Multisketches: Practical Secure Sketches Using Off-the-Shelf Biometric Matching Algorithms

Rahul Chatterjee
UW–Madison

M. Sadegh Riazi
UCSD

Tanmoy Chowdhury
GMU

Emanuela Marasco
GMU

Farinaz Koushanfar
UCSD

Ari Juels
Cornell Tech

ABSTRACT

Biometric authentication is increasingly being used for large scale human authentication and identification, creating the risk of leaking the biometric secrets of millions of users in the case of database compromise. Powerful "fuzzy" cryptographic techniques for biometric template protection, such as secure sketches, could help in principle, but go unused in practice. This is because they would require new biometric matching algorithms with potentially much diminished accuracy.

We introduce a new primitive called a *multisketch* that generalizes secure sketches. Multisketches can work with *existing biometric matching algorithms* to generate strong cryptographic keys from biometric data reliably. A multisketch works on a biometric database containing multiple biometrics — e.g., multiple fingerprints — of a moderately large population of users (say, thousands). It conceals the correspondence between users and their biometric templates, preventing an attacker from learning the biometric data of a user in the advent of a breach, but enabling derivation of user-specific secret keys upon successful user authentication.

We design a multisketch over tenprints — fingerprints of ten fingers — called TenSketch. We report on a prototype implementation of TenSketch, showing its feasibility in practice. We explore several possible attacks against TenSketch database and show, via simulations with real tenprint datasets, that an attacker must perform a large amount of computation to learn any meaningful information from a stolen TenSketch database.

CCS CONCEPTS

• **Security and privacy** → *Biometrics*;

ACM Reference Format:
Rahul Chatterjee, M. Sadegh Riazi, Tanmoy Chowdhury, Emanuela Marasco, Farinaz Koushanfar, and Ari Juels. 2019. Multisketches: Practical Secure Sketches Using Off-the-Shelf Biometric Matching Algorithms. In *Proceedings of 2019 ACM SIGSAC Conference on Computer and Communications Security, London, United Kingdom, November 11–15, 2019 (CCS '19),* 16 pages.
https://doi.org/10.1145/3319535.3363208

CCS '19, November 11–15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6747-9/19/11…$15.00
https://doi.org/10.1145/3319535.3363208

1 INTRODUCTION

Human authentication to computers has traditionally relied mainly on passwords. In recent years, the usability and security shortcomings of passwords have prompted a rise in the popularity of biometric authentication. Biometrics, however, presents a big challenge: It is hard to store biometric data in databases securely.

The problem is that biometric measurements are *noisy*. Two scans of the same physiological feature (finger, eye, etc.) almost always differ due to changes in reading conditions. Also, biometric features change subtly over time. Biometric matching in practice, therefore, involves measuring the *similarity* between a *template*, an explicit, registered set of biometric features, and a biometric measurement taken during authentication. A conventionally presented password, in contrast, is only accepted if it *exactly matches* a registered one. Passwords can therefore be hashed to protect against capture in a breach, but biometrics cannot.

To protect biometrics in a manner similar to hashing, researchers have long explored various forms of *fuzzy cryptography* [27, 28, 40]. In [28], Dodis et al. present the notion of *secure sketches* and *fuzzy extractors*, which attempt to extract high entropy keys reliably from noisy sources of data such as biometrics, but fall short in practice for two reasons. First, individual biometrics often have low effective entropy; e.g., TouchID claims 1:50,000 false acceptance rate, which implies at most 16 bits of entropy per finger, and maybe lower [34]. Second, a fuzzy extraction system must be custom-built for a particular distance metric. Doing so for the complicated distance metrics associated with many biometrics in practice, e.g., fingerprints, and achieving both rigorous security properties and good matching accuracy, remains an open challenge [38, 51]. To the best of our knowledge, despite almost two decades of research, nowhere is fuzzy cryptography used to protect biometrics in practice.

Techniques such as secure function evaluation and secure multiparty computation have been proposed for private queries against a biometric database [31, 53, 64], but do not protect against a database breach. Secure hardware, such as hardware security modules (HSMs)[1] and trusted execution environments (TEEs) can, in theory, offer strong protection for biometrics: They permit templates to be stored in a database in encrypted form and decrypted only in hardware to perform matching. Recent attacks against Intel SGX [42, 45, 59], however, highlight the fact that trusted hardware is far from bulletproof, and other security approaches are needed. Happily, the techniques we present in this paper are complementary to and usable in concert with trusted hardware as well as secure function evaluation.

[1] Mobile devices, for which biometric authentication is popular today, generally protect templates using trusted hardware [4, 9].

In this work, we introduce and formalize a new generalization of secure sketches called *multisketches*. A conventional secure sketch operates on individual biometric templates (e.g., a single fingerprint). In contrast, a multisketch operates on a batch of templates belonging to multiple users. While in secure sketches biometric data is concealed by performing lossy compression of a template, in a multisketch system unmodified templates are used, and security is achieved by concealing the correspondence between templates and users.

We present a specific multisketch construction that can use *off-the-shelf* biometric matching algorithms. Our multisketch construction thus realizes a long-elusive goal: Secure, practical derivation of strong cryptographic keys from users' biometric data.

Two notable requirements of multisketches are that they require registration of *multiple biometrics* by individual users and that they can only protect *databases* containing the biometrics of *many users*. Because individual biometrics lack adequate usable entropy to derive strong cryptographic keys, such use of multiple biometric readings is unavoidable in any biometric fuzzy-extraction system. Providing multiple biometrics for everyday authentication can be cumbersome, but for high security (and infrequent) operations, such as credential recovery or border crossing, they can be deployed. Tenprints, in particular, are commonly enrolled today in border control [6, 18], law enforcement [1], and social-service delivery databases [3]. These databases often have millions of users, as in India's Aadhaar system, with its over one billion enrolled users [3].

Data compromise is a serious concern for such databases containing sensitive biometric data of millions of users. The pressing need for protecting templates in such systems, and failure of existing fuzzy cryptographic schemes to provide it, motivate our work.

Main idea. In our multisketch construction, as applied to a database D, each user u registers with a set of n distinct biometric templates $\mathbf{w} = \mathbf{w}_1 \ldots \mathbf{w}_n$. The goal is to be able to recover $\mathbf{w}$ from D by providing new biometric readings $\tilde{\mathbf{w}} = \tilde{\mathbf{w}}_1 \ldots \tilde{\mathbf{w}}_n$, as long as most of $\tilde{\mathbf{w}}$ matches with $\mathbf{w}$.

To prevent an adversary that breaches D from learning $\mathbf{w}$, our construction *conceals which templates are associated with which user* and *randomly orders* all templates of all users in D. This is the pivotal idea in our construction. This randomization means an adversary cannot easily pick out sets of associated templates, but must attempt to reconstruct $\mathbf{w}$ via brute-force search over all possible conjectured sets $\mathbf{w}'$. Given enough users (say, thousands), such search is computationally infeasible in practice.

To reconstruct her set of templates $\mathbf{w}$, a user u presents a set of fresh biometric readings $\tilde{\mathbf{w}} = \tilde{\mathbf{w}}_1 \ldots \tilde{\mathbf{w}}_n$. An (off-the-shelf) biometric matching algorithm is then used to match each $\tilde{\mathbf{w}}_i$ to its corresponding, most similar template $\mathbf{w}'_i$ in D, yielding a conjectured template set $\mathbf{w}'$. Assuming that enough matches are successful (even if some are erroneous), i.e., $\mathbf{w} \,\Delta\, \mathbf{w}' = |\mathbf{w} \cup \mathbf{w}' - \mathbf{w} \cap \mathbf{w}'|$ is small, u can recover $\mathbf{w}$.

A secure sketch — and thus our multisketch scheme — can easily be converted into a *fuzzy extractor*, allowing $\mathbf{w}$ to be converted into a strong cryptographic key κ that can then be used for authentication, signing, or any other purpose.

The performance of multisketch largely depends on that of the underlying biometric matching algorithm. During recovery, the matching algorithm might need to be applied to all templates present in D to identify the closest matches. This can be expensive for large D. However, the search process can easily be parallelized by virtually any factor. For instance, a recent study shows how to perform 9 million matches per second using a single graphics processing unit (GPU) [19]. Similarly, matching accuracy can also be enhanced by using more accurate (proprietary) biometric matching algorithms. Being able to use any biometric matching algorithm as per the need of the application is one of the key contributions of multisketches. For scientific reproducibility, one of our key methodological choices for this paper and its proof-of-concept is to rely on an open-source matching algorithm, standard fingerprint datasets, and commodity hardware to measure matching accuracy and performance. We elaborate on the performance and limitations of our implementation of multisketch in Section 4.2.

The problem of correlations. If the templates in $\mathbf{w}$ are uncorrelated, i.e., an adversary cannot tell which sets of templates belong to the same user with probability better than random guessing, then the attacker must search *all* possible conjectured templates $\mathbf{w}'$. However, one major challenge in practice is that biometric templates from the same user might be correlated. As a simple example, gender correlates significantly with the appearance of fingerprint templates [49]. Conversely, it is not reasonable to expect that an adversary can directly determine the joint probabilities of biometric tuples, given the complexity of the distribution from which they are drawn. Much of our work in this paper evaluates empirically how effectively an adversary can correlate the fingerprint templates of individual users in practice. We experimentally explore plausible strategies for attacking our multisketch construction. Even with the best attack, we show that TenSketch with $N = 10^4$ users provides security equivalent to 70-bits in practice, assuming (pessimistically) that the attacker already knows one of the user fingerprints.

Contribution. Our contributions are as follows.

- We introduce and formalize *multisketches*, a generalization of secure sketches.
- We construct TenSketch, a multisketch for tenprints that works with off-the-shelf biometric matching algorithms.
- We study correlations among users' fingerprint templates and analyze the security of TenSketch given classifiers that can identify these correlations.
- Finally, we report on the security and matching functionality of TenSketch, demonstrating its practicality.

We will open-source our prototype implementation of TenSketch with the final version of the draft.

2 OVERVIEW OF MULTISKETCH

In this section, we present a simple example to illustrate how multisketches may be used in the biometric database setting. This example is a simplification. It does not reflect the detailed mechanics of our full multisketch scheme (given in Section 4), but is meant to convey the basic intuition. Consider a database with $N = 3$ users, each with $n = 5$ fingerprints. Templates corresponding to these fingerprints are shown conceptually in Figure 1.

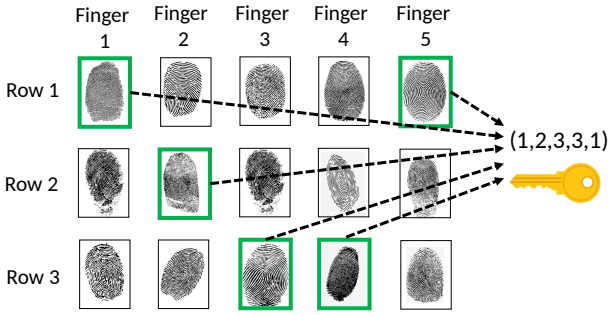

Figure 1: Conceptual diagram of multisketch scheme. Templates corresponding to each finger position are stored in random order. During authentication, presented finger images are matched to templates; corresponding row indices — e.g., here (1, 2, 3, 3, 1) — are used to compose the user's key.

A critical feature of our multisketch construction, shown in Figure 1, is that templates for each finger position are ordered *randomly*, and with *no user identifiers*. User 1's templates, for example, are scattered across different rows. Thus an adversary that breaches the database has to determine which template corresponds to which user. As we show via attack simulations, determining template associations is hard, and consequently, an attacker cannot feasibly identify which fingerprints belong to a particular user. In contrast, in an ordinary biometric database, templates are associated with user identities. For example, all of a given user's templates might be in the same row. So an adversary that breaches the database can immediately obtain sensitive biometric information of the users.

Given the database of our construction, users can derive keys from their templates at the time they authenticate. Conceptually, this process can be very simple: Each of the user's fingers is matched against all templates for the corresponding finger position (e.g., index finger). In an ideal setting, the closest match is the template belonging to that user.

Given ideal matching of this kind, the indices of the matched templates (which are random, of course) can be used to compose a key for the user. In our simple example in Figure 1, the key of one particular user is composed of the best matching row indices $(1, 2, 3, 3, 1)$ — one per finger position. In Section 4 we show how to do this composition. We also show given a sufficiently large database, our construction in Section 4 can produce high-entropy keys from tenprints, that can be used for public-key authentication, signing, decryption, etc.

