# Privacy Preserving Biometric Database

Melissa Chase[1] and Orr Dunkelman[2,3,⋆]

[1] Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
melissac@microsoft.com
[2] Computer Science Department
University of Haifa
Haifa 31905, Israel
orrd@cs.haifa.ac.il
[3] Faculty of Mathematics and Computer Science
Weizmann Institute of Science
P.O. Box 26, Rehovot 76100, Israel

**Abstract.** As part of a fight against ID misuse, the state of Israel has decided to establish a biometric database storing biometric information (fingerprints and high definition photographs) of all citizens (over the age of 18). Obviously, such a database is a privacy and security concern unless properly deployed and handled.
As the prominent publicly acknowledged target of this database is to detect fraudulent individuals who are in the possession of more than one ID, we devise a new scheme which allows identifying this behavior while offering maximal privacy, even in the case of database leakage. At the same time, our solution addresses the other publicly acknowledged goals of the database, again without violating the privacy of citizens.
Our proposed solution is based on the existence of minimally trusted *blinding entities*; sensitive operations are performed using a secure multi-party computation protocol, which guarantees security as long as some a fraction of these parties is honest. Furthermore, the protocols we use involve a small number of participants and a simple computation, and thus can be efficiently implemented and verified, even when the number of individuals in the database is very large.

## 1 Introduction

In this paper, at a high level, we consider the following problem: Suppose we have a very large number of people, we want to collect some data from each of them, and we want to determine whether all of the data items are unique (and identify duplicates if not). In general, this is an easy problem to solve using modern database techniques, even if the number of people to be sampled is very large. However, what if the data is private? Even more, what if we have the

⋆ The research described in this paper was done while the second author was visiting Microsoft Research Redmond.

additional requirement that no one should be able to test whether a particular person's data has been included in the database, even given access to that data? Is there some way that we can perform this same analysis while both preserving the participants' privacy and maintaining the efficiency and scalability of the basic database-based system?

*Is this a real problem? Yes!* In the last few months, the state of Israel has been discussing the establishment of a biometric database which will store biometric data of all Israeli citizens. This move is part of the new generation of ID cards which are going to include a smart card and biometric data that is bound to this smart card. As part of the smart ID card initiative in Israel, the biometric database has been promoted as a solution to a real problem: people who fraudulently hold several ID cards. This problem has several causes related to the (mis-)management of the Israeli ID registry. For example, people who permanently leave Israel may give/sell their IDs to other entities, or deceased people's ID cards may not be revoked and may continue to be used by other people (this apparently is done quite often, mostly by relatives of the deceased). This fraudulent behavior has both economic effects (as the social security for these "persons" is still collected), as well as social/political effects (as these "persons" can continue voting in elections). We note that due to the low security of the current ID cards (which are composed mostly of a laminated page containing a few lines of information about the holder and a picture), it is easy to "fix" the photo embedded in the card. [1]

Without going into the philosophical debate over whether these causes are sufficient to justify the creation of a biometric database for the entire population, we aim to offer a construction which allows the identification of people with multiple ID cards, without revealing any information about honest citizens.

*Our setting* We have a large number of people (citizens), each of whom will visit an acquisition station. The acquisition station determines what data must be stored for the person (e.g. by measuring his biometric info, and noting the issued id number). Now, we would like to store this data in such a way that we can check for duplicate values; in particular here we want to check whether any biometric appears with two different ID numbers.[2]

However, storing the biometric data directly in a database seems undesirable. This would potentially allow misuse by administrators of the database (who would need to have access to this sensitive biometric data in order to perform the desired queries). Moreover, if the database was ever compromised, all of this sensitive data might be leaked. ence, given the sensitivity of biometric data,

[1] We note that other countries in the world suffer from similar ID misuse. In South Africa, it is very common for identity thieves to bribe the issuing clerks, and to obtain a "legitimate" identification generated by the officials.

[2] We note that the biometric data stored on the card need not contain the same set of measurements as those used to generate the database. Thus, while we require a small amount of robustly measured data for index our database entries, the card may store a much wider range of biometric data.

assuming that a trusted party will manage the database, is not an ideal solution. However, we must make some trust assumptions in order to obtain the required functionality. We choose to assume only that there is some set of parties where it is safe to assume that at least a few of them are honest. We refer to these parties as "blinding entities", for reasons which we will explain shortly. In our application candidate blinding entities include NGOs, political parties, and even supreme court judges (or retired ones).

Once we assume the existence of these "blinding entities", the problem becomes theoretically very easy. We can simply use techniques from Secure Multi-Party Computation (MPC) [GMW87,Yao86] so that each party stores a share of the entire database, and then, when each person's data is inserted, they can perform an interactive protocol to update their shares of the database and check for any duplicates. However, as general MPC protocols are inefficient for even moderately complex computations, processing the entire database of millions of citizens again and again is completely impractical. Moreover, the leakage of a sufficient number of shares, would allow the reconstruction of the entire database, thus revealing all the biometric data that was hidden in it.

Here, we consider a somewhat different approach. We will construct a solution which does not rely on the secrecy of the database. Instead, we will require that before the data is stored, it is processed by the "blinding entities" in such a way that the resulting values 1) protects the participants' privacy, and 2) allows the identification of duplicate values without any secret knowledge. Thus, only this one-time processing of each entry needs to be done securely, and no special trust is required to store the database or search for collisions.

