Secure Non-Interactive User Re-Enrollment in Biometrics-based Identification and Authentication Systems

Ivan De Oliveira Nunes[1,2], Karim Eldefrawy[2], and Tancrède Lepoint[2]

[1] University of California Irvine, USA
ivanoliv@uci.edu
[2] SRI International, USA
{karim.eldefrawy,tancrede.lepoint}@sri.com

Abstract. Recent years have witnessed an increase in demand for biometrics based identification, authentication and access control (BIA) systems, which offer convenience, ease of use, and (in some cases) improved security. In contrast to other methods, such as passwords or pins, BIA systems face new unique challenges; chiefly among them is ensuring long-term confidentiality of biometric data stored in backends, as such data has to be secured for the lifetime of an individual. Cryptographic approaches such as Fuzzy Extractors (FE) and Fuzzy Vaults (FV) have been developed to address this challenge. FE/FV do not require storing any biometric data in backends, and instead generate and store helper data that enables BIA when a new biometric reading is supplied. Security of FE/FV ensures that an adversary obtaining such helper data cannot (efficiently) learn the biometric. Relying on such cryptographic approaches raises the following question: *what happens when helper data is lost or destroyed (e.g., due to a failure, or malicious activity), or when new helper data has to be generated (e.g., in response to a breach or to update the system)*? Requiring a large number of users to physically re-enroll is impractical, and the literature falls short of addressing this problem. In this paper we develop `SNUSE`, a secure computation based approach for non-interactive re-enrollment of a large number of users in BIA systems. We prototype `SNUSE` to illustrate its feasibility, and evaluate its performance and accuracy on two biometric modalities, fingerprints and iris scans. Our results show that thousands of users can be securely re-enrolled in seconds without affecting the accuracy of the system.

1 Introduction

Current Biometrics-based Identification and Authentication (BIA) systems[3] [1,2] typically store in the backend a reference Biometric Templates (BTs) of a users' biometrics, such as fingerprint minutiae points and/or iris code. If the backend is compromised or breached, sensitive biometric data of a large number of users

[3] We omit explicitly mentioning access control, we assume it implicitly when authenticating an individual and then granting access based on the authenticated identity.

may be leaked and enable adversaries to impersonate users and circumvent BIA systems. For example, in 2015, the Office of Personnel Management was compromised and led to the leakage of 5.6 million fingerprints of federal workers that applied for security clearances in the United States of America [3].

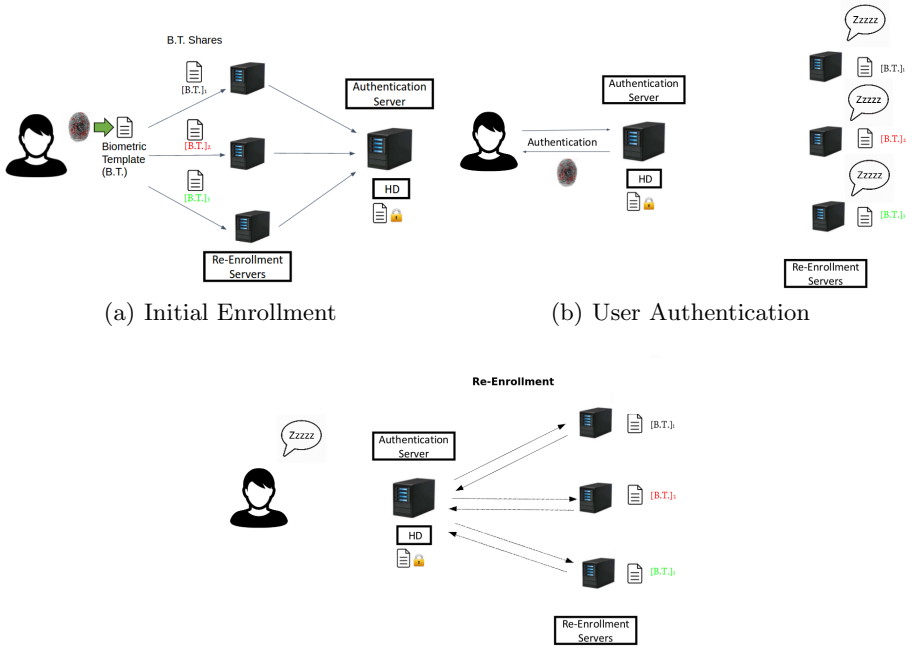
In order to protect BTs, typical solutions use secure elements or encryption. Secure elements are applicable when matching is performed against a few biometrics (e.g., the Touch ID technology on iPhones) but do not scale to a large number of biometrics. Encrypting BTs also brings challenges: (1) BTs have to be decrypted to match against the newly supplied biometric readings, (2) the decryption keys associated with the encrypted templates have to be stored somewhere close by (logically and maybe even physically) to perform matching, and (3) when the backend is compromised or breached, the encrypted templates may be leaked. As reference biometrics templates have to be protected for the lifetime of individuals, it is important to select algorithms and key sizes that remain secure for several decades (say, 40-50 years), which remains a challenging task.