Assuming that templates are independently distributed, i.e., have uncorrelated features, an adversary that has breached the database and tries to reconstruct *any* key faces a combinatorial explosion. In our example, the adversary can start with a given template and try to find its four $(n-1)$ associated ones. This requires searching a space of size $N^{(n-1)}$. For large N this can be very large. For example, if the database has $N = 10^4$ users and $n = 5$ as before, an attacker would need to try 10,000 trillion ($\approx 2^{53}$) combinations.

In practice, however, the naïve approach that we've just illustrated runs into two main complications.

Errors. First, matching biometrics in practice isn't ideal, and is subject to a range of errors resulting from the noisy nature of biometrics. A finger may match against the wrong template (belonging to a different user) for its corresponding position. Thus, some form of *error correction* is required to recover the key. As we explain in Section 4, we accomplish this in our complete multisketch scheme using an existing set distance secure sketch construction [29].

Correlations. The second problem is our simplistic security analysis above that assumes templates are *uncorrelated*, i.e., the templates of a given user are no more similar to one another than to those of other users. In practice, this isn't true; as just one example noted above, fingerprint appearance correlates with gender [49].

It is hard to obtain a principled upper bound on these correlations. However, for our construction, there is strong empirical evidence that the correlations among the templates of a user are relatively weak. In Section 6, we explore a variety of adversarial classifier constructions based on deep neural networks (DNNs) to classify pairs (and tuples) of fingerprint templates as belonging to the same user or not. The performance of these DNNs — false positive and false negative rates — is too poor to advantage an adversary significantly, as we show in Section 7. (The adversary gains a 9-bit advantage against a conservatively parameterized version of our TenSketch construction.)

When our multisketch construction is made error-tolerant and correlations are taken into account, we find that users can construct keys with relatively high entropy using fingerprints alone — achieving, e.g., 58-bit security in a version of TenSketch (which can be significantly boosted via a standard key-derivation function [21]). Of course, it is also possible to add additional biometrics (iris codes, voiceprint, etc.) to achieve stronger security and hedge against better correlation techniques.

3 PRELIMINARIES

We begin our discussion with some preliminary discussion of the notations and background on secure sketches.

Message spaces. We refer to the data in a biometric reading as a *message*. We denote a message space using capital letters and its members using lowercase letters. For example, W is a set of messages, while w denotes an element in W. We denote a vector or tuple using bold fonts; the i^{th} entry in a tuple is denoted by subscript i. For instance, a tuple of messages is denoted by $\mathbf{w}$, and the i^{th} entry in that tuple by $\mathbf{w}_i$. We use $\mathbf{w}_{i\ldots j}$ to denote a subset of the tuple $\mathbf{w}$, for $1 \leq i \leq j \leq |\mathbf{w}|$.

We explore settings in which users authenticate utilizing multiple biometric readings. These readings may be *multimodal*, i.e., involve different physiological feature types (e.g., fingerprint, iris scan, hand geometry, etc.), or may involve multiple instances of the same feature type (e.g., multiple fingerprints, two iriscodes, etc.). It is helpful to view such instances as distinct physiological features; for example, index fingers look different, in general, than thumbs, i.e., they are drawn from different distributions. Thus, we treat each such reading as coming from a distinct space W_i with probability distribution p_i. We denote a multi-template message-space by $\mathbf{W} \subseteq W_1 \times \ldots W_n$ and a message tuple in this space by $\mathbf{w} = \mathbf{w}_1 \ldots \mathbf{w}_n$. Let $\mathbf{p}(\mathbf{w})$ denote the joint probability of a message tuple $\mathbf{w} \in \mathbf{W}$.

As noted above, different biometric attributes of a given person are not in general mutually independent, i.e., unfortunately, we cannot for simplicity assume $\mathbf{p}(\mathbf{w}) = \prod_i p_i(\mathbf{w}_i)$. Moreover, we assume that an adversary does not know $\mathbf{p}$ and cannot directly compute joint probabilities. We instead consider the various estimation approaches an adversary might adopt in trying to identify users' tuples in the database D.

Biometric readings, as mentioned earlier, are always noisy; multiple reading of the same biometric template of a user produces slightly different outputs. Let e be a noise function that model the errors incorporated in the process of reading a biometric template. Therefore $\tilde{w} \leftarrow\$\ e(w)$, denotes the reading $\tilde{w}$ obtained while reading the template w. As the biometric readings and biometric templates are normally represented using similar message formats in practice, we will use the word *message* to denote either of them.

Biometric matching. A *biometric matching algorithm* $\mathcal{L}(w, \tilde{w})$ is an efficient algorithm that takes as input a template w and a biometric reading $\tilde{w}$ and outputs a *matching score* $\theta \in [0, 1]$ denoting how likely a biometric reading $\tilde{w}$ is indeed a reading of the template w. By setting a predetermined threshold τ, a score θ output by algorithm $\mathcal{L}$ can be converted into an `accept` (if $\theta \geq \tau$) or `reject` (if $\theta < \tau$) output.

The effectiveness of a matching algorithm is measured in practice by its false matching rate (FMR) and false non-matching rate (FNMR). For a given τ, a matching algorithm's FMR, denoted by α, is defined as $\Pr[\, \mathcal{L}(w, \tilde{w}) = \texttt{accept};\ \tilde{w} \leftarrow\$\ e(w'); w' \neq w\,]$, where w and w' are sampled randomly from W. The FNMR, denoted by β, is defined as $\Pr[\, \mathcal{L}(w, \tilde{w}) = \texttt{reject};\ \tilde{w} \leftarrow\$\ e(w)\,]$, where w is sampled randomly from W.

As noted before, multisketch can be extend to multi-modal biometric templates. In that case, each message space W_i can have separate matching procedures $\mathcal{L}_i$. Multisketch only requires matching algorithms $\{\mathcal{L}_i\}$ with low FMR and FNMR rates, but requires no assumptions on how the template similarity is measured.

Biometric matching algorithms can be extended to handle message tuples. For example, two message tuples can be matched by counting the number of message pairs among them that are accepted by the underlying matching algorithm $\mathcal{L}$. Let $\mathbf{L}_t$ denote the matching algorithm defined over message tuples; on input $\mathbf{w}$ and $\tilde{\mathbf{w}}$, it outputs `accept` if at least t pairs between $\mathbf{w}$ and $\tilde{\mathbf{w}}$ are accepted by $\mathcal{L}$, `reject` otherwise.

Secure sketches. Secure sketches (SS) [29] are primitives that perform error correction on a message without leaking much information about the message. Secure sketches are defined for a message space W with distance function d.

Definition 3.1. A $(W, \mu, \mu', \eta, \varepsilon)$-*secure sketch* is a pair of algorithm *sketch* (SS) and *recover* (Rec), defined as follows.

- The algorithm SS (possibly randomized), on input $w \in W$ outputs a public bit string $v \in \{0,1\}^*$.
- The deterministic algorithm Rec, on input the string v and a message $\tilde{w}$, outputs w, should w and $\tilde{w}$ are similar, else the output is indeterminate.
- **Correctness.** Correctness requires that for all messages w and $\tilde{w}$ such that $d(w, \tilde{w}) \leq \eta$, (where η is a parameter to the construction), $\Pr[\, \mathsf{Rec}(v, \tilde{w}) = w; v \leftarrow\$\ \mathsf{SS}(w)\,] \geq 1 - \varepsilon$.

$\underline{\mathsf{SS}^{\Delta}_{t}(\mathbf{w})}$:
$\kappa \leftarrow\$\ \mathsf{K}(1^{\ell})$; $\mathbf{h} \leftarrow \varnothing$
$c \leftarrow\$\ \mathsf{E}_{\kappa}(\mathbf{w})$
for $\tilde{\mathbf{w}} \in \binom{\mathbf{w}}{t}$ do
  $\mathbf{h} \leftarrow \mathbf{h} \cup \mathsf{E}_{\mathsf{H}(\tilde{\mathbf{w}})}(\kappa)$
$v \leftarrow (\mathbf{h}, c)$
return s

$\underline{\mathsf{Rec}^{\Delta}_{t}(\mathbf{w}', v)}$:
parse v as $(\mathbf{h}, c)$
for $(\tilde{\mathbf{w}}, h) \in \binom{\mathbf{w}'}{t} \times \mathbf{h}$ do
  $\kappa' \leftarrow \mathsf{D}_{\mathsf{H}(\tilde{\mathbf{w}})}(h)$
  if $\kappa' \neq \perp$ then
    $\mathbf{w} \leftarrow \mathsf{D}_{\kappa'}(c)$
    return $\mathbf{w}$
return $\perp$

Figure 2: Algorithms for sketch (SS) and recover (Rec) for secure sketch for set-distances without using polynomial reconstruction (Reed-Solomon decoding). It uses a symmetric key authenticated encryption scheme $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ and a cryptographic hash function $\mathsf{H} : \{0,1\}^* \mapsto \{0,1\}^{\ell}$.

- **Security.** For any message distribution over W with min-entropy μ, the secure sketch must ensure that average conditional min-entropy of W given $\mathsf{SS}(W)$ is at least μ'.

Secure sketch for set difference. *A set difference* is a standard distance function over sets defined as follows: given two sets $\mathbf{w}$ and $\tilde{\mathbf{w}}$, set difference between them $\mathbf{w}\Delta\tilde{\mathbf{w}} = |\mathbf{w} \cup \tilde{\mathbf{w}} - \mathbf{w} \cap \tilde{\mathbf{w}}|$. Dodis et al. [29] provide a secure sketch construction for sets of items, where the distance function between them is the set difference. The construction uses algebraic polynomial interpolation. We will use this secure sketch internally to build multisketches. Though Dodis et al. proved the construction to be secure in the single message setting, the analysis did not simultaneously consider multiple messages. While we believe that their construction is secure even in such a setting, we used a simpler construction whose security can be argued easily in the random oracle model. We describe Dodis et al. approach in Appendix A.

In Figure 2 we show the simple sketch construction. It uses a secure and robust symmetric key encryption scheme $\mathsf{SKE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ and a cryptographic hash function H. The hash function $\mathsf{H} : \{0,1\}^* \mapsto \{0,1\}^{\ell}$, where ℓ is the security parameter. Here K is a key generation function that selects keys uniformly randomly from $\{0,1\}^{\ell}$; E is an encryption function; and D is the decryption function. Loosely speaking, we want the encryption scheme to be robust, that is decryption with wrong key fails — outputs $\perp$ — and to be semantically secure — without the knowledge of the key, no one can distinguish between encryption of a message from a random bit-string. Our preferred choices for deployment are AES-HMAC in CBC mode with 128-bit key (and PKCS7 padding) for encryption, and SHA-256 as the hash function.

4 MAIN CONSTRUCTION

We begin this section with the definition of a *multisketch*, followed by a concrete construction. As detailed in Section 3, a conventional secure sketch operates on a single message — generally corresponding to a single user's secret, such as a biometric template. A multisketch, in contrast, operates on the messages of *multiple users*. Below we define multisketch more formally.

Definition 4.1 (Multisketch). A multisketch is a pair of algorithms $(\mathsf{MS}, \mathsf{MRec})$, that are defined as follows.

- MS is a randomized procedure that takes as input a user identifier u, a message tuple $\mathbf{w}$, and a state s (can be empty $\varnothing$). MS updates the state and returns the updated state s′.

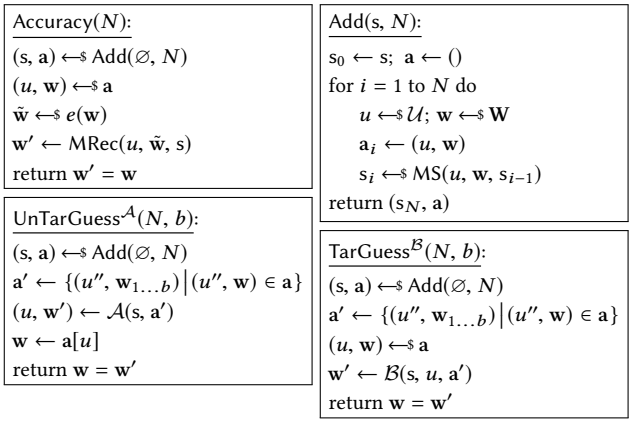

Figure 3: The accuracy (Accuracy) and security (TarGuess and UnTarGuess) measures of multisketch. All of the algorithms are implicitly parameterized by message tuple space $\mathbf{W}$ and space of user ids $\mathcal{U}$.