*Additional Requirements* We consider a number of additional requirements that would be beneficial in the biometric database scenario. These include functionality requirements, such as the ability to re-issue lost cards, to look up the identity of a citizen given their biometrics (but only with permission from the person or appropriate authorities), or to include some additional data which will only be revealed in case potential fraud is detected. They also include practicality restrictions, like the requirement that the process work online so that new citizens can constantly be added, the ability to store the data in a distributed database, and the limitation that the secure computation performed by the acquisition station be kept as minimal as possible so that the implementation can be independently verified. Details on all these are in the following section. We wish to satisfy all of these requirements and provide an efficient, practical, and private solution.

*Our solution.* Our solution combines several ideas from secure multi-party computation and electronic cash, and as mentioned earlier, assumes the existence of a set of *blinding entities*. These blinding entities are entities which do not collude and that have limited power in the suggested protocols.

It is important to note that our solution offers a level of privacy such that even if the entire database is given to the adversary together with the biometric of a specific individual, the adversary cannot determine whether that individual has been included in the database (without the consent of a threshold of the

blinding entities). This ensures that some delicate queries to the database are not only controlled by legal means, but also by technological means and shared trust, which reduces the possibility of misuse.

*Is a biometric database necessary?* In this paper we do not address the fundamental question of whether such a database should exist, or what the associated security risks would be (especially given Israel's geopolitical situation). The transition to a smart ID card ("Teo'dat Zehut Chachama" in Hebrew) seems to be in the consensus (as Israeli ID cards can be easily forged). At the same time, the current Israeli legislation mandates that as part of this initiative, a biometric database, which involves sampling the biometric data of all citizens, is to be created and controlled by the government. It is not surprising that such a requirement has been met with objections and has raised a lot of opposition.

Our goal here is to offer a solution that is practical, secure, and most importantly, privacy-preserving. In other words, we design a construction which can address legitimate objectives of the initiative (e.g., the identification of individuals in the possession of multiple ID cards) without revealing any information concerning the individual, even in the case where the entire database is leaked.

*Related work.* The problem of detecting duplication in data sets has been studied in many contexts. One can even treat the Lempel-Ziv compression algorithm as one such algorithm. At the same time, the problem of detecting duplication, where the data is held by an entity which is not trusted to protect the privacy of the data is more restricted and requires the use of special tools.

For example, the problem of search over encrypted data, falls into this category (e.g., [BCOP04]). However, thse protocols require either a trusted batch preprocessing step or linear search time to verify that a given item is not included in the database. (In our case this would translate into quadratic time as we need to verify that each individual has not already been added to the database.)

A different approach would be a direct application of secure multi-party computation, where all citizens submit their data in a secure and private manner to a protocol that outputs the colliding IDs (if any exist). This approach is completely impractical, as it would require all citizens to simultaneously be online and participate in a protocol (or at least contribute inputs before the protocol could be run). Furthermore, there would be no way to verify that the citizens provide valid inputs, and the citizens might have motives to manipulate the data they provide.

A slightly similar problem was studied recently in [ARF+10], where the problem of collecting data from different machines in a private manner was studied. The solution proposed in [ARF+10] is based on collecting all of the data using proxies which blinds and shuffle the data before passing it to the server After this aggregation, the server can then look for patterns in the resulting data (and depending on the settings, contact the proxy again in order to recover private information embedded into the entries). This type of *privacy preserving data aggregation* can solve the problem at hand, but at the same time, the proposed

solution of [ARF+10] is not optimal for our scenario. Most notably, the privacy guarantees of their solution require that the proxy process entries in large batches. In our setting we expect to be constantly adding more biometrics to our database (initially as citizens are processed, and later as more citizens are born or immigrate). However, this may not occur at a sufficient rate for this batching to provide any reasonable privacy guarantees. Thus our privacy concerns dictate a slightly different approach.

Moreover, the protocol of [ARF+10] requires the user to perform complex cryptographic operations. We cannot assume that all citizens have the ability to perform these operations, or (as mentioned above) trust them to provide the correct inputs. Thus, in our setting the citizen will be represented by the measurement device. As it is crucial that this device be trusted, we aim to minimize the work it must perform, so that these operations can be implemented in verified hardware.

Finally, we note that one can easily transform our solution to an efficient privacy preserving data aggregation with conditional release, for the functionality of threshold larger than 2. It is also possible to generalize our scheme to higher thresholds.

*Organization.* The remainder of this paper is organized as follows: Section 2 goes over the problem description in more detail, and describes the required properties and how they address the objectives of the database. In Section 3 we describe our proposed solution, and finish up with a discussion of a few caveats (in Section 3.8). Finally, this paper is summarized in Section 4.

## 2 Our problem

We now describe our problem in a bit more detail.

### 2.1 Describing the setting

*The data.* We assume that each person's data is composed of three parts:

**Indexing data** This is the data that will be used to sort the entries - we expect this data to be consistent across different measurements of the same person.[3] We also assume that most people's indexing data will be unique, and that this data is very private. In our application, this corresponds to the citizens' biometric data.

**Associated data** For honest people, we expect that all readings will also provide the same associated data. Our goal is to find cases where this does not happen, i.e. where the same index data appears with different associated data. In some settings this data may be private as well. In our application, this corresponds to the ID number of each citizen.

[3] For discussion of this issue, see Section 3.8.