A third cryptographic approach to address the above shortcomings and to construct secure BIA systems has been proposed in the form of Fuzzy Vaults (FV) [4] and Fuzzy Extractors (FE) [5]. FE and FV alleviate the need to store BTs in the system's backend; they enable to perform matching during the normal operation of a system from some Helper Data (HD), extracted during the user's initial enrollment into the system. The HD securely encodes[4] a secret, or cryptographic key, that cannot be retrieved unless the same (noisy) biometric used to generate the HD is provided as input. Thus, the system can determine if a new reading of a biometric corresponds to the user being authenticated. The security of the approach stems from the fact that HDs do not convey any information about the underlying biometric; and this guarantee can be information-theoretic/statistical or computational depending on the details of the FE/FV scheme.

To deploy this third cryptographic approach at scale, one has to address the challenge of re-enrollment: in the lifetime of a cryptographically secure BIA system, it will be often necessary to re-generate HDs because of breaches or to perform maintenance or updates. For example, the secret key may need to be revoked and replaced by a fresh one when the HD is leaked, damaged, or corrupted. A second example is when access control is enforced via encryption, e.g., via attribute-based encryption [6]; in this case, changing a given user's permissions implies changing the user's cryptographic keys. In both examples, existing schemes require physical presence of the user to refresh cryptographic material, which is laborious, slow, costly, and hence impractical. This problem is currently not addressed in the literature.

Our contributions. In this work we introduce `SNUSE` (Secure Non-interactive User at Scale Enrollment), a new approach for secure user re-enrollment that does not require user involvement, nor storing BTs, in the clear or in encrypted

[4] We use the term "encode" very loosely here as helper data may not actually encode the secret, it may only enable constructing it when the biometric is also present.



(a) Initial Enrollment (b) User Authentication

(c) User Re-Enrollment

Fig. 1. Initial user enrollment, user authentication, and user re-enrollment in SNUSE. During regular authentication, the user interacts with the authentication server only, which stores the HD of all the users and enables recovering the user's secret/key when the correct biometric is supplied. When re-enrollment is required, the authentication server communicates with the re-enrollment servers and uses MPC to compute new HD, which encodes a new secret/key, from the secret-shared BT. We emphasize that regular authentication does not require involvement of the re-enrollment servers and, conversely, the re-enrollment phase does not require user involvement.

form, in any single backend server. Instead, SNUSE uses secret sharing to distribute the original biometric templates (BTs) among several offline components, and performs the re-enrollment in a secure distributed manner by computing the required FE/FV HD generation algorithms using efficient secure multiparty computation (MPC). This approach ensures that at no point during the system's operation (after the initial enrollment) are original reference BTs reconstructed in the clear. The components storing shares of BTs can remain offline and inaccessible through the network during normal operation; when re-enrollment is required, the components are brought online and connected to the system for only a brief period (e.g., seconds). Figure 1 illustrates the initial enrollment, subsequent authentication, and re-enrollment phases in SNUSE. To the best of our knowledge, this is the first approach amending BIA systems to enable performing non-interactive user re-enrollment without storing the user's biometric

template in clear, or encrypted, in any single component or server. A detailed comparison with related work can be found in Appendix E. Due to space constraints, we defer the details of the implementation of a `SNUSE` prototype using Fuzzy Vaults in Appendix A; experimental results, and security analysis can be found in Appendices B to D. Our experimental results demonstrate a high detection accuracy with over 90% Genuine Acceptance Rate (GAR) and less than 5% False Acceptance Rate (FAR) when the right degree of polynomial is used (see Figs. 6(a) and 6(b)). Finally, we show in Appendix C that re-enrollment of thousands of users using standard computing servers only takes a few seconds.