- MRec, a deterministic procedure, similarly takes as input a user id u, a message $\tilde{\mathbf{w}}$, and a state s and outputs a message $\mathbf{w}'$, should $\tilde{\mathbf{w}}$ be similar to $\mathbf{w}'$, that is $\mathbf{L}_t(\mathbf{w}', \tilde{\mathbf{w}}) = \texttt{accept}$.

It is characterized by two key metrics:

- **Accuracy:** The accuracy of a multisketch is defined as the ability to recover the originally registered message $\mathbf{w}$ of a user given $\tilde{\mathbf{w}} \leftarrow\!\!\$\ e(\mathbf{w})$. The accuracy is measured using the probabilistic algorithm Accuracy shown in Figure 3. Observe the accuracy depends on the number of users registered in the state s. Therefore, we call a multisketch scheme (N, δ)-accurate if $\Pr[\,\mathsf{Accuracy}(N) = \mathsf{true}\,] \geq 1 - \delta$, where the probability is computed over the random coins used by the algorithm Add and the error function e. The algorithm Add takes as input an initial state s, and a number N, generates N random user identifiers u and their messages $\mathbf{w}$, registers them sequentially in the state s_i using MS, and returns the updated state s_N along with the list $\mathbf{a}$ of generated users and their messages.
- **Security:** The security of a multisketch is defined using the guessing games UnTarGuess and TarGuess shown in Figure 3. More detailed security discussion is given in Section 5.

4.1 Construction of TenSketch

We give a construction of multisketch for tenprints — i.e., tuples of prints of all ten fingers of a user — that we call TenSketch. The construction can easily be extended to include other biometrics, such as iris scans or hand geometry.

The pseudocode of the procedures for sketching (MS) and recovery (MRec) are given in Figure 4. TenSketch calls as subroutines the set-distance secure sketch algorithms given in Figure 2. Multisketch uses a matching algorithm $\mathcal{L}$. We assume for simplicity a single matching algorithm $\mathcal{L}$ that works for all message spaces W_i, although our algorithms can easily be adapted to distinct matching algorithms $\mathcal{L}_i$ for different message spaces.

The state s in TenSketch consists of two databases $\mathcal{I}$ and $\mathcal{F}$, where $\mathcal{I}$ is a key value store in which keys are user identifiers $u \in \mathcal{U}$ and values are the sketches output by SS^{Δ}_t. The other database $\mathcal{F}$ stores the fingerprint templates of users. It consists of $n = 10$ columns, one for each finger. Given a message tuple $\mathbf{w}$ and a user

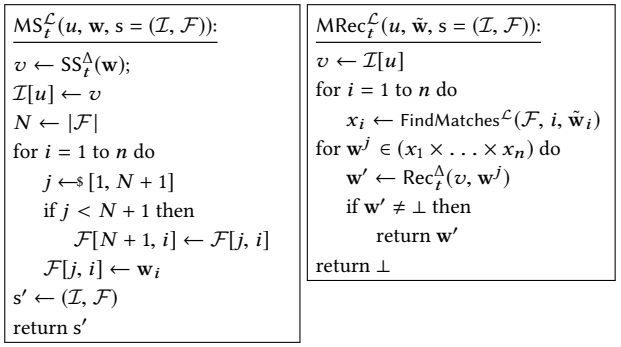

Figure 4: Multisketch (MS) and recover (MRec) algorithms. The function $\mathsf{FindMatches}^{\mathcal{L}}(\mathcal{F}, \mathbf{w}_i)$ finds at most l matches for message $\mathbf{w}_i$ from the database $\mathcal{F}$ using $\mathcal{L}$.

id u, MS first computes the set-distance sketch v of $\mathbf{w}$ and stores it at $\mathcal{I}[u]$. Then, MS inserts each message $\mathbf{w}_i$ in the i^{th} column in $\mathcal{F}$ at random locations.

During recovery, on input $\tilde{\mathbf{w}}$ and a user id u, MRec searches the database $\mathcal{F}$ for potential matches using the algorithm $\mathsf{FindMatches}^{\mathcal{L}}$, which calls $\mathcal{L}$ as a subroutine. $\mathsf{FindMatches}^{\mathcal{L}}$ returns at most l — a parameter — best matches for each message $\tilde{\mathbf{w}}_i$. Then, for every tuple of matches found, MRec tries to recover the original message $\mathbf{w}$ using Rec^{Δ}_t with the sketch value v stored at $\mathcal{I}[u]$. There can be l^t possible tuple combinations that have to be checked via Rec^{Δ}_t. Thus, while larger l increases accuracy, practical implementation requires relatively small values (e.g., 2).

4.2 Prototyping TenSketch

We build a prototype of TenSketch to evaluate practical feasibility. We use a tenprint dataset shared by NIST [2]. Below we give details of the dataset and our implementation.

Data source. NIST released a tenprint dataset titled SD09 [2] in 2009. It contains full tenprints for 2,700 individuals, two images per finger, for a total of 54,000 fingerprint impressions. This is one of *the largest* datasets of tenprints released by NIST and widely used by the biometric research community (e.g., [60]). It also contains the gender of each subject and some meta-level information about each fingerprint, such as finger position, NCIC class [43], etc.

Among the dataset subjects, 88.8% are male, and the remaining are female (except two subjects with unspecified genders). Each fingerprint is assigned one of the six classes defined by NCIC [43]: arch (A), left-loop (L), right-loop (R), tented-arch (T), whorl (W), and scar or mutilation (S). The majority (more than 93%) of fingerprints are either of type W, L, or R.

Matching algorithm. TenSketch can operate with any matching algorithm, and, in some cases, the algorithm can be changed without needing to make any changes in the database. For our experiment, we use Bozorth3 [41] as our matching algorithm $\mathcal{L}$ for matching fingerprints. Bozorth3 was published by NIST with its NIST Biometric Image Software (NBIS) package. It's a widely embraced fingerprint matching algorithm in research (see, e.g., [22, 44]).

Bozorth3 works on *minutiae*, distinguished points in a fingerprint, representing ridge, bifurcations, or endings. Each minutia is represented as a tuple consisting of x, y, an angle θ, and a quality

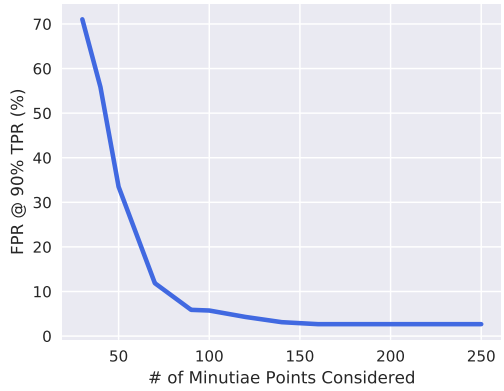

Figure 5: Performance of Bozorth3 for different numbers of minutiae points.

value q. A fingerprint template is represented as a list of minutiae — a format known as xyt format. We used the NIST mindtct algorithm (part of the NBIS package [41]) to extract minutiae points from each fingerprint image. Bozorth3 takes a pair of fingerprints in xyt format, and outputs a matching score.

Fingerprints differ in their numbers of readable minutiae. While more minutiae provide better matching accuracy, they might lead to better correlation attacks (see Section 5). We therefore truncate each fingerprint to have a maximum of r minutia points. For fingerprints with more than r minutiae, we considered the r best-quality minutiae (based on the quality value given by mindtct). Truncating minutiae points can affect accuracy, which we investigate next.

In the SD09 dataset, 47% fingerprints have 200 or more minutiae points. We tested the accuracy of Bozorth3 algorithm with different values of r, and found the performance changes negligibly for considering $r \geq 160$, as shown in Figure 5. We chose $r = 200$ as a limit with sufficient slack in case more advanced matching algorithms are proposed. Higher numbers of minutiae can also benefit the attacker, and therefore our security analysis in Section 7 provides a conservative estimate of an attacker's advantage.

Implementation of TenSketch. We implemented TenSketch as a Python 3.6 application. The database is instantiated with sqlite3 database. The fingerprint matching, as mentioned above, is done using Bozorth3 [41]. For the cryptographic functions, we used the Cryptography.io [8] library. We set $t = 7$. During recovery, for each fingerprint, we find at most $l = 2$ best matches with Bozorth3 matching threshold as 10.

We use the data of randomly sampled 2,500 SD09 subjects to construct the database D of TenSketch. Of the two impressions per finger in SD09, we pick the first set to store in the TenSketch database as templates; we use the second set of fingerprints for probing. The MS procedure does not check whether a user is already registered. In practice, it is important to have a supplementary procedure to ensure against duplicate users or fingerprints, which can undercut the correctness guarantees provided by TenSketch.

Computational performance. The computational performance of our implementation is shown in Figure 6. The median time taken among 100 runs for each of the core functions are reported. The experiment is done on a single thread of Xeon 5 processor with

Procedure	Time (s)	
	Avg	Var
SS_t^Δ	0.13	0.01
Rec_t^Δ	0.14	0.01
MS	0.13	0.02
MRec	176.67	59.34
MRec (parallelized)	59.12	9.23

Figure 6: Performance of prototype implementation of TenSketch on an Intel Xeon 5 machine. Other parameters are $N = 2500$, $t = 7$, and $l = 2$. The numbers are the average across 100 trials. The cause of the high standard deviation is from Bozorth3, as it takes different amounts of time for different probe fingerprints. The last row shows the performance when we parallelize the recovery procedure across 10 virtual cores of Xeon 5 machine using Python multiprocessing library.

46 GB RAM. Among all four procedures, the recovery procedure MRec consumes the maximum amount of physical memory, but that is still less than 256 MB. MRec is also relatively slow — takes about three minutes on a single thread of a Xeon 5 processor — compared to traditional login procedures during authentication. The MRec procedure tries to find the best matches of each fingerprint scan against all the fingerprint templates stored in the database using Bozorth3 one-to-many matching algorithm. This is the most expensive part of the computation. The performance can be *greatly improved via parallelization* and faster implementation of Bozorth3 using graphics processing units (GPUs) (c.f., [19]). We tried a naive parallelization using Python Multiprocessing and running the matching of ten fingers in ten (virtual) cores. The average time for MRec becomes in this case about a minute, but *essentially arbitrary parallelization is possible.* The variance of the time taken by Bozorth3 one-to-many algorithm matching is reflected in the time taken by MRec.

The storage overhead of TenSketch is nearly 2x compared to traditional biometric databases: The database has to store the original fingerprints, as well as the set-distance sketch output by Rec_t^Δ. If the goal of the TenSketch is only to recover a high entropy key, say 128-bit, then instead of storing the encryption of the whole tenprint, one can store the encryption of the 128-bit key. Each fingerprint tuple is represented as 600 integers (x, y, and θ; we ignore quality values). In our implementation of TenSketch, 64 KB of space is required per user (whereas a plaintext tenprints takes 24 KB).

Accuracy. We compute the accuracy of TenSketch according to the Accuracy function defined in Figure 3. As discussed before, we instantiate a TenSketch database with 2,500 randomly sampled users from SD09 dataset. We used the first impression as a template which is stored in the database, and the second impression in SD09 as a probe. To compute the false non-matching rate (FNMR), we sample 1,000 users randomly who are registered with our TenSketch, and then try to recover the users using the probe impression given in SD09 for those users.

We found the FNMR rate was quite high: 11%, 9%, and 8% for $l = 1,\ 2$, and 3, respectively. The higher value of l will require more combination of messages to check during MRec and adds significant computational overhead. We, therefore, decided on $l = 2$.

The high FNMR rate of TenSketch is due to low accuracy of Bozorth3 algorithm for many-to-one matching — finding a matching

fingerprint in a list. Though Bozorth3 has relatively low equal-error-rate for one-to-one matching (4%), in general, 67% of the correct template pairs are not assigned the highest matching score. FNMR rate (with $l = 2$) is worse for the little and the ring fingers (20%-25%), while around 7% for all the remaining fingers. Use of error correction in TenSketch helps improve accuracy. Therefore, we observed the FNMR rate of TenSketch can be reduced to 4% if we choose to have more error tolerance ($t = 6$).

For computing the false matching rate (FMR), we performed two experiments. First with fingerprints chosen from the one stored in the database, and the second one using unregistered tenprints. To simulate the first setting, we randomly pick $1,000$ tenprints from the database, and use that to recover any meaningful information from the database (i.e., a valid tenprint from the database). In the second setting, we use the hold-out 200 users from SD09 and tried to apply MRec with those template tuples. In either of the settings, MRec failed to recover any valid tuple, meaning FMR of MRec is zero for a database size of $2,500$ users.

