.

## 5. SECURING THE ALGORITHM

In this section we describe a robust approach which makes use of the *secure greater-than* (SGT) comparison [7], thus preventing Bob from having access to Alice's data in plaintext. The SGT works by comparing two values at the bit level. Consider the bit representation of values $\alpha = a_i - a_j$ and $\beta = b_j - b_i$ to be $\alpha = \alpha_n \alpha_{n-1} ... \alpha_1$ and $\beta = \beta_n \beta_{n-1} ... \beta_1$ respectively. For each bit $m$, $m = 1, ..., n$, Bob computes $d_m = \alpha_m - \beta_m$, $f_m = \alpha_m \oplus \beta_m = \alpha_m - 2\alpha_m \beta_m + \beta_m$, $\gamma_m = 2\gamma_{m-1} + f_m$ and $\delta_m = d_m + r_m(\gamma_m - 1)$, with $r_m$ a random number. Bob then permutes all the $\delta_m$ and sends them to Alice. Alice now only has to check each $\delta_m$ and see if any of them is 1 or $-1$. If $\exists m : \delta_m = 1$, then $\alpha_m > \beta_m \Rightarrow \alpha > \beta$; if $\exists m : \delta_m = -1$, then $\alpha_m < \beta_m \Rightarrow \alpha < \beta$; else $\alpha = \beta$.

We analyze three different ways to tackle this problem, each providing different levels of privacy and performance: the first aims at optimum performance, the second aims at optimum privacy preservation and the third uses the best characteristics of the other two. We also analyze the execution time of each approach in comparison with $T_{\text{Step2}}$, which is the total execution time of Step 2. This step is also the bottleneck of the original algorithm in terms of execution time.

### 5.1. Maximal performance approach

The first approach uses only the minimum amount of comparisons that one must make in order to find the maximum value from a set of $K$ numbers, which is $K - 1$. The time required to do this is $T^1_{\text{Step3}} = \frac{1}{\eta} \cdot T_{\text{Step2}}$, since now Bob only needs to compute one SGT comparison for each song in his database. However, if he computes only this minimum amount, after the last comparison he knows that Alice's song corresponds to one of the two indexes used in the last comparison, which means that he has a $1/2$ probability of making a correct guess. This approach preserves almost no privacy on Alice's side, and therefore it is not suitable for our purpose.

### 5.2. Maximal privacy preservation approach

The second approach, which provides the maximum privacy possible, requires all cross-correlation values to be compared with each other, implying a total of $\frac{K(K-1)}{2}$ comparisons. If Bob does this, he will have no idea of which song Alice has, but he would require $T^2_{\text{Step3}} = \frac{K}{2\eta} \cdot T_{\text{Step2}}$, which is much larger than $T_{\text{Step2}}$ since $K >> \eta$. Using this approach leads to an unacceptable increase in the execution time of the overall algorithm when compared with its original formulation, and therefore it is also not suitable for our purpose.

### 5.3. Compromise approach between performance and privacy

From the two previous approaches we realize that it is possible for Alice to get the index she wants from Bob's database either in a completely privacy preserving manner or a time efficient way, but not both at once. This leads us to devise a compromise approach that only leaks partial information, so that Bob cannot learn anything he is not supposed to, while also not turning this step of the algorithm into its new bottleneck. The approach we developed achieves this using $\frac{K}{2} \log K$ SGT comparisons, which means that its execution time is $T^3_{\text{Step3}} \approx \frac{\log K}{2\eta} \cdot T_{\text{Step2}}$, and therefore smaller than $T_{\text{Step2}}$ for reasonable values of $K$ and $\eta$. The basic idea is to perform $\log K$ rounds of comparisons with $\frac{K}{2}$ comparisons each. In the next section we will show how we obtained these numbers.

## 6. ANALYSIS OF THE COMPROMISE APPROACH

This section illustrates our approach one step at a time, for small values of $K$, as the results obtained are independent from the value of $K$ considered.

### 6.1. Example with $K = 8 = 2^3$ songs

We start with an example of how our approach works for $K = 8$. Consider the cross-correlation values $z_i$, $i = 1, ..., 8$, such that $z_1 > z_2 > ... > z_8$. Any other combination of greater-than orderings between these numbers does not affect our analysis, which means that any conclusions we take from this specific ordering will also hold for all the other possible orderings.