**Payload data** This is the data that will be revealed in the case of duplicate entries. In our application, this should be whatever information is necessary to investigate potential fraud. This could include contact information, or more detailed biometric information, or simply the ID number. Some portion of this information could also be encrypted using a traditional public key encryption scheme so that it could only be read by the appropriate authorities. However, we will guarantee that even these authorities will have access to this information only in the case of duplicate entries.

*The entities.* We consider a scenario involving the following entities:

**People to be sampled.** These are the parties from whom we wish to collect data. We assume that there are many of them, potentially millions. In our application, this would include all Israeli citizens.

**Measurement devices.** Each person will visit an acquisition station to have her or his data collected. The measurement device will collect the data (e.g. read the biometric), and then communicate with the blinding entities as necessary. For the most part, we assume these devices are trusted not to reveal people's data (see below). We also require that these devices keep no state between different readings.

**Blinding entities.** The blinding entities will be responsible for cooperating with the measurement device to process the data before it is stored in the database. They will each store some secret key to be used in this process. We assume that at least one (or more generally, some threshold) of these entities are honest.

**Database.** The blinded entries resulting from this process will be stored in the database. Our solution is such that for our privacy guarantees we do not need to assume that the database is stored securely.

**Fraud detector.** Any party with access to the database should be able to check for duplicate entries, and, if any are found, to return the corresponding payload data.

*The protocols.* Our solution will consist of the following protocols:

BlindEntry**.** This is a protocol performed between the measurement device and the blinding entities. The measurement device has as input a single person's data, and each blinding entity has as input his own secret key. The output of this protocol is a blinded entry to be stored in the database. In our scenario, this protocol is run each time a citizen visits an acquisition station.

CollisionQuery**.** This protocols can be run by any party with access to the database, acting as fraud detector – it requires no secret information. It will produce a list of all the payloads corresponding to pairs of entries where the same indexing data appears with different associated data. (In our application this is used to identify when a specific person is issued more than one ID card. In this case we should be able to find the ID numbers of all cards issued to that person and the associated contact information, but no information about honest citizens should be revealed.)

BESetup. This is the protocol that the blinding entities run to set up their secret keys. This protocol is normally run only once, before any data is sampled.

We also consider two additional protocols that might be useful in our application

ReverseLookup. This is a protocol performed between the blinding entities, presumably only with the permission of the appropriate authorities, which allows them to look up a person given their indexing data and to retrieve the associated payload data. Each blinding entity has as input his own secret key and a given indexing data value that was supplied in the court order and not obtained from the measurement station. The output is a pseudo-entry that allows locating the associated payload data. If we consider our application this corresponds to a query where given a biometric measurement, we want to identify the corresponding person; this might be used to identify an Alzheimer patient, an unconscious person, or a corpse who cannot be identified in any other manner.

VerifyEntry. This is a protocol performed between the measurement device and the blinding entities. The measurement device has as input a single person's index data and associated data, and each blinding entity has as input his own secret key and access to the database. The output of this protocol is a bit representing whether this index data and associated data have already been stored in the database. In our application, this corresponds to the scenario where a person who lost his ID wants to obtain a new ID card. In this case we only need to verify that this person has correctly provided the original ID number corresponding to his biometric.

### 2.2 Properties of our system

Here we describe the properties that we require from the above system.

First, we require some basic ***correctness properties***. Informally, we require that if a majority of the blinding entities follow the protocol, and if the measurement device performs honestly, then the following is true: the BlindEntry procedure will always produce entries such that 1) CollisionQuery finds the payloads corresponding to all pairs of entries where one index data value appears with two different associated data values; 2)ReverseLookup finds the payload corresponding to the provided index data (if it exists in the database); and 3) VerifyEntry returns true for any index data and associated data pair that has in fact already been stored.

We also have some ***efficiency requirements***.

- The computation performed by the measurement device during BlindEntry and VerifyEntry should be minimal. This is because we assume that the measurement device will be independently verified. The measurement device must be trusted not to leak people's data, and at the same time, it may run in a somewhat untrusted environment. Thus, verification is essential, and in order to make verification and hardware implementation more pracical, we must keep the device's computation as simple as possible. [4]

[4] See section 3.8 for more discussion.

– The protocol BlindEntry must be fairly efficient, as it will be run for all people to be sampled.
– The protocols CollisionQuery, ReverseLookup, and VerifyEntry must make limited use of the database - essentially they should be restricted to a small number of lookup operations. (This is important as we expect the database to be quite large.)

At the same time, we wish to offer full privacy to the people in the database. We require the following ***privacy properties***.

– If some subset (less than a given threshold) of the blinding entities are corrupt and given access to the database, the stored data should reveal no information about people's data (i.e., citizens' biometrics or ID numbers). Furthermore, given access to some of a person's data, (e.g., a person's biometric and/or ID number) and to the stored data there should be no way for this subset of entities to determine whether that person has been included in the database (unless there is a collision, and this information is revealed by the payload data).
– If some subset (less than a given thresholds) of the blinding entities are corrupt and given access to the database, and if at some point some of the measurement devices are corrupted, it is unavoidable that they will learn the data from people who are measured at that device after that point, and that they will be able to test whether various index data values and/or associated data values have been stored in the database and learn the corresponding payload data if so (by "injecting" the same data used in ReverseLookup). However, they should not be able to learn anything else, and the number of queries they can make should be limited by the number of times the honest parties execute the protocol.
– To the extent possible, we would like to guarantee some security even if all the blinding entities' keys are compromised and the database is revealed. In this case we will not be able to prevent the attacker from testing whether certain biometrics are contained in the database. However, we would like to make it difficult for the attacker to use the database to extract biometrics that he does not already know. Essentially, we want to guarantee that testing is the only thing the attacker can do.
– It should be impossible to forge electronic evidence of fraudulent behavior (i.e., the state cannot forge a second entry for an honest person) without cooperation of a threshold of the blinding entities.