We can use a matching threshold to ignore low-value matches, and thereby further reduce the FMR. However, in our experiments with Bozorth3 and SD09 dataset, we found the threshold on the matching score does no affect the false matching rate. It is very unlikely that t fingers of a user will be matched to t fingers of an imposter. If a different matching algorithm is used, instead of Bozorth3, that has higher FMR for TenSketch, FMR can be adjusted by varying the matching threshold.

Improving performance and accuracy. As shown above, the computational and matching performance of TenSketch heavily depends on the underlying matching algorithm, which is in this case Bozorth3. Here we give analytical estimates of TenSketch performance for alternative matching algorithms.

Accelerating fingerprint matching using GPUs is an active topic of research (c.f. [19, 35]). Cappelli et al. [19] demonstrate fingerprint matching rates of 9 million matches per second with a single NVIDIA Tesla C2075 GPU. For a database with a million users, using the Cappelli et al.'s algorithm on, e.g., an 4-GPU server, MRec would take less than half a second to complete. For systems like Aadhaar [3], with more than a billion users, the databases $(\mathcal{I}, \mathcal{F})$ can be partitioned into clusters of a million users each. (TenSketch with a million users is sufficient to provide 110 bits of security. See Section 5 for details.)

The FNMR rate can also be reduced by having a better fingerprint matching algorithm. Cappelli et al.'s algorithm [19] has an FNMR of 1.25% with FMR of 0%. It is unclear, however, how the matching accuracy varies across different fingers. (We could not obtain their code for testing.) We therefore report on experiments with another proprietary matching algorithm, Verifinger, from Neurotechnology [10]. Verifinger works well over mated fingerprints, but not with inked fingerprint. SD09 contains only inked fingerprints. Therefore, to measure the efficacy of our multisketch scheme with Verifinger, we used a mated fingerprint dataset [25], which we refer to as the WVU dataset. (More details of WVU dataset are given in Appendix C.) TenSketch with Verifinger as matching algorithm achieves perfect accuracy for $l = 2$ and $t = 7$ on the WVU dataset.

We used Bozorth3, because it is open-source. However, changing the matching algorithm in a multisketch system is easy — a key benefit of multisketches over prior template protection techniques. Moreover, in multisketches swapping in a new matching algorithm requires no database changes, provided that the stored biometrics are compatible with the new matching algorithm.

5 SECURITY OF TENSKETCH

The main concern with large biometric databases is the scale of impact of compromise, which could in some cases impact millions or billions of users. We design multisketches that decouples biometric information from individual identities to prevent an attacker form learning any meaningful information after stealing the biometric database. In this section, we formalize the threat model we consider for TenSketch and analyze its security.

Threat model. We consider *smash-and-grab* attacks, where the adversary *ephemerally* compromises the server and grabs the sketch databases $(\mathcal{I}, \mathcal{F})$. Such smash-and-grab attacks are the most prevalent form of credential compromise, e.g., passwords. The adversary aims to learn the tenprint of some user. We assume the adversary knows the matching algorithm $\mathcal{L}$ and the multisketch $\mathsf{MS}_t^{\mathcal{L}}$ and $\mathsf{MRec}_t^{\mathcal{L}}$ procedures. Additionally, the attacker might also know at most b fingerprints of some users, where b is a parameter of the security definition.

We don't consider stronger adversaries, e.g., a persistent passive adversary that can observe user registrations. Such an adversary can indeed learn users' biometrics trivially: It observes database changes as users register or authenticate — either action revealing their tenprints. Similar to persistent attackers, an attacker with access to multiple snapshots of the database at different times can identify newly added templates, resulting in a narrowed search space. We do observe, though, that even against these stronger adversaries, a multisketch system provides meaningful protection: Users whose templates are not observed by an adversary remain protected as long as there are sufficiently many uncompromised users.

We note that fingerprint matching times can vary for different fingerprints. Thus an attacker who can observe the authentication timings of a user might be able to prune the set of candidate fingerprints for that user. Such timing-based side channels, however, can be mitigated by normalizing the matching times for all users.

We also assume that every user is present at most once in the database and the fingerprint template database $\mathcal{F}$ does not contain any duplicates. The first assumption can be enforced at registration time by ensuring an individual user does not register more than once. Such is enforced in large scale biometric database systems [3]. The second assumption is normally true for biometrics because the representation of biometric templates are large and it is extremely unlikely that templates of two different individuals are the same. (The entropy of the biometric recognition system, however, is not much (about 16-bits [7]) due to error in readers and unavailability of better matching algorithms.)

Attack models. The threat model as described is very similar to the one used for password databases. Following the precedence from passwords [17], we model two different types of attacks against TenSketch: *targeted* and *untargeted*. In the targeted setting, the attacker seeks to learn the tenprint $\mathbf{w}$ of a *particular* user u. In

the untargeted setting, the attacker tries to find the tenprint of *any* user in the database. We specify security games TarGuess and UnTarGuess respectively for these adversaries in Figure 3; adversarial success is the probability that the relevant security game outputs true.

Note that for any attacker $\mathcal{B}$ for TarGuess, we can find an attacker $\mathcal{A}$ for UnTarGuess that queries $\mathcal{B}$ with each user in $\mathcal{I}$, and outputs any result from $\mathcal{B}$ other than $\perp$. Thus:

$$\Pr\left[\mathsf{UnTarGuess}^{\mathcal{A}} = \mathsf{true}\right] \geq \Pr\left[\mathsf{TarGuess}^{\mathcal{B}} = \mathsf{true}\right].$$

We can come to another inequality by constructing an adversary $\mathcal{B}$ which, when challenged on a user u, queries $\mathcal{A}$. If $\mathcal{A}$ outputs a tenprint for a user u' and $u = u'$, $\mathcal{B}$ outputs that. Therefore, $\mathcal{B}$ wins with probability equal to $\mathcal{A}$ when $u = u'$, else it fails. Therefore,

$$\Pr\left[\mathsf{TarGuess}^{\mathcal{B}} = \mathsf{true}\right] \geq \frac{1}{N} \cdot \Pr\left[\mathsf{UnTarGuess}^{\mathcal{A}} = \mathsf{true}\right].$$

Consequently, it suffices for us to explore TarGuess only, which will also provide an upper bound on the success probability of UnTarGuess game.

The ideal setting for TenSketch would be if the different templates of different fingers of a user were completely independent. However, this is not true in practice. Therefore, we first analytically compute the attacker's advantage assuming complete independence between different fingers. Next, we will give an empirical result about the advantage of an attacker lifting the assumption.

Mutually independent messages. The messages are said to be independent if the attacker is not able to find other fingerprints of a user from the database given one or more of their fingerprints with probability better than random guessing.

In this setting, to find the $\mathbf{w}$ associated with a user u, the attacker's best guess is to brute-force try all possible subset of size t of $\mathcal{F}$. Note the attacker only need to be able to guess t fingerprints of the user correctly, because remaining fingerprints can be recovered via Rec_t^{Δ} algorithm.

In the TarGuess setting, the attacker might know some $b < t$ of the target user's fingerprints. Let $N = |\mathcal{I}|$. Then the number of possibilities for the remaining part of the message tuple that the attacker has to compute is $\binom{n-b}{t-b} \cdot N^{t-b}$. The total number of valid solutions is $\binom{n-b}{t-b}$ — any combination $t-b$ messages from $n-b$ fingerprints. Therefore, if all the message tuples are equally likely, the attacker has to make on expectation $N^{(t-b)}$ queries to Rec_t^{Δ}. This is equivalent of $(t-b) \cdot \log N$ bits of security.

To get a sense of how large this value is for a realistic size biometrics database, let assume $N = 10^4$ (ten-prints for ten thousand users) and $t = 7$ (at least 7 fingers has to match for Rec_t^{Δ} to successfully recover the tenprint of the user). Assuming the attacker already know $b = 1$ fingerprint of the target user u, then the security in bits is $(7-1) \cdot \log_2(10^4) \approx 79.7$-bits. With $N = 10^6$ users, the security can be 119.6 bits.

Security without independence assumption. Previous security analysis assumes independence of the fingerprints, however, in many situations, particularly for tenprints, this assumption might not hold true. If that is the case, the attacker might be able to avoid checking every message tuples with Rec_t^{Δ} and improve it's performance significantly.

While the attacker could take advantage of the correlation among fingerprints in many ways, the most obvious and easy to use strategy would be to build a classifier that classifies a tenprint as belonging to a single user, or not. In Section 6 we will show how to build such classifier that exploits the similarities between fingerprints belonging to an individual. We call such classifiers *adversarial classifiers*. In Section 7, we show how to use adversarial classifiers to attack TenSketch.

6 ADVERSARIAL CLASSIFIERS

In this section, we show how to build *adversarial classifiers*, that can identify a tuple of fingerprints belonging to a single user or not. We use SD09 dataset for training our adversarial classifiers. We randomly partition the dataset into two groups of size 2500 and 200 users as training and test sets, respectively. In SD09 dataset, each user contains two sets of tenprints — 20 fingerprints in total.

Due to the low volume of available training data and the curse of dimensionality, building a classifier that can classify a tuple of size $n = 7$ could be difficult. Therefore, we build classifiers for various tuple sizes, starting with a *pair classifier* that classifies only template pairs. We then go on to explore ways to build *higher-order classifiers*.

All of the experimental results are performed on a server with 128 GB of memory, two Intel Xeon E7 CPUs (12 core each), and four Nvidia Titan Xp GPUs (each with 12 GB of memory). We describe the DNN models in Python using Keras library [24] with Tensorflow [13] as the backend.

Notation. Let $\mathcal{C}_i$, for $2 \leq i \leq t$, be the classifier that takes as input a tuple of fingerprint templates of size i and outputs a score in $[0, 1]$. Therefore, $\mathcal{C}_i(\mathbf{w}_1, \ldots, \mathbf{w}_i)$ denotes the likelihood (according to the classifier) that the message tuple $\mathbf{w}_1 \ldots \mathbf{w}_i$ belongs to a single user. Each fingerprint template ($\mathbf{w}_j$) comprises of 200 minutiae points and each minutiae point has x, y, an angle θ, and a quality value q. [2] As such, each template has 600 integer values (800 when using quality values). Utilizing quality values during the training process has a significant effect on the accuracy as well as the true and the false positive rates. If the quality value of the minutiae points are used, we denote the corresponding classifier as $\mathcal{C}_i^q$.

6.1 Pair Classifier

A pair classifier outputs the probability that two given fingerprint templates belong to the same person. We explored various DNN architectures to construct a pair classifier. In particular, we tried a single DNN architecture and a split Siamese architecture.

Dataset. The datasets for training and testing are created by taking all possible pairs of tenprints for each person — 45 pairs for each tenprint, 90 pairs for each person. These intra-person pairs are labeled as "1" (belonging to the same person). We generate inter-person fingerprint pairs by selecting randomly two fingers from two different users in SD09. To have an unbiased classification, we have generated the same number of inter-person fingerprints as that of intra-person. This is done separately for training (2500 users)

[2] For TenSketch we do not need to store quality values explicitly. Nevertheless, we used them for constructing the adversarial classifiers to get a more conservative estimate of attacker's advantage.

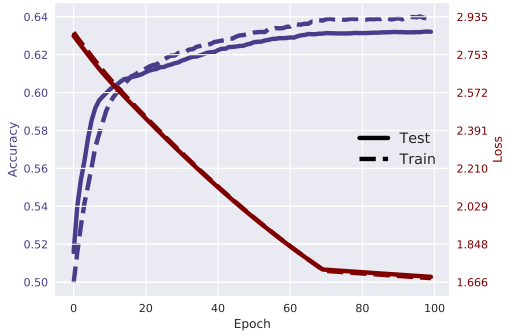

Figure 7: Accuracy and loss on train and test datasets for $\mathcal{C}_2$.

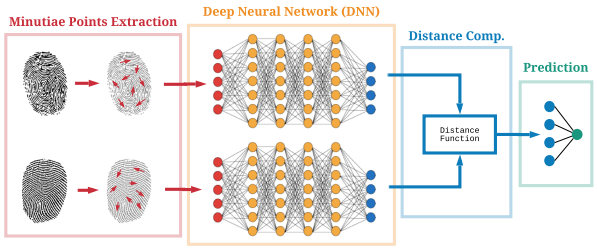

Figure 8: High-level diagram of $\mathcal{C}_2$ based on the Siamese architecture.

and testing (the remaining 200 users). Therefore, the training and testing sets contain in total 450,000 and 36,000 pairs of fingerprint templates. The feature set (x, y, θ, and q) is normalized to have zero mean and unit standard deviation, and all pairs within each training and testing set are permuted so that the training algorithm converges faster and to a more accurate model.