2 Background

We overview here the building blocks used in `SNUSE`, and introduce the notation that will be used in the rest of the paper.

2.1 Biometrics-Based Authentication

During user enrollment into a BIA system, a reference Biometric Template (BT), composed of features uniquely identifying an individual, is sampled and stored at an Authentication Server (AS). Later, when a user tries to authenticate, a biometric reading is collected and the same feature extraction process is applied to generate a second BT. This new BT is compared to the stored one, and if their similarity exceeds a given threshold, the user is successfully authenticated, otherwise, user authentication fails. A feature extraction procedure, applied to a biometric sample of a user U, results in a BT that can be represented as:

$$BT_U = \{p_1, \ldots, p_M\}\,, \tag{1}$$

where each p_i is a data point representing a unique detail of U's biometric. For example, fingerprint BTs include location and orientation of *minutiae*, i.e., of regions in the image where fingerprint lines merge and/or split. Such minutiae points are encoded as:

$$p_i = (x_i, y_i, \theta_i)\,, \tag{2}$$

where, x_i is the x coordinate, y_i is y coordinate, and θ_i the orientation of the minutiae i extracted from U's fingerprint. Analogous methods can be proposed to encode other biometrics such as iris scans (see Appendix A).

Traditional BIA systems store BTs of thousands (or even hundreds of thousands) of users in clear form. The reason for this is that each time a biometric is sampled by a sensor, it is slightly different due to noise. Standard mechanisms for secure storage of passwords (e.g., salted-hashing) cannot match two noisy readings of the same biometric because they are not exactly the same. Unfortunately, the advantages of biometrics come with a high risk; if the leaked biometric's modality is stable, leakage of a user's biometric at any point in time affects security of *all* authentication systems using this biometric for years. Consequently, protecting the confidentiality of biometric data is of utmost importance.

As discussed in Section 1, Fuzzy Extractors (FE) and Fuzzy Vaults (FV) are cryptographic solutions that use an input BT to (i) generate Helper Data (HD), which encodes a secret/key k; and (ii) ensure that the HD does not reveal anything about the BT or k, unless prompted with (a noisy version of) the same biometric used to generate the HD. Note that k can be an arbitrary secret and not necessarily a symmetric key. In current approaches, the secret k stored in the HD cannot be refreshed without requiring the physical (or remote) presence of user for the re-enrollment process in which a new biometric sample is collected and new HD is computed for the new secret. This manual approach does not scale as re-enrolling thousands (or more) users is impractical; this papers proposes a MPC-based solution that enable large-scale automatic re-enrollments.

2.2 Secret Sharing

In K-out-of-N secret sharing [7] a dealer distributes a secret among N parties such that subsets of K or more parties can recover it; knowing up to $K-1$ shares leaks no information about it. In `SNUSE`, we will generate N shares of a biometric template and store them on N re-enrollment servers. Given a secret X, let $[X]_j$ denote the j-th share and denote the generation of N shares of X by:

$$\{[X]_1, \ldots, [X]_N\} \leftarrow X\,. \tag{3}$$

Denote the reconstruction of secret X from K shares by:

$$X \leftarrow \{[X]_1, \ldots, [X]_K\}\,. \tag{4}$$

In Shamir's (K, N) secret sharing scheme [7] over a finite field $\mathbb{F}$, one randomly generates the coefficients of a polynomial P of degree $d = K-1$. The independent term a_0 is then set to the secret X, and one can generate N secret shares $\Phi = \{(i, P(i))\}_{i=1}^{N}$. Since P has degree $K-1$, K points in Φ are enough to interpolate P and reconstruct its coefficients, including the secret $a_0 = X$. A set of $L < K$ shares does not reveal any information about X because there's an infinite number of polynomials of degree K that can be constructed from these points.

2.3 Secure Multi-party Computation (MPC)

MPC protocols enable mutually distrusting parties to jointly compute a function f of their private inputs while revealing no information (other than the output of f) about their inputs to the other parties [8].