*Variations* We can also consider simpler variations, where the associated data need not be hidden, or where there is no payload data. Constructions for these cases can be derived via straightforward simplifications of the protocols presented in the following sections - we postpone this to the full version.

### 2.3 Why this model is appropriate for our application

We need a solution which is practical, and which will scale to millions of citizens, and yet which protects the citizens' privacy. The fact that we can store

the blinded entries in a standard database, and find fraudulent entries without involving the blinding entities or any complex cryptographic protocol makes this solution much more scalable. At the same time, the introduction of the blinding entities allows us to distribute the trust involved. And, while the state holds the database, the entries obtained from the acquisition protocol are completely blinded, so even if this information is leaked there will be no breach of privacy. [5]

Another advantage of our proposal is the ability to identify the cheating entities in a convincing manner. First of all, once a fraud exists, the blinding entities cannot hide it (in other words, to identify the fraud, no cooperation from the blinding entities is needed). Moreover, unless the secrets of the blinding entities are revealed, the state cannot forge fraudulent entries.

Finally, we note that any query concerning whether a person is in the database or to find his (or her) ID requires the cooperation of a threshold of the blinding entities. This is extremely important in two cases: The first is preventing abuse (as any inverse information query must be authorized by the court, and done with the participation of the blinding entities). The second is the case where the regime change, at which point the blinding entities can destroy the secrets, thus preventing the abuse of the database by a corrupted regime.

We note that one practical issue to address is how to ensure that the data provided by the acquisition station is correct. Having a single acquisition station is impractical. Hence, we propose to have many stations all running verified hardware. All blinding entities will use point-to-point communication lines to communicate with each acquisition station. We note that ideally a representative of each blinding entity should always be present to verify that the acquisition is done properly.

## 3 The Proposed Solution

Here we present the details of our construction, which makes use of various cryptographic tools including pseudo-random functions (PRFs), secret sharing, and secure multi-party computation (MPC). (For an explanation of these tools, see appendix A.) We will first describe the solution in terms of these building blocks, and then discuss advantages of various specific instantiations. We also discuss some challenges and caveats in Section 3.8.

### 3.1 Blinding Entities

Our proposed solution relies on the existence of several blinding entities. These entities are used as a way to distribute the trust required to compute a database

[5] We note that as all entries are blinded, in theory (as long as the secrets of the blinding entities are not revealed), one could publish the entire database online. Of course, this should not be done, but even if such a leak occurs, it would not compromise the citizens' privacy, as without these secrets all entries are indistinguishable from random strings.

entry from each biometric sample. In other words, only if a threshold of these entities agree to perform the biometric acquisition (or identification) of a person does this acquisition (identification) take place.

Our system will be based on four secret values $s, r, t, z$. These values will not be known to anyone. Instead, we will use secret sharing, and each of the $\ell$ blinding entities $\mathcal{BE}_1, \mathcal{BE}_2, \ldots \mathcal{BE}_\ell$ will hold a share of each of the four secrets. (We call entitiy $\mathcal{BE}_i$'s shares $s_i, r_i, t_i, z_i$ respectively). We note that these four secret shares together comprise the secret key of the blinding entity. They are to be kept in a secure hardware, in an encrypted manner (preventing the access to the secret without the knowledge and consent of the blinding entity). [6]

The secrets can be generated in a shared manner among the blinding entities using a fairly standard secure multi-party computation (this defines BESetup). Moreover, in case of change in the entities themselves, one can run secure multi-party protocol for reconstructing the original secret and redistributing it to the new entities. We note that as these events are supposed to be rare, one could use the generic secure multi-party protocols, which may be slow, but are proven to be secure.

We note that one can define the number of entities and the threshold of honest non-colluding entities, as one sees fit. Of course, the more resilient the parameters, the expected running time of the protocols is expected to become slower, so there is a tradeoff to consider. We defer more discussion to the full version.

### 3.2 The Acquisition Process

We assume that we can sample each person's biometric data in such a way that there exists some (possibly short) string $\mathcal{B}$ that can be consistently extracted. This information can differ from the one stored on the card, as its only purpose is to allow fraud detection. For more discussion on how the sampling can be done, see Section 3.7.