Single DNN architecture. We explored several DNN architecture. In order to train the DNN models, we have used mini-batch gradient descent (with a batch size of 1024). Two optimization methods have been used: Root Mean Square (`RMSProp`) and `Adam`. The best results are achieved using `Adam` optimization algorithm with the learning rate of $\gamma = 10^{-6}$.

The DNN models are trained until the test accuracy is saturated and/or the model starts to overfit to the training data. Figure 7 shows accuracy and the value of the loss function for the training and test datasets over 100 epochs. In order to avoid overfitting to the training data, L_2 regularization is used with the regularization parameter of $\lambda = 0.001$ for all hidden layers. The best performing architecture is a 5-layer DNN model consisting of

$$\mathsf{FC}^{320}_{\mathsf{ReLu}} \mapsto \mathsf{FC}^{640}_{\mathsf{ReLu}} \mapsto \mathsf{FC}^{1280}_{\mathsf{ReLu}} \mapsto \mathsf{FC}^{640}_{\mathsf{ReLu}} \mapsto \mathsf{FC}^{320}_{\mathsf{ReLu}} ,$$

where $\mathsf{FC}^{n}_{\mathsf{ReLu}}$ denotes a *Fully Connected* layer with `ReLu` activation function and n output neurons. The input of a fully-connected layer is multiplied by the weight matrix and the resulting vector is passed to a non-linear function. One of the most popular non-linear functions is Rectifier Linear Unit, $\mathsf{ReLu}(x) = \max(x, 0)$. The output of the last layer of DNN is then passed to the Sigmoid function, $\sigma(x) = (1 + e^{-x})^{-1}$, which represents the probability that the input belongs to the class "1". The values of the weights in the `FC` layer is learned using Back Propagation Algorithm.

The best performing $\mathcal{C}_2$ achieves 65% accuracy. If the quality values are available in the template the accuracy goes up to 70%. In both the settings, the false negative rates (FNR) are high, more than 25%. The false positive rates (FPR) are also high: 38% and 32% respectively for with and without quality values. Nevertheless, this classifiers show that different fingerprints of users are indeed correlated.

Siamese Architectures. In addition to standard deep neural network architectures, we explore a special class of DNNs, called *Siamese Networks* [56, 58]. In Siamese networks, two input feature sets are passed to two sister networks with identical parameters (weights). We give a high-level diagram of the setup in Figure 8. More details on the training Siamese network is given in Appendix E.

Despite Siamese being useful for facial recognition, we found Siamese networks perform poorly compared to single DNNs. The accuracy of the best performing Siamese network with the quality values is only 66%, and without quality values it is 62%.

The DNN model outperforms the Siamese network consistently. We therefore choose the DNN model as $\mathcal{C}_2$ and use for the attacks described in Section 7. Next, we elaborate on how one can generalize the pair classifier to obtain higher-order classifiers.

6.2 Higher-Order Classifiers

The ultimate goal of an attacker is to identifying a certain number (e.g., 7) of fingerprint templates that belong to the same person. There are two main strategies that an attacker can pursue: (1) she can leverage the pair classifier as an oracle and perform multiple calls to it, or (2) she can train more complex DNNs that accept multiple templates as input. In what follows, we discuss these two strategies in more detail and compare their performances.

$\mathcal{C}_2$-based higher-order classifiers. Using a pair classifier an attacker can build a higher-order classifier $\mathcal{C}_i$ by checking all possible $\binom{i}{2}$ pairs of templates in the template tuple via $\mathcal{C}_2$. The attacker accepts the i templates if the all pairs are accepted by $\mathcal{C}_2$. We call such classifiers *extrapolating classifiers*, denote by $\hat{\mathcal{C}}_i$ (and $\hat{\mathcal{C}}^q_i$ when quality values are available). In Figure 9 (left two figures), we show the ROC curve of this approach for different values of $2 \le i \le 7$ with and without the quality values. When the quality values are available, extrapolating classifiers are effective. $\mathcal{C}^q_7$ obtains $< 30\%$ FPR at 1% FNR. Without the quality values the extrapolating classifier performs poorly and not very useful for classification. This is important as we remove quality values from the templates for TenSketch.

Standalone higher-order classifiers. The second strategy is to build higher-order classifiers by training more complex DNNs that take multiple templates as input. This strategy turns out to be more effective and leads to higher accuracy than extrapolating classifiers. The process of creating training and test sets are an extension of the process we elaborated for $\mathcal{C}_2$ in Section 6.1. We leverage the same architecture as for $\mathcal{C}_2$ to generalize $\mathcal{C}_i$ by scaling the width of each layer linearly with respect to i.

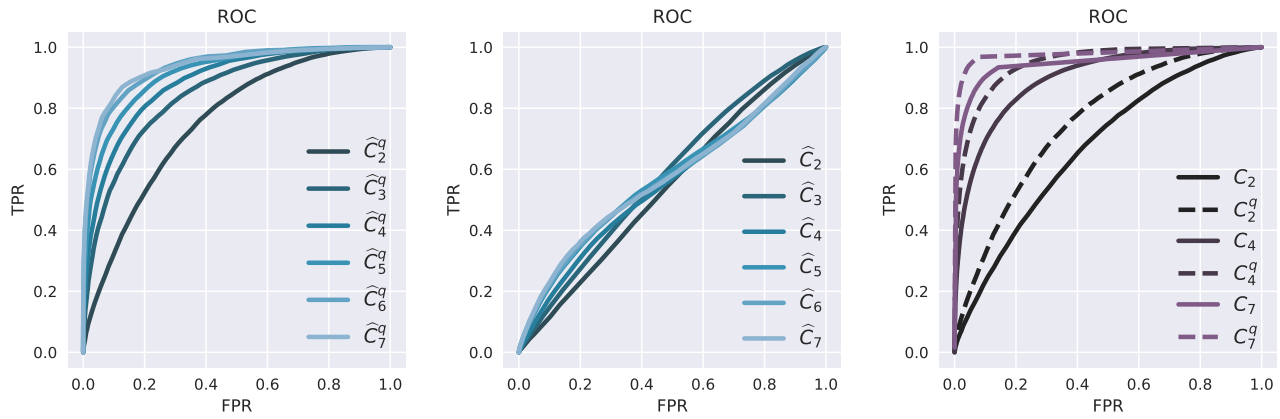

Figure 9: Illustrating the effect of the number of fingerprint templates and the quality values on the performance of higher-order classifiers. The **left two** figures show the ROC curve for with and without the quality values for extrapolating classifier $\hat{\mathcal{C}}_i$. The **rightmost** figure shows the performance of learned higher-order classifier $\mathcal{C}_i$.

Classifier	# Param.	Acc (%)	FPR (%)	FNR (%)	Classifier	# Param.	Acc (%)	FPR (%)	FNR (%)	# Tr. Smp.	# Te. Smp.
$\mathcal{C}_2$	2.4M	62	43	30	$\mathcal{C}_2^q$	2.6M	69	36	25	450.0K	36.0K
$\mathcal{C}_3$	5.5M	74	25	24	$\mathcal{C}_3^q$	5.8M	81	18	19	1.2M	96.0K
$\mathcal{C}_4$	9.7M	81	14	22	$\mathcal{C}_4^q$	10.2M	86	10	16	2.1M	168.0K
$\mathcal{C}_5$	15.2M	84	7	24	$\mathcal{C}_5^q$	16.0M	90	4	15	2.5M	201.6K
$\mathcal{C}_6$	21.9M	84	4	25	$\mathcal{C}_6^q$	23.1M	89	1	18	2.1M	168.0K
$\mathcal{C}_7$	29.8M	86	3	23	$\mathcal{C}_7^q$	31.4M	92	2	12	1.2M	96.0K

Figure 10: Complexity and performance of $\mathcal{C}_i^q$ and $\mathcal{C}_i$ classifiers. "# Tr. Smp." and "# Te. Smp." represent the number of training and test samples, respectively (same number for both $\mathcal{C}_i^q$ and $\mathcal{C}_i$). "# Pram." represents the number of trainable parameters in DNN.

The performance of different $\mathcal{C}_i$ is given in Figure 10. Notably, the accuracy increases with the size of the tuples. But, note the false negative rate remains more or less unchanged.

Effect of quality values. Having access to the quality values consistently increase the accuracy. We, therefore, remove quality values before storing templates in $\mathcal{F}$. Fortunately, Bozorth3 uses the quality values only to order the minutiae points. Therefore, in our prototype implementation, we decide to remove quality values and store ordered minutiae points only. As some other matching algorithm might take advantage of the quality values, we conservatively consider the classifiers with quality values for evaluating attacks using adversarial classifiers.

7 SECURITY EVALUATION USING ADVERSARIAL CLASSIFIERS

Using the adversarial classifiers described in Section 6, an attacker can expedite the recovery of a valid fingerprint tuple. In this section, we will explore a few different attack strategies and their performance by simulating them on SD09 dataset.

One limitation of our attack simulation is that it uses the same data used for training the classifiers to instantiate TenSketch database. This was unavoidable due to the lack of availability of a larger tenprint dataset. However, we emphasize that as shown in Section 6, the training and testing accuracy of the classifiers are very similar, we anticipate the simulation results will hold for larger datasets too. Moreover, the simulation with the training data benefits the attackers and provides a conservative bound of security.

Recall, we want to determine the complexity of the computation work an attacker has to do to win the TarGuess game (See Figure 3.) The attacker's goal here is to find a valid tenprint of a user u given $b = 1$ of the user's fingerprints. The attacker only needs to find a message tuple where at least t fingerprints are correct — belongs to the user u. Using such a tuple and Rec_t^Δ, the attacker can recover all the fingerprint templates of the user.

We will assign a cost of c_c and c_s as the cost of calling the classifier and the cost of calling the Rec_t^Δ procedure on a fingerprint tuple. Every other computation, conservatively, is assumed to be free for the attacker. In general, c_c will be much smaller than c_s (but not negligible). For ease of reporting the results, we will often assume $c_s \gg c_c$.

If there were no correlation between fingerprints the optimal strategy of an attacker would be to apply Rec_t^Δ on all possible message tuple of size t. Therefore, the cost of such naive attack will be $\text{C} = \binom{n-b}{t-b} \cdot N^{t-b} \cdot c_s / \binom{n-b}{t-b} = N^{t-b} \cdot c_s$. (The attacker only have to find one of $\binom{n-b}{t-b}$ many valid tuples.) For a TenSketch with $N = 2,500$ users and $t = 7$, this will result in $6 \times \log 2500 = 67.7$-bits of security (assuming applying Rec_t^Δ is of unit cost). We define the *advantage* of an classifier-based attack algorithm as $\log(\text{C}/\text{C}_{\text{ML}})$, where C_{ML} is the cost of the attack using adversarial classifiers. The advantage of an attack represents the loss in number of bits of security due to correlation among fingerprints of a user as captured by the adversarial classifiers. Below, we will report on both the cost and the advantage of each attack.

There are many different ways an attacker can leverage classifiers to identify t-tuples. In Appendix F we consider in detail the strategy

of combining different classifiers, and show that using $\mathcal{C}_7$ alone incurs the lowest computational overhead. Our first three strategies below use $\mathcal{C}_7$. The last strategy we consider identifies t-tuples using different $\mathcal{C}_i's$ incrementally.

1. Pruning improbable candidates. An obvious strategy to find a correct message tuple of size t would be to prune out improbable tuples using the classifier $\mathcal{C}_t$, and thereby, to reduce the number of tuples to check using Rec_t^{Δ}. However, this strategy requires the attacker to apply the classifier on all possible $\binom{n-b}{t-b} \cdot N^{t-b}$ message tuples. This number can be very large, posing a serious engineering challenge for the attacker. Moreover, as shown in Section 6, the classifier will have some non-zero false positive rate, say ν. Therefore, the total cost of this attack will be $\hat{C} = \binom{n-b}{t-b} \cdot N^{t-b}\,(c_c + c_s \cdot \nu) \geq \binom{n-b}{t-b} \cdot N^{t-b} \cdot c_s \cdot \nu = C \cdot \nu$. The advantage of this attack is at most $-\log \nu$. As per our experiment in Section 6.2, without the quality values $\nu = 0.1$ at true positive rate of 90%, which translates to 3.3-bits of security loss. With the quality values, $\nu = 0.05$ (See Figure 9), and the security loss would be 4.3 bits.