In standard algebraic MPC protocols, each party usually generates shares of its input (using, for instance, Shamir's secret sharing scheme) and distribute one share to each party. A key observation is that if one is able to compute both addition and multiplication on the shares, such that the resulting shares can be combined into the correct result of the operations, one can implement any function f from these two basic operations. Different schemes were proposed

to compute addition and multiplication over private inputs [9–14]. Nevertheless, most of them share the following common characteristics in the computation of these operations:

- **Addition** of secret shares can be computed locally. To that purpose each party computes addition of its own secret shares. The N local results, once combined, yield the result of an addition of the actual secret(s).
- **Multiplication** of secret shares requires communication. Even though different schemes exist, most require parties to broadcast an intermediate (blinded) result during the computation of multiplication, such that individual shares of the multiplication result can be correctly computed.

We do not specify details of the multiplication sub-protocols and we refer the reader to [11, 12] for further details. The takeaway is that multiplication requires communication between parties, and in practice the overhead to multiply is usually orders of magnitude higher than the overhead of addition. As we describe in Section 3, `SNUSE` uses MPC to securely compute an FV generation algorithm from secret shared biometric templates in the honest-but-curious threat model.[5] In our design we take advantage of the unique characteristics of a biometric enrollment system to reduce the number of multiplications to a minimum (sometimes even eliminating it), allowing cost-effective scalable MPC-based user re-enrollment.

2.4 Fuzzy Vault Scheme

A Fuzzy Vault (FV) scheme [4] is designed to work with a BT represented as an unordered set of points as shown in Eq. (1). The security of FV relies on the infeasibility of the polynomial reconstruction problem [15]. FVs allow using a user's biometric template BT_U to hide a secret k. The secret can be, for instance, some private data or a cryptographic key.

A FV scheme consists of two algorithms, a generation part (FV_{GEN}) and a secret reconstruction part (FV_{OPEN}), which can be informally defined as follows:

- FV_{GEN}: receives as input the secret k and a biometric template BT_U. It uses BT_U and k to generate a Helper Data HD. This HD encodes the secret k without revealing information about k and BT_U:

$$HD = FV_{GEN}(BT_U, k) \tag{5}$$

- FV_{OPEN}: receives as input the HD and BT'_U. It retrieves k from HD if, and only if, BT'_U is a template extracted from the same biometric as extracted in the input BT_U to FV_{GEN} when generating HD. In other words, given a distance function D and a threshold w:

[5] In the Honest-But-Curious (HBC) model, corrupted parties collaborate to learn private inputs of other parties but they do not deviate from the protocol specification.

$$FV_{OPEN}(BT'_U, HD) = \begin{cases} k & \text{if } D(BT_U, BT'_U) \leq w \\ \perp & \text{otherwise} \end{cases} . \tag{6}$$

The threshold w is a security parameter that allows control of the trade-off between minimizing false acceptance (revealing k to the wrong user) and false rejection (refusing to reveal k to the rightful user).

FV_{GEN} algorithm starts by selecting a polynomial P of degree d defined over a field $GF(2^\tau)$ and encoding (or splitting) the secret k into the $d+1$ coefficients $a_0, \ldots, a_d$ of P. The resulting polynomial is defined as:

$$P_k(x) = \sum_{i=0}^{d} a_i x^i \tag{7}$$

where the coefficients $\{a_0, \ldots, a_d\}$ are generated from k and can be used by anyone to reconstruct k. Since P is defined over $GF(2^\tau)$, each coefficient can encode τ bits; this implies that the maximum size of the secret k that can be encoded is $(d+1) \times \tau$. After encoding k as $P_k(x)$, each of the M data points in BT_U is evaluated with the polynomial $P_k(x)$ generating a set of points in a two-dimensional space:

$$L_P = \{(p_1, P_k(p_1)), \ldots, (p_M, P_k(p_M))\} . \tag{8}$$

Note that the field must be large enough to encode a data point from BT_U as a single field element. The resulting set L_P contains only points in the plane that belong to the polynomial $P_k(x)$. In addition to L_P, a set of chaff points L_S of size $S \gg M$ is generated by randomly selecting pairs (r_x, r_y), where r_x and $r_y \in GF(2^\tau)$ resulting in:

$$L_S = \{(r_{x1}, r_{y1}), \ldots, (r_{xS}, r_{yS})\} \tag{9}$$

Finally, L_P and L_S are shuffled together using a random permutation π and the result is included in the HD:

$$\pi(L_P + L_S) \in HD \tag{10}$$

The HD also includes the set of public parameters $\Phi = \{\mathbb{F}, d, M, H(k)\}$, where $\mathbb{F}$ is the field in which $P_k(x)$ is defined and d is its degree, M is the size of BT_U, i.e., the number of points in the HD that belong to $P_k(x)$, and $H(k)$ is a cryptographic hash of the secret k allowing one to verify if the correct secret was reconstructed using FV_{OPEN}.