The basic idea behind our approach is to iteratively perform greater-than comparisons between the maxima of two adequately selected comparisons from the previous round and greater-than comparisons between the minima of the same two comparisons from the previous round. For the first round of comparisons we use the original sequence split into pairs. The sequence of comparisons one needs to make is as follows:

$$1^{\text{st}} : (z_1, z_2)\boxed{1}, (z_3, z_4)\boxed{2}, (z_5, z_6)\boxed{3}, (z_7, z_8)\boxed{4}$$
$$2^{\text{nd}} : (z_1, z_5)\boxed{5}, (z_2, z_6)\boxed{6}, (z_3, z_7)\boxed{7}, (z_4, z_8)\boxed{8}$$
$$3^{\text{rd}} : (z_1, z_3)\boxed{9}, (z_5, z_7)\boxed{10}, (z_2, z_4)\boxed{11}, (z_6, z_8)\boxed{12}$$

For any possible greater-than ordering of cross-correlation values, the previous sequence of comparisons generalizes to:

$$1^{\text{st}} : (z_1, z_2)\boxed{1}, (z_3, z_4)\boxed{2}, (z_5, z_6)\boxed{3}, (z_7, z_8)\boxed{4}$$
$$2^{\text{nd}} : (\max\boxed{1}, \max\boxed{3})\boxed{5}, (\min\boxed{1}, \min\boxed{3})\boxed{6},$$
$$(\max\boxed{2}, \max\boxed{4})\boxed{7}, (\min\boxed{2}, \min\boxed{4})\boxed{8}$$
$$3^{\text{rd}} : (\max\boxed{5}, \max\boxed{7})\boxed{9}, (\min\boxed{5}, \min\boxed{7})\boxed{10},$$
$$(\max\boxed{6}, \max\boxed{8})\boxed{11}, (\min\boxed{6}, \min\boxed{8})\boxed{12}$$

Notice that in each step one can start by comparing either the maxima or the minima of the previous round of comparisons, as long as they are alternating with each other. Getting back to our example, after each round Bob knows:

$1^{\text{st}}$ : $\alpha_1 \leftarrow ((z_1 > z_2 \text{ and } z_5 > z_6) \text{ and } (z_3 > z_4 \text{ and } z_7 > z_8))$ or
$\beta_1 \leftarrow ((z_2 > z_1 \text{ and } z_6 > z_5) \text{ and } (z_4 > z_3 \text{ and } z_8 > z_7))$

$2^{\text{nd}}$ : $\gamma_1 \leftarrow ((z_1 > z_5 \text{ and } z_3 > z_7) \text{ and } (z_2 > z_6 \text{ and } z_4 > z_8))$ or
$\delta_1 \leftarrow ((z_5 > z_1 \text{ and } z_7 > z_3) \text{ and } (z_6 > z_2 \text{ and } z_8 > z_4))$

$3^{\text{rd}}$ : $-$

He can try to guess which song Alice has, by assuming any of the 4 possible combinations of partial greater-than relationships between the cross-correlation values. For each combination Bob obtains:

$$\alpha_1 + \gamma_1 : z_1 >^{z_2}_{z_5}> z_6, z_3 >^{z_4}_{z_7}> z_8 \longrightarrow z_1 \text{ or } z_3$$
$$\alpha_1 + \delta_1 : z_5 >^{z_6}_{z_1}> z_2, z_7 >^{z_8}_{z_3}> z_4 \longrightarrow z_5 \text{ or } z_7$$
$$\beta_1 + \gamma_1 : z_2 >^{z_1}_{z_6}> z_5, z_4 >^{z_3}_{z_8}> z_7 \longrightarrow z_2 \text{ or } z_4$$
$$\beta_1 + \delta_1 : z_6 >^{z_5}_{z_2}> z_1, z_8 >^{z_7}_{z_4}> z_3 \longrightarrow z_6 \text{ or } z_8$$

At the final round Bob does not learn anything new because he does not get any more greater-than comparison requests from Alice. Also, because of the ambiguity introduced by letting Alice decide whether Bob starts the next round of comparisons using the maxima or the minima from the previous round, he cannot make a correct guess on what song Alice has with probability $P$ more than $1/8$, even with partial information he has regarding the correct greater-than ordering of the $z_i$'s. Generalizing this to any $K = M = 2^m$, Bob cannot correctly guess Alice's song with $P > 1/K$.

### 6.2. Generalization for any $K$ songs

Finally, all it remains is to study our approach for the most general case, where $K$ can be any positive integer. Considering for instance the limit situation of $K = 5 = 2^2 + 1$ and any possible ordering of the $z_i$'s, the sequence of comparisons requested by Alice would be:

$1^{\text{st}}$ : $(z_1, z_2)\boxed{1}, (z_3, -)\boxed{2}, (z_5, -)\boxed{3}, (z_7, -)\boxed{4}$

$2^{\text{nd}}$ : $(\max\boxed{1}, z_5)\boxed{5}, (\min\boxed{1}, z_5)\boxed{6}, (z_3, z_7)\boxed{7}, (z_3, z_7)\boxed{8}$