When a person comes to obtain an identity card for the first time, his (or her) biometric data is measured and stored in the ID card (which may be different than the biometric data that goes to the database). At this point, the following process (referred to as BlindEntry) is invoked to add the person to the database:

First $\mathcal{B}$, the biometric data, is sampled and temporarily held in a secure hardware device (which will erase it at the end of the acquisition protocol), and then the following protocol is invoked:

– Let $s, r, t, z$ be the secrets reconstructed from the shares $s_i, r_i, t_i, z_i$. Using a secure multi-party computation, the following entry is computed:

$$f_s(\mathcal{B}), f_r(\mathcal{B}) \cdot f_t(\mathcal{B})^{f_z(id)}, f_z(id), E_{f_r(\mathcal{B})}(id), \sigma_1, \ldots, \sigma_\ell$$

[6] Moreover, depending on the exact adversarial model, it may be beneficial to store these secrets using a steganographic file system [ANS98,MK99] (to further protect against adversaries who obtain the actual storage device).

where $id$ is the person's ID number, $E_k(\cdot)$ is the symmetric-key encryption under the key $k$, $f_x(y)$ is a pseudorandom function whose image is $Z_p$ (we discuss its properties and implementation in Section 3.6), for a large prime $p$ (such that the length of $p$ is larger than 128 bits), and $\sigma_i$ is a signature from blinding entity $i$ on the rest of the entry. (Here we assume the payload data is simply the citizen's ID number, but one could easily change this to use any other relevant information.) We note that $h(\mathcal{B})$ can be computed by the acquisition station, and used instead of $\mathcal{B}$ to provide some form of security even if all of the blinding authorities' keys are revealed.

– All the blinding entities as well as the state's entity verify all the signatures, and if the verification succeeds, the entry is added to the database.

*Privacy guarantees* It is easy to see that given a person and his (or her) ID number, one cannot verify whether a specific entry of the database corresponds to him (or her), without the consent (and participation) of the blinding entities. The computation of any of the $f_i(\mathcal{B})$ requires a threshold of the blinding entities, and without their cooperation, the entry will be indistinguishable from three random values and a ciphertext under an unknown random key.

Moreover, given secure hardware for the sampling, and the execution of the secure multi-party computation, then the biometric data, $\mathcal{B}$, never leaves the acquisition system, and thus even a malicious blinding entity cannot obtain this information.

### 3.3 Fraud Detection

Here we describe the approach for CollisionQuery. When the same person is added to the database with two different $id$'s but the same biometric information $\mathcal{B}$, there will be a collision in the first field. The other three fields are

$$f_r(\mathcal{B}) \cdot f_t(\mathcal{B})^{f_z(id)}, f_z(id), E_{f_r(\mathcal{B})}(id)$$

and

$$f_r(\mathcal{B}) \cdot f_t(\mathcal{B})^{f_z(id')}, f_z(id'), E_{f_r(\mathcal{B})}(id')$$

for the two ID numbers $id$ and $id'$ that the person "possesses".

Given the fact that $f_z(id)$ and $f_z(id')$ are both known (from the second field in each entry), we can compute

$$\left(f_r(\mathcal{B}) \cdot f_t(\mathcal{B})^{f_z(id)}\right)^{-f_z(id')} \cdot \left(f_r(\mathcal{B}) \cdot f_t(\mathcal{B})^{f_z(id')}\right)^{f_z(id)} = f_r(\mathcal{B})^{f_z(id)-f_z(id')}$$

over the field $Z_p$, from which it is possible to retrieve $f_r(\mathcal{B})$, which can be used to deduce the respective ID numbers from the fourth field. We emphasize that this can done by anyone with access to the database - it does not require any involvement from the blinding authorities.

We note that there are cases where different sampling of the same person results in different biometric measurements, $\mathcal{B}$ and $\mathcal{B}'$. While these are to be

avoided as much as possible, we note that as long as there is a very high probability that the same person would suggest the same biometric data (i.e., the consistency of extracting $\mathcal{B}$ from the person is high enough), then most of the fraud attempts will be caught. Even a 95% success rate in identifying the second acquisition should be sufficiently high to scare away fraud attempts (given a severe enough legal sanction).

### 3.4 Lost ID Cards

Once an ID card is lost, the person holding it has to provide his ID number and request a new ID card. This is done by performing the acquisition process a second time. Of course, the person has to announce that this is the case, as otherwise a collision will be detected and the person may be accused of fraud. Here we use the VerifyEntry protocol.

We note that if the person gave the right ID number, then the entire entry related to him should be duplicated (up to the signatures), so the acquisition station will verify that the entry has already been recorded (up to the signatures) and then issue a new card with the same ID.

As stated before, if the person gives a different ID number, a fraud will be detected, and it would be possible to further investigate the event.

### 3.5 Inverse Retrieval Queries

In the cases where the identity of a person needs to be extracted from the biometric data $\mathcal{B}$, given a court order, the blinding entities would run the ReverseLookup protocol. Essentially this can be implemented by having the blinding entities run the acquisition protocol with a bogus id number (for example, the all zeros, or an ID number that does not conform to the 9-digit standard). At this point, the state can perform a collision search query, and find the colliding IDs — the real ID number and the bogus one.

It is easy to see that this protocol succeeds in identifying the person, and that it cannot be executed without the consent of the required threshold of blinding entities. Thus, this delicate query is unlikely to be misused (under the assumption that the blinding entities are to be trusted).

### 3.6 Instantiating the cryptographic tools

Here we focus on identifying the pseudo-random function $f_y(x)$ – this choice will then determine which MPC and secret sharing protocols we use.

The proposed scheme uses a pseudo-random function $f_y(x)$. Our requirements are that given $f_y(x)$ (or even several of these) it would be impossible to learn anything about either $x$ nor $y$, even given $x$, which follows directly from the pseudo-randomness property, and thus will hold for any PRF. Ideally, we would like this to hold even when $y$ is revealed, to guarantee some security even when the blinding authorities are compromised. In this case, we cannot prevent the

authorities from testing various $x$'s against $f_y(x)$, but we would like to ensure that if it is hard for the adversary to guess $x$, then it will be hard to extract $x$ given $f_y(x)$.