The key idea behind security of the FV scheme is that with $d+1$ distinct points $(p_i, P_k(p_i))$, one can interpolate $P_k(x)$, retrieve its coefficients and thus recover k. However, finding which $d+1$ points to interpolate out of the $M+S$ in HD is hard if $M+S$ is sufficiently larger than d.

When attempting to reconstruct k from the HD using a new biometric reading BT'_U, the FV_{OPEN} algorithm will use a distance function (which must be

defined according to the biometric type) to select, out of the $M+S$ points in the HD, the M points that are the closest matches to the points in BT'_U. If, out of the M selected points, at least $d+1$ points are points that belong to the original L_P, then the algorithm will be able to interpolate the correct polynomial and recover k. To verify that k was correctly recovered, the algorithm hashes the result and compares it to $H(k)$, which was published together with the HD. If less than $d+1$ correct points are among the M points selected via distance matching, no interpolation with combinations of $d+1$ points out of M will yield a match in the hash, because $P_k(x)$ will not be interpolated correctly. Therefore, FV_{OPEN} will reject BT'_U.

Note that the FV scheme does not rely on the order of the elements in BT_U and BT'_U and does not require all points to be present in both templates. Instead, $d+1$ data points in BT'_U must be close enough to points in BT_U. In that sense, the polynomial degree d acts as a security parameter that allows calibration of the scheme to reduce false acceptance by increasing the required number of matching data points.

3 The SNUSE Approach

Figure 1 illustrates the operations of SNUSE during *a*) initial user enrollment, *b*) regular user authentication, and *c*) non-interactive user re-enrollment. SNUSE involves three types of parties: a User (U), an Authentication Server (AS), and n Re-Enrollment Servers ($\{RES_1, \ldots, RES_n\}$).

During the initial enrollment of U into the system, U's biometric template BT_U is secret shared into n shares $([BT_U]_1, \ldots, [BT_U]_n)$; each share is distributed to one of the n Re-Enrollment Servers ($\{RES_1, \ldots, RES_n\}$). The RESs then jointly generate U's secret k_U and use an MPC protocol to compute FV helper data, HD_{k_U}, from their shares $([BT_U]_1, \ldots, [BT_U]_n)$, thus securely "locking" k using the user's BT_U.

Regular user authentication happens between U and AS. Since AS only stores HD_{k_U} (which reveals nothing about BT_U), the FV_{OPEN} algorithm must be used to retrieve k. Now, if HD_{k_U} was correctly computed using a secure FV_{GEN} algorithm, the only way to retrieve k from HD_{k_U} is by providing a second biometric template BT'_U which is close enough to the original BT_U used in computation of FV_{GEN}. Therefore, $FV_{OPEN}(BT'_U, HD_{k_U})$ will successfully retrieve k if and only if $BT'_U \approx BT_U$, i.e., BT'_U is a noisy version of the same biometric used to generate the HD. After this stage, k can be used in standard cryptographic primitives, e.g., decrypt a user's files or messages, or grant them access to resources based on the recovered secret.

Whenever k needs to be replaced with a fresh secret k' (this process may happen periodically within an organization to guarantee freshness of users' cryptographic keys), RESs are brought online and connected to the system, and AS issues a request to the RESs to compute a new HD for U. The RESs will securely generate a new k' and (as in the enrollment protocol) use U's secret shares $[BT_U]_1, \ldots, [BT_U]_n$, stored during U's enrollment, to compute a new $HD_{k'_U}$. This

way, users' cryptographic material can be refreshed and brand new HD can be constructed without bringing the user in to re-sample her biometric and without storing her biometric template in clear.