$3^{\text{rd}}$ : $(\max\boxed{5}, \max\boxed{7})\boxed{9}, (\min\boxed{5}, \min\boxed{7})\boxed{10}$,
$(\max\boxed{6}, \max\boxed{8})\boxed{11}, (\min\boxed{6}, \min\boxed{8})\boxed{12}$

In this situation, after each round of comparisons Bob learns:

$1^{\text{st}}$ : $\alpha_2 \leftarrow ((z_1 > z_2 \text{ and } 1) \text{ and } (1 \text{ and } 1))$ or
$\beta_2 \leftarrow ((z_2 > z_1 \text{ and } 1) \text{ and } (1 \text{ and } 1))$

$2^{\text{nd}}$ : $\gamma_2 \leftarrow ((z_1 > z_5 \text{ and } z_3 > z_7) \text{ and } (z_2 > z_5 \text{ and } z_3 > z_7))$ or
$\delta_2 \leftarrow ((z_5 > z_1 \text{ and } z_7 > z_3) \text{ and } (z_5 > z_2 \text{ and } z_7 > z_3))$

$3^{\text{rd}}$ : $-$

As before, Bob can assume any of the 4 possible combinations:

$$\alpha_2 + \gamma_2 : z_1 > z_2 > z_5, z_3 > z_7 \longrightarrow z_1 \text{ or } z_3$$
$$\alpha_2 + \delta_2 : z_5 > z_1 > z_2, z_7 > z_3 \longrightarrow z_5 \text{ or } z_7$$
$$\beta_2 + \gamma_2 : z_2 > z_1 > z_5, z_3 > z_7 \longrightarrow z_2 \text{ or } z_3$$
$$\beta_2 + \delta_2 : z_5 > z_2 > z_1, z_7 > z_3 \longrightarrow z_5 \text{ or } z_7$$

Notice that in this situation Bob also has no idea of what song Alice has, but because $K$ is not a power of 2, the probabilities of him guessing correctly depend on actual index of the right song. This means that, in the current situation, the probability of guessing Alice's song correctly is $1/4$ if it has the index 3, 5 or 7 and it is $1/8$ if it has the index 1 or 2. Like before, this situation can be easily expanded to any value of $K$, and therefore the description of our approach is now complete.

## 7. CONCLUSIONS

In this paper we presented a robust approach to correct a flaw in a privacy preserving music matching algorithm. We identified the source of problem and described several ways to tackle it, illustrating several levels of performance and privacy preservation that can be obtained. This paper also confirms that although SMC protocols allow parties to perform operations in a secure way, how these protocols are used is equally important, as the amount of privacy they guarantee depends on the situation in which they are used. Although our approach for correcting the attack was presented in a privacy preserving music matching algorithm context, it can easily be used in many other situations which at some point need to compute the maximum of an array in a secure way.

## 8. ACKNOWLEDGEMENTS

José Portêlo was supported by FCT grant SFRH/BD/71349/2010. Bhiksha Raj was partially supported by NSF grant 1017256.

## 9. REFERENCES

[1] M. Shashanka and P. Smaragdis, "Privacy-Preserving Musical Database Matching", in *2007 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Platz, NY, October 21-24, 2007, pp. 319-322.

[2] J. Portêlo, B. Raj, A. Abad and I. Trancoso, "On the Implementation of a Secure Musical Database Matching", in *Proccedings of European Signal Processing Conference - EUSIPCO 2011*, Barcelona, Spain, August 29-September 2, 2011, pp. 1949-1953.

[3] Y. Lindell and B. Pinkas, "Secure Multiparty Computation for Privacy-Preserving Data Mining", in *Journal of Privacy and Condentiality*, 1(1):5998, 2009.

[4] P. Smaragdis and M. Shashanka, "A Framework for Secure Speech Recognition", in *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 15, No. 4, May 2007, pp. 1404-1413.

[5] P. Paillier, "Public-key Cryptosystems based on Composite Degree Residuosity Classes", in *Proceedings of Advances in Cryptology - EUROCRYPT'99*, ser. Lectures Notes in Computer Science, J. Stern, Ed., vol. 1592, 1999, pp. 104-120.

[6] M. Atallah, F. Kerschbaum and W. Du, "Secure and Private Sequence Comparisons ", in *Proceedings of Workshop on Privacy in the Electronic Society*, Washington, DC, October 2003.

[7] I. Blake and V. Kolesnikov, "Strong Conditional Oblivious Transfer and Computing on Intervals", in *Proceedings of Advantages in Cryptology - ASIACRYPT'04*, volume 3329 on LNCS, pp. 515-529, Springer-Verlag, 2004.

[8] M. Naor and B. Pinkas, "Oblivious Transfer and Polynomial Evaluation", in *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, 1999, pp. 245-254.