*Approach 1: the random oracle model.* It is easy to see that if $f_y(x)$ is implemented using a random oracle $\mathcal{O}$, the definition of $f_y(x) \triangleq \mathcal{O}(x, y)$ guarantees the above security requirements.[7] While random oracles may be used in proofs, they are not instantiable in practice, and one has to resort to hash functions. Hence, we can define $f_y(x) \triangleq h(x, y)$ for some good hash function $h(\cdot)$. As long as the hash function is preimage resistant and secure against partial message recovery, then so does our proposed scheme.

While this allows for a secure implementations, we feel that the current state of the art in hash functions, both in the practical sense, and the theoretical foundations, may put this option at a disadvantage. This is mainly due to the introduction of new cryptanalysis techniques, as well as the debate concerning the exact security definitions for hash functions.

*Approach 2: AES + Davies-Meyer* Another option is to use a compression function which is based on a block cipher in a Davies-Meyer mode. For example, one could use $f_y(x) = AES_x(y) \oplus y$. While not provably secure (in the context of partial recovery attacks), Davies-Meyer is considered a good heuristic, which ensures that as long as AES is a secure block cipher, then $f_y(x)$ is pseudo-random. Furthermore, Davies-Meyer is considered a good heuristic to achieve optimal security even when the key $y$ has been revealed. We note that there is some discrepancy between the formal definitions of what we are after and the security of a Davies-Meyer compression function which relies on AES as an ideal pseudo-random permutation. At the same time, any attack on this compression function is expected to break at least one of AES' security requirements, and thus, this seems like a reasonable option for the implementor.[8]

We note that [DK09] gives an MPC protocol for computing AES. They show a 4-party protocol with up to one malicious adversary takes 7 seconds to encrypt one block using AES-128 (in a Python implementation). The time is given for an instance where the key has already been shared, but includes the time to distribute shares of the message. The communication overhead is roughly 140KB.

[7] We note that as the computed values are not randomized, in theory, an adversary who obtains $y$ can exhaustively search for the corresponding $x$ for a given entry (especially if $x$ contains little entropy). This inherent problem prevails as long as the entries do not contain randomness, which as we discuss later, seems to contradict the ability to detect mis-use. Hence, we can only hope to force an adversary who obtained $y$ to exhaustively search the possible $\mathcal{B}$ before extracting $x$ from a given record.

[8] We note that while several attacks on the full AES-192 and AES-256 may contradict these security reasoning, and actually offer some results in the security of AES-256 used in a Davies-Meyer mode in [BK09,BKN09] one can easily solve these issues by adding more rounds to AES.

Hence, our multi-party computation can be done relatively efficiently based on this Damgård-Keller multi-party computation.[9]

*Approach 3: Number theoretic constructions* Finally we note that some number theoretic constructions of pseudo-random functions (e.g. [Nie02,DY05]) may allow for more efficient multi-party computation. We chose here to focus on symmetric key constructions because they allow us, at least heuristically, to guarantee some security in the case where all of the blinding authorities' keys are compromised.

### 3.7 Possible Honest Collisions in the Biometric Data

Our proposed solution uses a few sampling stations. Hence, we can assume that these stations can extract sufficient entropy from $\mathcal{B}$ in a consistent manner. While it is highly unlikely that the full biometric data collides, there is a chance that the extracted part, $\mathcal{B}$, collides for two different persons.[10] In these cases, the CollisionQuery operation will identify these two individuals as a fraud attempt. Hence, we devise a method to resolve this issue.

The base of our proposal to deal with these collisions, which we refer to as "honest" collisions, is to introduce a second sampling method (hereandafter referred to as "procedure 2"). The new method may use the same sampling equipment with different parameters in extracting $\mathcal{B}$, or using a completely different equipment.

Given that a "honest" collision will trigger an investigation, calling the two persons to interrogation, it will be able to identify this collision as honest. Once the investigation concludes that the collision has occurred in the biometric data sampled from two different persons, we suggest to invoke the following procedure:

- Given the collision in $f_s(\mathcal{B})$, both entries are removed and replaced with the entry $< f_s(\mathcal{B})$, "*collision*" $>$. Note that this entry does not contain the IDs of the involved entities.

[9] The acquisition station will first distribute shares of $\mathcal{B}$ and $id$ to all blinding authorities. The AES multi-party computation will then be run twice to compute $f_s(\mathcal{B})$ and $f_z(id)$. It will then be run twice more to compute $f_t(\mathcal{B})$ and $f_r(\mathcal{B})$, but this time instead of immediately combining their shares to reconstruct these values, the authorities will use these shares to compute $f_r(\mathcal{B}) \cdot f_t(\mathcal{B})^{f_z(id)}$ and (running the AES multi-party computation one more time) $E_{f_r(\mathcal{B})}(id)$.

This protocol should take less than a minute (for 4 blinding authorities where at most one is assumed to be dishonest), which is well within reasonable limits given that we need to do this only once during the biometric acquisition and ID card generation process. (We note that this would also work for the random oracle approach mentioned above.)

[10] We note that the number of Israeli citizens is less than 8,000,000, hence, 46 bits of entropy in $\mathcal{B}$ should be sufficient to offer a sparse enough distribution of biometric samples.

– Both persons will be processed again, this time using a different biometric sampling method (producing different biometric samples $\mathcal{B}'$), with the new respective eateries will be inserted into the database.