The true positive rate of the classifier dictates the attacker's chance of winning the game should she be able to finish going over all the message tuple combinations. However, the attacker can forego higher accuracy at the cost of faster retrieval. As we show in Figure 9, reducing the true positive rate does not result in a sufficiently low false positive rates, and will not be effective.

2. Ordering message tuples based on the classifier. An attacker can try to order the tuples using the classifier outputs (confidence values), and check using Rec_t^{Δ} in decreasing order of their confidence values. To generate high confidence — according to the classifiers — message tuples, the attacker can use hill-climbing approach as described below, or an approach similar to generating passwords from a Markov model [47]. However, for any of these approaches to succeed, the correct message tuple must be assigned high confidence by the classifier.

To verify that, we experimented to find the average rank of a correct tuple in a list of randomly sampled tuples. We pick a random user u, and pick t fingerprints of the user u randomly, containing b known fingerprints. We also pick 10^4 fingerprint tuples from $\mathcal{F}$ each containing the b known fingerprints and rest randomly sampled. We found the rank of the correct tuple $\mathbf{w}$ is at $a = 12\%$ of the length of the list, on an average. That means even if the attacker manages to order all the fingerprint tuples, she has to go through on an average 12% of the list before encountering a valid message tuple.

Assuming, if an attacker first prune the database of the improbable candidate tuples (very low classifier's outputs), and then sort the list based on the output of $\mathcal{C}_7$, will have to apply Rec_t^{Δ} on $C \cdot \nu \cdot a$ many fingerprint tuples. This translates to $-\log_2(0.05 \times 0.12) = 7.4$-bits of security loss due to correlation among fingerprints.

3. Finding high confidence message tuples. To avoid computing all possible message combinations as before, the attacker can try to find message tuples for which the classifier assigns high confidence value. This can be done efficiently by using an approach similar to hill-climbing used in optimization techniques. The attacker starts with a random fingerprint tuple of size t (that includes b known fingerprints) and replaces the unknown fingerprints one at a time in such a way that improves the output of the classifier $\mathcal{C}_t$. The replacement procedure stops at a tuple $\mathbf{w}$ if no other replacement is possible that yields higher score than $\mathbf{w}$. The attacker tests $\mathbf{w}$ with Rec_t^{Δ} for correctness. If it fails, it restarts with another random tuple and continues. The algorithm is given in Figure 11.

Global input: $\mathsf{s} = (\mathcal{I}, \mathcal{F})$, t

HillClimb(u, $\overline{\mathbf{w}}$):

$\mathsf{s} \leftarrow \mathcal{I}[u]$; $\tilde{\mathbf{w}} \leftarrow\!\!\$\ \mathcal{F}$
if $\overline{\mathbf{w}} \neq \varnothing$ then $\tilde{\mathbf{w}}_{1\ldots|\overline{\mathbf{w}}|} \leftarrow \overline{\mathbf{w}}$
$Y \leftarrow \{\tilde{\mathbf{w}}\}$ /* Done tuples */
$x_{\max} \leftarrow \infty$; $x \leftarrow 0$
while $x_{\max} \geq x$ do
  $x \leftarrow \mathcal{C}_t(\tilde{\mathbf{w}})$; $X \leftarrow \varnothing$
  for $i \in \{|\overline{\mathbf{w}}| + 1, \ldots, n\}$
    $X \leftarrow X \cup \{\mathbf{w} \leftarrow \tilde{\mathbf{w}}; \mathbf{w}_i \leftarrow w \,\big|\, w \in \mathcal{F}_i\}$
  $X \leftarrow X \setminus Y$; $Y \leftarrow Y \cup X$
  $\tilde{\mathbf{w}} \leftarrow \text{argmax}\{\mathcal{C}_t(\mathbf{w}) \,\big|\, \mathbf{w} \in X\}$
  $x_{\max} \leftarrow \mathcal{C}_t(\tilde{\mathbf{w}})$
return $\tilde{\mathbf{w}}$

Figure 11: Hill-climbing algorithm to find high confidence fingerprint tuple for attacking TenSketch. The algorithm also takes $\overline{\mathbf{w}}$, which is b known fingerprints of the user.

We simulated this attack, but even with 10^12 iterations (restarts), the algorithm failed to find a correct t-tuple. On a closer look at the classifier outputs, we found that the correct tuple never obtains the highest classifier's confidence among its neighboring tuples — tuples that differ by only one fingerprint; which is the major reason for the failure of this attack. Therefore, hill-climbing is not an effective way for the attacker to take advantage of the adversarial classifiers.

4. Generate message tuples incrementally. Finally, we consider generating a t-size message tuple incrementally beginning with b known fingerprints. The attacker uses the classifier $\mathcal{C}_i$ to find a i-tuple including the b known messages that have high confidence and move on to find a $(i+1)$-tuple using $\mathcal{C}_{i+1}$, $(i+2)$-tuple using $\mathcal{C}_{i+2}$, and so on. The algorithm is given in Figure 12.

For each classifier $\mathcal{C}_i$, we also specify a threshold τ_i and consider the tuples with higher confidence values than τ_i. These thresholds are computed in such a way that ensures $\mathcal{C}_i$ has a true positive rate of 90%. We fix the true positive rates instead of false positive rate, as failing to consider the correct tuple for further processing immediately fails the attacker.

Now we want to compute the number of times the attacker has to query Rec_t^{Δ} in this attack. Following the algorithm in Figure 12, we can see, when the size of a partially constructed message tuple $\mathbf{w}$, such that $|\mathbf{w}| = j - 1$ and $j \geq 2$, the size of T_j is $|T_j| = N \cdot \delta_j$, where $N = |\mathcal{F}_j|$ and δ_j is as defined below.

$$\delta_j = \Pr\left[\ \mathcal{C}_j(\mathbf{w}_1, \ldots, \mathbf{w}_j) > \tau_j \ \mid\ \mathcal{C}_{j-1}(\mathbf{w}_1, \ldots, \mathbf{w}_{j-1}) > \tau_{j-1}\ \right],$$

for $j > 2$; for $j = 2$, we define $\delta_2 = \Pr\left[\,\mathcal{C}_2(\mathbf{w}_1, \mathbf{w}_2) > \tau_2\,\right]$. Therefore, total number of t-tuples generated by the algorithm is

$$\prod_{j=b+1}^{t} |T_j| = N^{t-b} \cdot \prod_{j=b+1}^{t} \delta_j\ .$$

Global input: $\mathsf{s} = (\mathcal{I}, \mathcal{F}),\ t, (\tau_1, \ldots, \tau_t)$

GenTupIncr(u, $\overline{\mathbf{w}}$):

$\mathsf{s} \leftarrow \mathcal{I}[u];\ \ \tilde{\mathbf{w}} \leftarrow \perp$
if $\overline{\mathbf{w}} \neq \varnothing$ then $X \leftarrow \{\overline{\mathbf{w}}\}$; else $X \leftarrow \mathcal{F}_1$
while $X \neq \varnothing$ do
  $\mathbf{w} \leftarrow X$.pop()
  $j \leftarrow |\mathbf{w}| + 1$
  if $j >= t$ then
    $\tilde{\mathbf{w}} \leftarrow \mathsf{Rec}_t^{\Delta}(\mathsf{s}, \mathbf{w})$
    if $\tilde{\mathbf{w}} \neq \perp$ then **break**
  else
    $T_j \leftarrow \left\{v \in \mathcal{F}_j \,\middle|\, \mathcal{C}_j(\mathbf{w}_1, \ldots, \mathbf{w}_{j-1}, v) > \tau_j\right\}$
    $X \leftarrow X \cup \left\{\mathbf{w} \cup \{v\} \,\middle|\, v \in T_j\right\}$
return $\tilde{\mathbf{w}}$

Figure 12: Algorithm for generating message tuples of size t incrementally. $\mathcal{F}_i$ is the database containing fingerprint template of the i^{th} finger of each user, and t is the minimum number of fingers required to recover the whole tenprint set.

The cost of this attack will be

$$N^{t-b} \cdot \prod_{j=b+1}^{t} \delta_j \cdot (c_s + 2 \cdot c_c) \approx C \cdot \prod_{j=b+1}^{t} \delta_j \ .$$

We computed δ_j's for $2 \leq j \leq t$ empirically from the SD09 data and the classifiers $\mathcal{C}_j$ and $\mathcal{C}_j^q$. We found the $\prod_{j=b+1}^{t} \delta_j \geq 2^{-9}$ with quality values, and 2^{-8} without quality values. Therefore overall loss in security via this attack is at most 9 bits.

Discussion. We explored possible attack strategies using generic classifiers across all fingers. We show that an attacker can at most reduce the security by 9 bits by exploiting the fingerprint correlations. Final security of TenSketch with $t = 7$ and $N = 2,500$ registered users in bit-strength is 58-bits, assuming the attacker knows one of the fingerprints of the user.

We claim this is a conservative estimate, as we did not count for the computational cost associated with applying the classifier. Moreover, the loss in security can be further reduced by increasing the hashing cost of H used by SS_t^{Δ} and Rec_t^{Δ}, and thus increasing c_s.

The classifiers that we built is generic in a sense that they do not consider positions of fingers. The classification accuracy might improve if one tries to build finger-specific classifiers. Training such classifiers however would require access to a large amount of biometric training data.

8 RELATED WORK

Biometric authentication is increasingly being used during border crossing [6], to provide essential government services (such as Aadhaar [3]), and for criminal investigation [5]. In addition to traditional fingerprints and iris scans [26, 36], other physiological attributes, such as ear [14] and hand geometry [55], palmprints [30], heartbeat [54], and body temperature are potential to be used as biometrics. For a detailed discussion on different potential "what you are" biometrics, refer to [37]. Multisketches can be used to protect a combination of different biometric attributes.

Biometric attributes are, however, noisy and change (slightly) every time they are recorded. Therefore, unlike passwords, there is no effective way of storing biometric templates securely. A number of research works have looked into protecting biometric templates using different techniques [39]. One of the most popular attempts to protect biometrics is using secure sketches and fuzzy extractors, which was first introduced by Dodis et al. [29]. Secure sketches can be used to recover a fingerprint provided during enrollment using a helper string (which is generated during enrollment) and a small variant of the original fingerprint. Fuzzy extractors, similar to secure sketches, allow extracting reliable random bits from noisy biometric data. Though such cryptographic techniques were seminal, they require redesigning biometric matching algorithms, often at the cost of accuracy. Prior research has looked into designing such matching algorithms (or distance functions) for fingerprints [52], iris [36, 63], and face [57]. However, such custom matching algorithms invariably degrade the matching performance and make it unusable in practice [32]. Therefore, to date fingerprints are not protected beyond encrypting them at rest and decrypting during matching. In mobile devices, often dedicated hardware is used to protect the fingerprint. See [38, 51] for a detailed discussion on the gap between theory and practice in this research area.

We propose a new template protection technique, what we call multisketch. A Multisketch system works with any matching algorithm and therefore removes the decades long problem that prevented template protection algorithms from being used in practice. The main drawback of multisketch, however, is that it requires multiple biometric features — which can be from different modalities — to be recorded for each individual and a large (e.g., thousands) number of users to provide meaningful security. Both of these requirements are met in current large-scale biometric databases used in practice [3, 11].

Our multisketch construction is conceptually similar to the set-distance secure sketch construction of Juels and Sudan [39], which conceals valid set elements using fake "chaff" ones. In our construction, however, the real set elements of other users are the "chaff." Additionally, Jules et al. [39] assume exact matching between a conjectured and valid set of elements during decoding, while we introduce the idea of approximate matching via biometric matching algorithms. Approximate matching passphrases were also proposed by Bard [15] by allowing a user to log in with small typographical mistakes. Bard proposed using a fixed dictionary of words carefully constructed so that no two words in the dictionary are closer than edit distance two; the user has to pick a passphrase consisting of words from this dictionary. Multisketch works without such a constraint and a dictionary.

Last but not least, we emphasize that secure multi-party computation (sMPC) and secure function evaluation (SFE) protocols [16, 23, 62] are *complementary* to multisketches, not alternatives. Templates and proffered biometrics in a multisketch system could, for instance, be (k, n)-secret-shared among n distinct servers and matching is performed using MPC. In this case, an attacker would need to compromise at least k servers in order to obtain the underlying database and *start* to attack the multisketch system.