At a first sight, a simple approach for generating a fresh k' would be to multiply the y-coordinates in the FV by a random σ, which would result in a fresh random key $k' = k \cdot \sigma$ encoded in the FV. However, this approach does not work for several reasons. First, as discussed before, k might not be an independent random byte-stream, such as a symmetric key; it could, for instance, include a set of user permissions or an asymmetric private-key (sk') associated with the user's public-key. In the latter case, the generation of fresh $k' = sk'$ implies deriving the corresponding new public-key (pk'). `SNUSE` can handle these cases while simple multiplication by randomness can not. Second, while this approach using randomization might work for a classic FV with symmetric keys, because the key is encoded as coefficients of a polynomial, it does not necessarily apply to other FV/FE constructions; `SNUSE` on the other hand is a generic approach (as any computable function can be computed using MPC), that could be implemented with other FV/FEs. Third, even in the case where the scheme uses a classic FV to encode a random symmetric key, multiplying by randomness σ will update the encoded secret but will prevent the reconstruction of the secret, i.e., $k' = FV_{OPEN}(BT'_U, HD)$ cannot be computed; this is because FV_{OPEN} must verify the hash of each candidate recovered key with the stored $H(k)$ to decide if the correct key was reconstructed. However, $H(k')$ cannot be updated in the same way, because for any reasonable hash function, $H(\sigma \cdot k) \neq \sigma \times H(k)$.

Throughout the rest of this section we describe the steps of `SNUSE` in more details. We have implemented `SNUSE` with fingerprints and iris scans, and evaluated `SNUSE` performance in terms of computation and storage requirements. Our evaluation shows that `SNUSE` can re-enroll thousands of users in seconds; the storage requirements for thousands of users is a few MBytes. Though `SNUSE` does not affect the accuracy of the underlying biometric feature extraction tool, we also provide accuracy results for completeness. For implementation and evaluation details, we defer the reader to Appendices A to C; Appendix D contains a security analysis of `SNUSE`.

Remark 1. All message exchanged in the following protocols are assumed to be through secure authenticated channels, such as standard TLS. Establishing such secure channels is omitted from the protocols for the sake of clarity.

3.1 Initial User Enrollment

The initial user enrollment, presented in Fig. 2, is the only protocol in `SNUSE` that involves all parties, i.e., U, AS, and RES_i $\forall i \in [1, N]$. This protocol is executed only once for each user, all interactions after the initial enrollment are performed either between U and AS (authentication), or between AS and $RESs$ (re-enrollment).

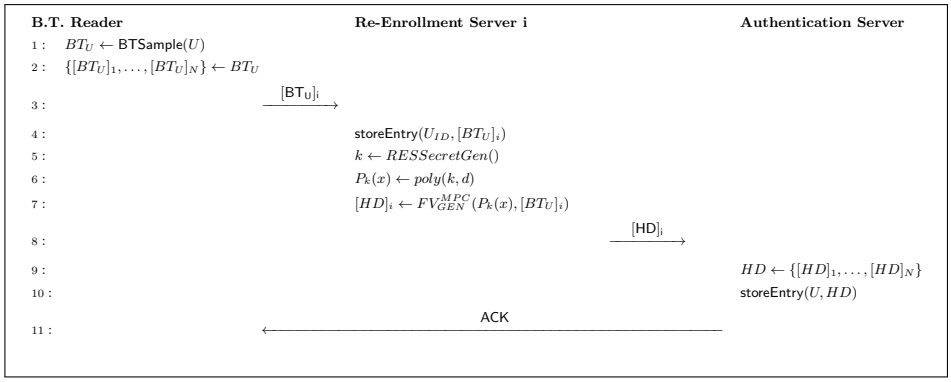


Fig. 2. User enrollment protocol in `SNUSE`

The protocol starts with U using a trusted enrollment device (e.g., fingerprint sensor, iris reader), referred to as B.T. Reader, to measure and extract U's biometric template BT_U (Fig. 2, line 1). BT_U is then split into N secret shares, where N is the number of RESs. Each share $[BT_U]_i$ transmitted to, and stored by, the respective RES_i (Fig. 2, lines 2-4). Note that in Fig. 2 we only depict one RES_i, however, in reality each of the N RESs receives and stores its share $[BT_U]_i$.