We note that "colliding" people, will just need to remember that their biometrics are to be measured using "procedure 2", a fact that will be embedded into their ID cards.

When such a person loses his identity card he has to announce the fact that procedure 2 is to be used, thus allowing for an identification as in the other cases. (If he does not, the entry $< f_s(\mathcal{B}), \text{“}collision'' >$ will be found, which will remind him or her to use "procedure 2").

If at some point this person tries to obtain a second ID card, his biometric will first be sampled with the standard procedure, and the insertion into the database will cause a collision with the $< f_s(\mathcal{B}), \text{“}collision'' >$ entry, and this will be identified as potential fraud and further investigated.

### 3.8 Some Caveats

The main disadvantage we could identify with respect to our solution is the fact that it relies on untampered and secure hardware. However, we note that the only hardware that has to be agreed as secure is the hardware that does the real sampling of the biometric data. The rest of the computing devices can be built by each blinding entity on its own. Of course, it would be wise if they verify and secure the hardware to the maximum, but given that the secure multi-party protocol can handle a small number of "problematic" devices, this seems to be of a lesser effect.

As we discuss below, we assume that the hardware in use is trusted to perform the required operations (this is to be verified using publicly verified design, as recommended for voting machines). Of course, one can never be 100% sure that this is the case, but given the "common" approaches promoted by the e-voting community, we could reach good enough coverage, especially given the fact that secure multi-party computation can sustain some threshold of illegitimate behavior. Moreover, this seems be an inherent problem of any possible solution, which must rely on the correct operation of the hardware.

We note that our solution assumes the ability to extract biometric information in a somewhat reliable fashion. This can be achieved by using high quality sampling equipment in the acquisition stations. As the number of acquisition stations is relatively small, we can safely assume that increasing the accuracy of the scans by purchasing better equipment is acceptable.

While most of the biometric sampling algorithms assume the existence of some helper string to allow correcting distorted samples, e.g., in [DORS08], it is not clear how we could use such helper strings in our setting. (A straightforward application would need to store a different helper string for each person, and it is not clear how this could be done without yet another privacy-compromising

database.)[11] However, using a combination of distortion correction algorithms such as [CCHW97,CTYZ06], one can easily exploit the 140 entropy bits[12] in iris scans and 58 bits of entropy in face recognition (along the biometric information of the fingerprint) to allow a resilient sample of enough entropy. Furthermore, as we can tolerate some number of honest collisions, we also have the option of beginning with very basic information (for example unique information that can be derived from fingerprints), and then using iris/face scans only in the case of a collision.

## 4 Summary and Conclusions

In this paper we have suggested a construction for a privacy-preserving biometric database. This database can be implemented such that even if the entire contents is leaked, it reveals no biometric information, and does not even allow for membership tests (i.e., whether a given person is in the database or not). Such a solution may be deployed in many countries, e.g., Israel were the issue was first raised or South Africa, where there is a lot of identity thefts.

The trust assumptions in our suggestion are relatively small, or almost as minimal as possible. Specifically, one needs to trust that the blinding entities do not collude and that the hardware is trustworthy (which will be an issue with any solution).

## Acknowledgments

We would like to thank Benny Applebaum, Eli Biham, Yaniv Carmeli, Yoram Oren, Alon Rosen, Adi Shamir, and Erez Waisbard, for the fruitful discussions and their comments. We would also like to thank the anonymous referees who commented on an earlier versions of this paper.

## References

[ANS98] Ross J. Anderson, Roger M. Needham, and Adi Shamir. The Steganographic File System. In David Aucsmith, editor, *Information Hiding*, volume 1525 of *Lecture Notes in Computer Science*, pages 73–82. Springer, 1998.

[ARF+10] Benny Applebaum, Haakon Ringberg, Michael J. Freedman, Matthew Caesar, and Jennifer Rexford. Collaborative, Privacy-Preserving Data Aggregation at Scale. In Mikhail J. Atallah and Nicholas J. Hopper, editors, *Privacy Enhancing Technologies*, volume 6205 of *Lecture Notes in Computer Science*, pages 56–74. Springer, 2010.

[11] We suggest that finding a secure and private way to use these helper strings in our setting is an interesting open problem.

[12] These figures are based on the claims in [CS07].

[BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public Key Encryption with Keyword Search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and Related-Key Attack on the Full AES-256. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 231–249. Springer, 2009.

[Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS*, pages 136–145, 2001.

[CCHW97] Shih-Hsu Chang, Fang-Hsuan Cheng, Wen-Hsing Hsu, and Guo-Zua Wu. Fast algorithm for point pattern matching: Invariant to translations, rotations and scale changes. *Pattern Recognition*, 30(2):311–320, 1997.

[CS07] Ann Cavoukian and Alex Stoianov. Biometric Encryption. *Biometric Technology Today*, 15(3):11, 2007.

[CTYZ06] Xinjian Chen, Jie Tian, Xin Yang, and Yangyang Zhang. An algorithm for distorted fingerprint matching based on local triangle feature set. *IEEE Transactions on Information Forensics and Security*, 1(2):169–177, 2006.

[DK09] Ivan Damgård and Marcel Keller. Secure Multiparty AES. Technical report, IACR ePrint Report 2009/614, 2009. available online at http://eprint.iacr.org/2009/614.