Unfortunately, though, such use of MPC would be prohibitively expensive. Existing literature thus instead treats the different and simpler task of using SFE for privacy-preserving queries, i.e., so that when a user makes a biometric matching query, no information about the query is revealed to the server, and no information about

the database is revealed to the client — unless a successful match occurs. SCiFI [53] accomplishes this goal for face recognition, but requires 0.3 second per matching operation — or more than three days for a database with one million users. The harder task of MPC-based fingerprint matching would clearly not scale to large databases. In contrast, multisketches provide strong protection against biometric database breaches with a practical computational cost.

9 CONCLUSION

Biometric template protection is a long-elusive goal for decades. As large-scale biometric databases are being built across the globe and are being adopted more widely as a mode of authentication, such databases would be a lucrative target for attackers. Breaches of such databases will compromise the biometric identities of millions, and even billions of users.

We introduce a new cryptographic primitive, called *multisketch*, to protect large scale biometric databases. Multisketch works with multiple biometrics, even with different modalities, and can operate with any underlying biometric matching algorithm — a key difference with prior template protection techniques. To protect the stored templates from a breach of the database, multisketch removes the association between the identity of the user and their biometrics, and stores them in random order. Only when a user provides a set of biometrics that is nearly as correct as the originally given ones, the templates of the user can be recovered. Multisketch can be used for verification, as well as for generating high entropy keys.

We detail the design a multisketch for tenprints, which we call TenSketch. We prototype TenSketch using tenprint dataset released by NIST. Using Bozorth3 matching algorithm, we show the feasibility of TenSketch for real-world deployment. The computational and matching performance can be improved by using different, more advanced matching algorithms, and by updating (if necessary) the template information stored in the database. This changes can be done with the least overhead. Such biometric agility is one of the key benefits of a multisketch system.

We explore the correlation between different fingerprints of a user using advanced machine learning techniques. We build classifiers to identify fingerprints from the same user. Using the classifier we propose possible attacks and simulate their efficacies. We finally show that even in a conservative setting, the attacker's benefit is marginal, and TenSketch provides equivalent of at least 58-bits of security from a database of $2,500$ individuals. Database with more users would be more secure for TenSketch. Synthetic fingerprints [20] can be used to improve security, though more investigation is required to understand how easy it is to distinguish synthetic fingerprints from the real ones.

Multisketch built on multi-modal biometrics are expected to be more secure. Though, more investigation on the correlation among fingerprints and other biometrics of different modalities will help better estimate security of multisketch in practice. Also, building techniques to de-correlate fingerprints will improve the security of multisketches.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their insightful comments. This work was supported in part by NSF grants CNS-1514163, CNS-1564102, CNS-1704615 as well as ARO grant W911NF16-1-0145. UCSD co-authors were supported by Office of Naval Research (ONR) N00014-17-1-2500 grant. Emanuela Marasco was support in part by NSF under award no. CNS-1822094.

Note

Some of the research represented in this paper was performed previously by Ari Juels in his non-academic role as a consultant. The paper also includes later, distinct work performed in his role as a Cornell Tech faculty member.

REFERENCES

[1] 1973. National Background Information. (1973). https://www.nbinformation.com/.
[2] 2010. NIST 8-Bit Gray Images of Mated Fingerprint Card Pairs (MFCP) Volumes. (2010). https://www.nist.gov/srd/nist-special-database-9.
[3] 2018. Aadhaar. (2018). https://en.wikipedia.org/wiki/Aadhaar.
[4] 2018. About Touch ID advanced security technology. https://support.apple.com/en-us/HT204587. (2018).
[5] 2018. Fingerprints and Other Biometrics. (2018). https://www.fbi.gov/services/cjis/fingerprints-and-other-biometrics.
[6] 2018. Government Biometrics Month 2018: The Ascent of Biometric Border Control. https://findbiometrics.com/government-biometrics-month-biometric-border-control-509130/. (2018).
[7] 2018. iOS Security. https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf. (2018).
[8] 2018. Pyca/Cryptography. https://cryptography.io/en/latest/. (2018).
[9] 2018. Understand fingerprint security. https://support.google.com/nexus/answer/6300638?hl=en. (2018).
[10] 2018. Verifinger. https://www.neurotechnology.com/verifinger.html. (2018).
[11] 2019. Integrated Automated Fingerprint Identification System (IAFIS). https://en.wikipedia.org/wiki/Integrated_Automated_Fingerprint_Identification_System. (2019).
[12] 2019. Multilayer Perceptron. https://en.wikipedia.org/wiki/Multilayer_perceptron. (2019).
[13] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Operating Systems Design and Implementation (OSDI)*.
[14] Ayman Abaza, Arun Ross, Christina Hebert, Mary Ann F Harrison, and Mark S Nixon. 2013. A survey on ear biometrics. *ACM computing surveys (CSUR)* 45, 2 (2013), 22.
[15] Gregory V Bard. 2007. Spelling-error tolerant, order-independent pass-phrases via the Damerau-Levenshtein string-edit distance metric. In *Proceedings of the fifth Australasian Symposium on ACSW Frontiers-Volume 68*. Australian Computer Society, Inc., 117–124.
[16] Donald Beaver, Silvio Micali, and Phillip Rogaway. 1990. The round complexity of secure protocols. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. ACM, 503–513.
[17] Joseph Bonneau. 2012. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *IEEE Symposium on Security and Privacy (SP)*. IEEE, 538–552.
[18] Tracey Caldwell. 2015. Market report: border biometrics. *Biometric Technology Today* 2015, 5 (2015), 5–11.
[19] Raffaele Cappelli, Matteo Ferrara, and Davide Maltoni. 2015. Large-scale fingerprint identification on GPU. *Information Sciences* 306 (2015), 1–20.
[20] Raffaele Cappelli, D Maio, and D Maltoni. 2004. SFinGe: an approach to synthetic fingerprint generation. In *International Workshop on Biometric Technologies (BT2004)*. 147–154.
[21] Lily Chen. 2008. Recommendation for key derivation using pseudorandom functions. *NIST special publication* 800 (2008), 108.
[22] Sharat Chikkerur, Alexander N Cartwright, and Venu Govindaraju. 2006. K-plet and coupled BFS: a graph based fingerprint representation and matching algorithm. In *International Conference on Biometrics*. Springer, 309–315.
[23] Seung Geol Choi, Kyung-Wook Hwang, Jonathan Katz, Tal Malkin, and Dan Rubenstein. 2012. Secure multi-party computation of boolean circuits with

applications to privacy in on-line marketplaces. In *CryptographersâĂŹ Track at the RSA Conference*. Springer, 416–432.
[24] François Chollet et al. 2015. Keras. https://keras.io. (2015).
[25] Bojan Cukic, Jeremy M. Dawson, and Simona Crihalmeanu. 2010. Non-contact Multi-sensor Fingerprint Collection. https://www.ncjrs.gov/pdffiles1/nij/grants/246711.pdf. (2010).
[26] John Daugman. 2009. How iris recognition works. In *The essential guide to image processing*. Elsevier, 715–739.
[27] G. I. Davida, Y. Frankel, and B. J. Matt. 1998. On enabling secure applications through off-line biometric identification. In *IEEE Symposium on Security and Privacy*. 148–157. https://doi.org/10.1109/SECPRI.1998.674831
[28] Y. Dodis. 2005. On extractors, error-correction and hiding all partial information. In *Theory and Practice in Information-Theoretic Security, 2005. IEEE Information Theory Workshop on*. 74–79. https://doi.org/10.1109/ITWTPI.2005.1543961
[29] Y. Dodis, L. Reyzin, and A. Smith. 2004. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *Eurocrypt 2004*, C. Cachin and J. Camenisch (Eds.). Springer-Verlag, 523–540. LNCS no. 3027.
[30] Nicolae Duta, Anil K Jain, and Kanti V Mardia. 2002. Matching of palmprints. *Pattern Recognition Letters* 23, 4 (2002), 477–485.
[31] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. 2009. Privacy-preserving face recognition. In *International symposium on privacy enhancing technologies symposium*. Springer, 235–253.
[32] Davrondzhon Gafurov, Bian Yang, Patrick Bours, and Christoph Busch. 2010. Independent performance evaluation of fingerprint verification at the minutiae and pseudonymous identifier levels. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*. IEEE, 3186–3193.
[33] Shuhong Gao. 2003. A new algorithm for decoding Reed-Solomon codes. In *Communications, Information and Network Security*. Springer, 55–68.
[34] Vindu Goel. 10 April 2017. That Fingerprint Sensor on Your Phone Is Not as Safe as You Think. *New York Times* (10 April 2017).
[35] Pablo David Gutierrez, Miguel Lastra, Francisco Herrera, and Jose Manuel Benitez. 2013. A high performance fingerprint matching system for large databases based on GPU. *IEEE Transactions on Information Forensics and Security* 9, 1 (2013), 62–71.
[36] Gene Itkis, Venkat Chandar, Benjamin W Fuller, Joseph P Campbell, and Robert K Cunningham. 2015. Iris Biometric Security Challenges and Possible Solutions: For your eyes only? Using the iris as a key. *IEEE Signal Processing Magazine* 32, 5 (2015), 42–53.
[37] Anil K Jain, Ruud Bolle, and Sharath Pankanti. 2006. *Biometrics: personal identification in networked society*. Vol. 479. Springer Science & Business Media.
[38] Anil K Jain, Karthik Nandakumar, and Abhishek Nagar. 2008. Biometric template security. *EURASIP Journal on advances in signal processing* 2008 (2008), 113.
[39] Ari Juels and Madhu Sudan. 2006. A fuzzy vault scheme. *Designs, Codes and Cryptography* 38, 2 (2006), 237–257.
[40] A. Juels and M. Wattenberg. 1999. A fuzzy commitment scheme. In *Sixth ACM Conference on Computer and Communications Security (ACM CCS)*, G. Tsudik (Ed.). ACM Press, 28–36.
[41] Kenneth Ko. 2007. *User's guide to NIST biometric image software (NBIS)*. Technical Report.
[42] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. 2018. Spectre attacks: Exploiting speculative execution. *arXiv preprint arXiv:1801.01203* (2018).
[43] Peter Komarinski. 2005. *Automated fingerprint identification systems (AFIS)*. Elsevier.
[44] Dongjin Kwon, Il Dong Yun, Duck Hoon Kim, and Sang Uk Lee. 2006. Fingerprint matching method using minutiae clustering and warping. In *18th International Conference on Pattern Recognition (ICPR'06)*, Vol. 4. IEEE, 525–528.
[45] Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado. 2017. Inferring Fine-grained Control Flow Inside {SGX} Enclaves with Branch Shadowing. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 557–574.
[46] Andy Liaw, Matthew Wiener, et al. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.
[47] Jerry Ma, Weining Yang, Min Luo, and Ninghui Li. 2014. A Study of Probabilistic Password Models. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 689–704.
[48] Emanuela Marasco, Stefany Cando, Larry Tang, and Elham Tabassi. 2019. Cross-Sensor Evaluation of Textural Descriptors for Gender Prediction from Fingerprints. In *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*. IEEE, 55–62.
[49] Emanuela Marasco, Luca Lugini, and Bojan Cukic. 2014. Exploiting quality and texture features to estimate age and gender from fingerprints. In *Biometric and Surveillance Technology for Human and Activity Identification XI*, Vol. 9075. International Society for Optics and Photonics, 90750F.
[50] Emanuela Marasco, Luca Lugini, Bojan Cukic, and Thirimachos Bourlai. 2013. Minimizing the impact of low interoperability between optical fingerprints sensors. In *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*. IEEE, 1–8.
[51] Karthik Nandakumar and Anil K Jain. 2015. Biometric template protection: Bridging the performance gap between theory and practice. *IEEE Signal Processing Magazine* 32, 5 (2015), 88–100.
[52] Karthik Nandakumar, Anil K Jain, and Sharath Pankanti. 2007. Fingerprint-based fuzzy vault: Implementation and performance. *IEEE transactions on information forensics and security* 2, 4 (2007), 744–757.
[53] Margarita Osadchy, Benny Pinkas, Ayman Jarrous, and Boaz Moskovich. 2010. SCiFI-a system for secure face identification. In *2010 IEEE Symposium on Security and Privacy*. IEEE, 239–254.
[54] Konstantinos N Plataniotis, Dimitrios Hatzinakos, and Jimmy KM Lee. 2006. ECG biometric recognition without fiducial detection. In *Biometric Consortium Conference, 2006 Biometrics Symposium: Special Session on Research at the*. IEEE, 1–6.
[55] Raul Sanchez-Reillo, Carmen Sanchez-Avila, and Ana Gonzalez-Marcos. 2000. Biometric identification through hand geometry measurements. *IEEE Transactions on Pattern Analysis & Machine Intelligence* (2000), 1168–1171.
[56] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.
[57] Yagiz Sutcu, Qiming Li, and Nasir Memon. 2007. Protecting biometric templates with sketch: Theory and practice. *IEEE Transactions on Information Forensics and Security* 2, 3 (2007), 503–512.
[58] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. DeepFace: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1701–1708.
[59] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. 2018. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient out-of-order execution. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 991–1008.
[60] Zhicheng Wang, Hongwei Zhang, Juan Peng, and Li Yang. 2018. Fingerprint identification based on neural network for large fingerprint database. In *Third International Workshop on Pattern Recognition*, Vol. 10828. International Society for Optics and Photonics, 1082804.
[61] Stephen B Wicker and Vijay K Bhargava. 1999. *Reed-Solomon codes and their applications*. John Wiley & Sons.
[62] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *Annual Symposium on Foundations of Computer Science*. IEEE.
[63] Long Zhang, Zhenan Sun, Tieniu Tan, and Shungeng Hu. 2009. Robust biometric key extraction based on iris cryptosystem. In *International Conference on Biometrics*. Springer, 1060–1069.
[64] Ye Zhang and Farinaz Koushanfar. 2016. Robust privacy-preserving fingerprint authentication. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 1–6.