Once each share $[BT_U]_i$ is stored on the corresponding RES_i, the RESs agree on the new authentication material k for the user, by using $RESSecretGen$ (Fig. 2, line 5). $RESSecretGen$ may be implemented by simply generating k at one of the RES_i and transmitting the value of k to the other RESs through secure channels; alternatively, group key agreement protocols could be utilized depending on the structure of k. Each RES then encodes k as a polynomial of degree d, denoted as $P_k(x)$. The polynomial $P_k(x)$ is the same polynomial described in the standard FV scheme (see Section 2). The difference is that, instead of generating the HD with BT_U, as in the standard FV scheme, the protocol uses $FV_{GEN}^{MPC}(P_k(x), [BT_U]_i)$, which uses MPC to compute a share of the HD ($[HD]_i$) from the secret shared $[BT_U]_i$. Each of the RESs computes the same function with its own share. Finally, each share $[HD]_i$ is then sent to AS, which reconstructs the actual HD from the received shares and acknowledges the completion of the enrollment protocol. This is depicted in Fig. 2, lines 9-11. Finally, AS stores an entry for newly generated HD, associated with user U.

Note that, during user enrollment protocol, BT_U is only visible in clear to the trusted sensor device that reads and then secret shares the biometric. Each RES_i only sees its own share which leaks no information about BT_U itself. AS only sees HD, which can not be used to reconstruct BT_U by the security of FV. Therefore, confidentiality of the biometric is ensured during user enrollment. In fact, there is no single server from which BT_U can be retrieved in clear. BT_U only

exists in clear ephemerally at the B.T. Reader and that must happen anyway because B.T. Reader is the sensor device used to sample the user's biometric.

3.2 User Authentication

A consequence of correct computation of HD using MPC in the enrollment phase is that the user authentication protocol consists of simply using standard local computation of FV_{OPEN} with a new biometric reading BT'_U and the stored HD. The RESs do not participate in user authentication, but only in the enrollment and re-enrollment protocols.

The authentication protocol, shown in Fig. 3, starts with a user supplying its ID (U_{ID}) and biometric sample to the B.T. Reader. A biometric template BT'_U is generated from the new sample and kept locally. U_{ID} is sent to AS which fetches U's HD from its database based on the supplied U_{ID}, and sends the associated HD as a reply. BT'_U is then used to extract k from HD using the FV_{OPEN} algorithm. Note that here, and similar to user enrollment, U's BT only exists in clear in the B.T. Reader.

After retrieving the secret k, the user can authenticate to the server using, for example, standard challenge-response mechanisms based on the secret k. Suppose, for instance, that k is actually part of an asymmetric encryption scheme (sk, pk), where $k = sk$ and pk is a public key, known to AS, associated to the secret key sk. AS can then authenticate the user by sending a challenge $Enc_{pk}(nonce)$, where $nonce \leftarrow_{\$} \{0,1\}^n$. If the user is able to retrieve k from the FV, it can then use $k = sk$ to compute $nonce \leftarrow Dec_{sk}(Enc_{pk}(nonce))$ and send the result back to AS, proving its claimed identity.

3.3 Non-Interactive User Re-Enrollment

Non-interactive user re-enrollment works by having the RESs compute a new HD based on a fresh secret k'. Since the shares of the biometric template are stored during the initial enrollment, this step does not need user involvement, even though the biometric template does not exist in clear neither at RESs nor AS.

In this protocol, AS sends a request for re-enrollment for each RES_i, containing the ID of the user. The RESs will then jointly generate a new k' for the user and encode it as a polynomial of degree d. Each RES_i then uses U_{ID} to fetch the secret share $[BT_U]_i$ associated with U_{ID} and uses FV_{GEN}^{MPC} to compute $P_{k'}(x)$ on the secret share $[BT_U]_i$. The result is a secret share $[HD]_i$ for a brand new HD which encodes the freshly generated k' under the same biometric template BT_U. This process is depicted on Fig. 4. Finally, AS receives all N shares $[HD]_i$ and compute the new HD, which can, from this point on, be used for authenticating user U with the protocol in Fig. 3. Notice that, during the execution of the re-enrollment protocol, BT_U is not reconstructed in clear at any point. This is only possible because of the computation of HD using MPC over