[DNW09] Ivan Damgård, Jesper Buus Nielsen, and Daniel Wichs. Universally Composable Multiparty Computation with Partially Isolated Parties. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 315–331. Springer, 2009.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A Verifiable Random Function with Short Proofs and Keys. In Serge Vaudenay, editor, *Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 416–431. Springer, 2005.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC*, pages 218–229. ACM, 1987.

[Gol01] Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[MK99] Andrew D. McDonald and Markus G. Kuhn. StegFS: A Steganographic File System for Linux. In Andreas Pfitzmann, editor, *Information Hiding*, volume 1768 of *Lecture Notes in Computer Science*, pages 462–477. Springer, 1999.

[MNPS04] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay - Secure Two-Party Computation System. In *USENIX Security Symposium*, pages 287–302. USENIX, 2004.

[Nie02] Jesper Buus Nielsen. A Threshold Pseudorandom Function Construction and Its Applications. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 401–416. Springer, 2002.

[Sha79] Adi Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.
[Yao86] Andrew Chi-Chih Yao. How to Generate and Exchange Secrets (Extended Abstract). In *FOCS*, pages 162–167. IEEE, 1986.

## A Cryptographic Tools

We use the following cryptographic primitives[13] in our solution:

**Pseudo-random function (PRF):** A keyed function $F$ is said to be a PRF if, for a random secret key $s$, $F_s$ is indistinguishable from a random function. I.e. even if $F_s$ is evaluated on many arbitrary inputs $x$, the results will appear to be independent random values. PRFs are widely used in cryptography, e.g., as message authentication codes.

**Secret sharing:** A secret sharing scheme [Sha79] is a scheme for distributing secrets to a set of entities such that any authorized subset can reconstruct it, while any other subset, does not obtain a single bit of information about the secret. Shamir's secret sharing scheme [Sha79], defines a threshold $t$ of parties out of the $n$ different parties. After sharing the secret, any $t$ out of $n$ of these parties can combine their shares to reconstruct the secret. At the same time, any set of less than $t$ parties will not be able to learn any information about the secret.

**Secure multi-party computation (MPC):** Secure multi-party computation [GMW87,Yao86] allows several mutually distrusting parties to jointly perform some computation in a secure manner. In particular, it guarantees that no matter how the other parties behave, each honest player's input remains secret (all parties only learn the final output of the function), and that if the honest party obtains a result, he can be certain that this result is correct. We will consider protocols which are secure even when a certain fraction of the parties are arbitrarily malicious.

Recent years have seen significant progress in making multi-party computation practical. Simple arithmetic and boolean operations can be performed extremely efficiently [MNPS04,DNW09]. In some cases more complex operations can also be performed — a recent result by Damgård and Keller [DK09] gives a reasonably practical [14] multi-party protocol for AES, in which each party's input includes a share of the message and a share of the key, and each party receives a share of the result.

**Universal Composability (UC):** Secure multi-party computations are often run in environments where more than one concurrent execution happens. As some secure multi-party computation protocols may break in these settings, the framework of universal composability was suggested in [Can01] to develop protocols that can be run in as many executions as needed. Hence, a

[13] See [Gol01] for more formal definitions.

[14] A 4-party protocol with one malicious adversary takes 7 seconds to encrypt 1 block when implemented using the script language Python. (This assumes that the key has already been shared, but includes the time to distribute shares of the message.) The communication overhead is roughly 140KB.

protocol which is universal composable, can be run in as many instances in parallel as needed, without losing any security.

## B Alternatives

We briefly discuss other concepts for alternate solutions, and why we believe that they do not fit.

- Secret sharing the entire database and keeping a share at each government branch or blinding entity. This will not provide much security if we must reconstruct the database in order to perform each query — once the database is reconstructed, the knowledge of the full database may leak (and might not be "forgotten"). Alternatively, one could use multi-party computation each time to compute a query on the secret-shared database, however this would be extremely inefficient, as the protocol would have to act over the entire database to ensure privacy.
- Using only government branches in place of blinding entities — in this case, the trust model assumes that the different branches of the government do not collude. Of course, in the case of a regime change, this assumption may no longer be valid. Moreover, it seems that in this case, it may be easier to access the database without any accountability or tractability.
- Using one central database but storing only a digest of the biometric data — this allows anyone with access to the database to query whether a person is in the database or not (preventing Israelis from denying their citizenship).
- Storing $< f_s(\mathcal{B}), id >$ in the database, where $f_s$ is computed by blinding entities — This solution clearly reveals the ID numbers, and allows the government to forge new entries.
- Storing $< f_s(\mathcal{B}), f_r(id) >$ in the database — this will not allow the state to retrieve the IDs of fraudulent people (note that $f_r(\cdot)$ is not necessarily invertible).
- Storing $< f_s(\mathcal{B}), E_r(id) >$ in the database — the identification of the fraudulent people requires the involvement of the blinding entities (which increases the trust and required from them, and the computational costs involved).
- Adding randomness to each entry — while such addition may allow perfect forward secrecy (in the sense that even after the secret keys of the blinding entities are found, the adversary would need to exhaustively search a large space of possible random values to extract biometric information fro the entry), it seems to contradict the ability to efficiently identify frauds without the help of the blinding entities.
- Using privacy preserving data aggregation [ARF+10] — while privacy preserving data aggregation offers a solution to the problem at hand, it is not designed to deal with entries which arrive at a very slow rate, thus not offering the required privacy (as the proxies, which are the equivalent of the blinding entities in our solution) cannot mix a sufficient amount of entries.