A SECURE SKETCHES FOR SET DIFFERENCE FROM [29]

For our multisketch construction, we use the secure sketch construction for set-distance from [29]. We give a new (slightly less efficient) construction for set-distance using authenticated encryption scheme (see Section 3). Here we give the original set-distance construction given in [29] for reference purpose.

Let w and $\tilde{w}$ be two sets of size n. We also assume that there is a one-to-one mapping between the set elements and $\mathbb{Z}_{2^k}$, where 2^k is the size of the universe from where the set elements are chosen. Set-distance is defined as $d(w, \tilde{w}) = w \Delta \tilde{w} = |w \cup \tilde{w} - w \cap \tilde{w}|$. The sketching function SS_t^Δ takes as input a set w and outputs a bit-string s (a.k.a, the sketch of w). The recovery function Rec_t^Δ on input s and a new set $\tilde{w}$, outputs w if at least t out of n points match between w and $\tilde{w}$; that is to say $|\tilde{w} \Delta w| \leq 2(n - t)$. The algorithms for sketch and recover are shown in Figure 13.

According to the security analysis in [29], the conditional min-entropy of the message distributions (uniform messages over $\mathbb{Z}_{2^k}^n$)

$\mathrm{SS}^{\Delta}_t(w = \{x_1, \ldots, x_n\})$:

$s \leftarrow () \; ; \; t' \leftarrow 2(n-t)$

for $j = 1, 2, \ldots, t'$ do

$$s_j \leftarrow \sum_{\substack{S \subseteq \mathbb{Z}_n \\ |S|=j}} \left(\prod_{i \in S} x_i \right)$$

return s

$\mathrm{Rec}^{\Delta}_t(s, w' = \{x'_1, \ldots, x'_n\})$:

$t' \leftarrow 2(n-t)$

$f_h(z) \leftarrow z^n + \sum_{i=1}^{t'} s_i z^{t'-i}$

$S \leftarrow \{(x'_i, f_h(x'_i)) \mid i \in \mathbb{Z}_n\}$

Find a $(n - t' - 1)$-degree polynomial f_l that agrees with at least t points in S using $[n, n - t', t' + 1]$ Reed-Solomon decoding [33, 61].

return the roots of $f_h - f_l$.

Figure 13: Algorithms for sketch (SS) and recover (Rec) for secure sketch for set-distances. (From [29].)

given the sketch values is $nk - 2(n-t)k = (2t-n)k$. In our construction, we used $t = 7$, $n = 10$, and therefore, the conditional min-entropy will be at least $4k$ bits.

B DIAGRAMMATIC VIEW OF MULTISKETCH OPERATIONS

A user registers with a user id u and a vector of biometric templates $\mathbf{w} = \mathbf{w}_1 \ldots \mathbf{w}_n$. Multisketch uses two databases $\mathcal{I}$ and $\mathcal{F}$, where $\mathcal{I}$ contains user id u and the corresponding helper data v, and $\mathcal{F}$ contains the random permutation of the templates registered so far. Highlighted are the entries corresponding to a user u_4 with templates $\mathbf{w}_4 = \mathbf{w}_{41} \ldots \mathbf{w}_{4n}$.

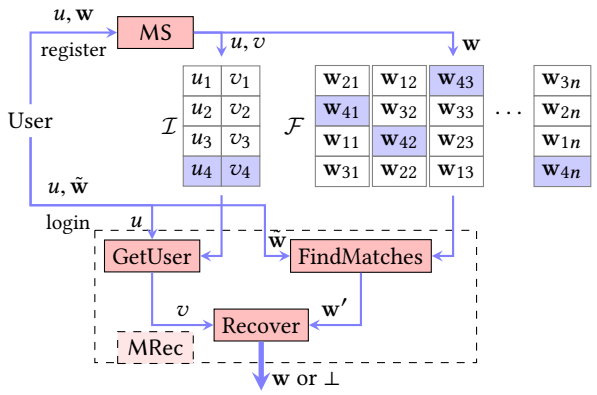

Figure 14: Diagram of multisketch as a part of an authentication service.

C WVU DATA

One of the fingerprint dataset we used for evaluating the accuracy of TenSketch is a subset of the dataset collected through ManTech Innovations Fingerprint Study Phase I [25]. The fingerprints were collected at West Virginia University and released at GMU. We call this dataset WVU dataset. The database contains fingerprints of 500 people acquired using 7 optical sensors like CrossMatch Guardian, I3 digID mini, CrossMatch Seek II as well as ink-based Tenprint cards. Among the participants, most of the participant were in the age group 20-33, accounting for 60.6% percent of people. Among the ethnicity, Caucasians accounted for 57.2%.

Also, there were nearly equal number of male (51%) and female (48%) participants. The users provided the two sets of fingerprints in the sequence of rolled individual fingers on the right and left hands, left slap, right slap, and thumb slap for the live sensor. In order to carry out the experiments of this paper, for every subject we used the fingerprint images of the ten fingers acquired using the I3 digID Mini sensor. Every subject provided two sets of fingerprints each of: rolled fingers on right and left hands, left slap, right slap, and thumb slap. This data has been used in multiple prior work [48, 50].

Attribute	# cls	Base Acc.	ML Acc.	Classifier
Gender	2	88.8%	88.8%	MLP
Class	6	31.9%	**60.3%**	RF
Position	10	10.0%	**28.2%**	MLP, RF
NCIC class	18	56.9%	57.4%	MLP
Scan Type	2	92.3%	93.6%	MLP, RF
FS	2	50.0%	58.9%	MLP

Figure 15: Performance of machine learning classifiers (MLP: multilayer perceptron; RF: random forest, and LR: Logistic Regression) in predicting different attributes of fingerprints. The base accuracy (probability of the most probable class) and the accuracy of the best performing classifier averaged across 5-fold cross-validation is presented in the third and fourth column. The last column notes the best performing classifier. In case the performance of multiple classifiers are similar, we note all of them. Attribute FS denotes whether the fingerprint is from the file (template) or the one used for searching.

D INFERENCE FROM FINGERPRINT TEMPLATES.

We explore what information can be learned from the fingerprints in xyt format (list of minutiae points). The template might reveal some non-trivial information about the finger, such as which finger it is, what is the NCIC class type, or the gender of the user, etc. This information might lead to improved guessing strategy. For example, an attacker can split the database of the fingerprint templates based on the type of the finger.

We tried random forest [46] and multi-layer perceptron [12] to determine each of the attributes: gender, class, finger position, NCIC class type, scan type, and first or second scan (FS). We report the result of our classifier in Figure 15. Notably, finger position and the class of the fingerprint can be retrieved with significant accuracy from the fingerprints (in xyt format).

E TRAINING ADVERSARIAL CLASSIFIERS USING SIAMESE ARCHITECTURE

To construct adversarial classifiers, we tried both deep neural network (DNN) and Siamese networks. In standard DNNs, the order of input features does not carry any information. Therefore, both fingerprint templates are concatenated and passed directly to a DNN model. However, in Siamese networks [56, 58], we can enforce the knowledge that the model is receiving *two* separate sets of features and the goal is to infer some information (identifying if they are from the same person, in our case) about these two sets. Siamese architectures are first shown in [56, 58] to be useful for face recognition and verification.

In Siamese networks, two input feature sets are passed to two sister networks with identical parameters (weights). (See Figure 8 for a high-level diagram of the setup.) The job of two sister networks is to transform the input features into an encoded vector of n_e values, $\mathbf{v}_0$ and $\mathbf{v}_1$. The two vectors are merged using a specific function, e.g.,

absolute value, and passed to a fully connected layer (with single output neuron) that outputs the probability. Two distance functions, i.e., absolute value $|\mathbf{v}_0 - \mathbf{v}_1|$ and the square of differences $(\mathbf{v}_0 - \mathbf{v}_1)^2$ have been used in our experiments. Absolute distance function provided superior performance with $n_e = 64$. We have explored multiple Siamese architectures with various hyperparameters.

The best performing Siamese based classifier underperforms single DNN models. The accuracy with and without the quality values is 66% and 62% respectively.

F GENERALIZED "STITCHING" ATTACK USING CLASSIFIERS

The trained classifiers $\{\mathcal{C}_i\}$ can be used to determine if a message tuple of size i belongs to a single user. An attacker can use a combination of $\mathcal{C}_i$'s to find a t-tuple of fingerprints belonging to the same user. For example, the attacker, given $b = 1$ fingerprint $\mathbf{w}_1$, can use $\mathcal{C}_4$ to find all 4-tuples S_1 likely belonging to the same user and containing $\mathbf{w}_1$. For each tuple $\mathbf{w}_{1\ldots4} \in S_1$, the attacker then uses $\mathcal{C}_4$ again to find remaining three fingerprints, by finding all 4-tuples which include, say, $\mathbf{w}_4$. Let call the second set, S_2. Finally, the attacker can apply MRec on the tuples in S_2, to find a valid 7-tuple. The cost of this attack is $\mathrm{C} = N^3 \cdot c_c + |S_1| \cdot N^3 \cdot c_c + |S_2| \cdot c_s$. Let the false positive rate (FPR) of $\mathcal{C}_4$ be ν_4, then $|S_1| = N^3 \cdot \nu_4$, and $|S_2| = |S_1| \cdot N^3 \cdot \nu_4 = N^6 \cdot \nu_4^2$. Therefore, $\mathrm{C} \approx N^6 \cdot \nu_4^2 \cdot c_s = \mathcal{O}(N^6)$.

More generally, an attacker can use a sequence of $\mathcal{C}_i$'s to stitch a t-tuple. We can find all possible such classifier combinations using the following equation.

$$\sum_{i=2}^{t} (i-1) \cdot x_i = (t-b), \text{where } x_i \in \{0, 1, \ldots, t\}.$$

Each solution of $\{x_i\}$'s in the above equation is a valid classifier combination, where x_i denotes the number of times the classifier $\mathcal{C}_i$ is used in the sequence. In the previous example, $\mathcal{C}_4$ is used twice, therefore $x_4 = 2$ and all other $x_i = 0$. If we assume the classifiers' performances are independent, the effective FPR of the stitched classifier would be $\Pi_{i=2}^{t} \nu_i^{x_i}$, where ν_i is the FPR of classifier $\mathcal{C}_i$. Therefore, for any combination of classifiers, the total number of t-tuples that need to be checked through MRec is $N^{t-b} \cdot \Pi_{i=2}^{t} \nu_i^{x_i} = \mathcal{O}(N^{t-b})$. A stitched classifier with a long combination of classifiers — high value of $\sum x_i$ — might have lower FPR, for example, $x_2 = 6$. However, a long combination of classifiers will also degrade the true positive rate of the stitched classifier; the attacker might not find the correct tuple at the end due to one of the classifiers misclassified it. We therefore adjust the classifiers to have a true positive rate (TPR) of 0.9 or more.

We empirically compute the FPR of the stitched classifiers for all possible combinations. There are only 11 possible solutions to the above equation for $b = 1$ and $t = 7$. We found $x_7 = 1$ has the least FPR, 0.1, and $x_2 = 6$ has the second lowest FPR. In Section 7 we give different attacks using $\mathcal{C}_7$ and $\mathcal{C}_2$ and show their efficacy. Overall, we show that no the attacker gets any better than 3.3-bits of advantage via different combinations of classifiers.