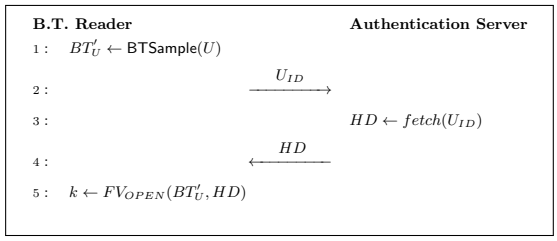


Fig. 3. User authentication protocol in SNUSE

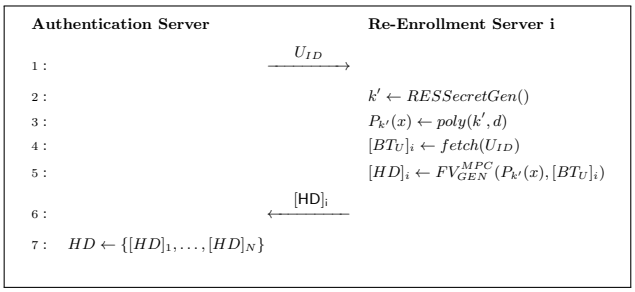


Fig. 4. Re-Enrollment protocol in SNUSE

the secret shares. Otherwise, this process would require either *i*) user involvement to collect another biometric reading or *ii*) storing the biometric template in clear in the backend servers.

3.4 Using MPC to generate the HD

The fundamental part of SNUSE that allows non-interactive user re-enrollment (without storing BT_U in clear) is the ability to compute the HD from the secret shares $\{[BT_U]_1, \ldots, [BT_U]_N\}$ of BT_U. In the protocols of Fig. 2 and Fig. 4, this is represented by the computation of the function $FV_{GEN}^{MPC}(P_k(x), [BT_U]_i)$, resulting in a secret share $[HD]_i$ that can be interpolated to reconstruct the actual HD.

In this section, we discuss how FV_{GEN} is computed from the secret shares. We start by outlining the basic operations needed to compute FV_{GEN}, and then describe details of how each is performed using secret shared data. The standard local computation of FV_{GEN} algorithm, involves three basic types of operations:

1. Evaluation of the polynomial $P_k(x)$, that encodes k, on each of the M data points ($\{p_1, \ldots, p_M\}$) that compose BT_U to generate the list of points in the polynomial P_k:
$$P = \{(p_1, P_k(p_1)), \ldots, (p_i, P_k(p_i)), \ldots, (p_M, P_k(p_M))\} \tag{11}$$
2. Generation of a list S composed of s random chaff points (r_x, r_y), to be shuffled together with the polynomial points:
$$S = \{(r_{x,1}, r_{y,1}), \ldots, (r_{x,i}, r_{y,i}), \ldots, (r_{x,s}, r_{y,s})\} \tag{12}$$

3. Random permutation π to shuffle the elements of lists R and P together generating the HD:

$$HD = \pi(P \cup R) \tag{13}$$

Steps 2 and 3 are relatively easy to compute when compared step 1. For the random chaff point generation (Step 2), each RES_j computes a random share $[(r_{x,i}, r_{y,i})]_j$. When the randomly generated shares are merged together they will result in random chaff points.

For Step 3, all M RESs agree on a single random permutation π, and all of them permute their secret shares according to this same randomly chosen permutation. Note that the permutation π is kept secret from AS, because knowing π would allow an adversary who takes control of AS to separate chaff points from the points in the polynomial by computing π, allowing reconstruction of BT_U. Nevertheless, even though AS does not know which permutation was used, because each RES use the same permutation to compute $\pi(P \cup R)$ on their shares, each share will be matched to its correct set of shares during the reconstruction of the HD at AS.

Since we are able to generate random chaff points and compute a permutation on the secret shares (which results in a permutation on the reconstructed HD), the remaining task for FV_{GEN} is to compute the polynomial P_k using MPC on each of the secret shares. We discuss the classic approach to compute P_k using MPC and then we introduce our optimized version that takes advantage of pre-computation of secret exponents before the secret sharing phase.

In the classical approach, assuming that BT_U is composed of M data points, each secret share $[BT_U]_j$ corresponding to RES_j would be a vector in the form:

$$[BT_U]_j = \{[p_1]_j, \ldots, [p_M]_j\} \tag{14}$$

The polynomial P_k can be generically defined as:

$$P_k(x) = \sum_{i=0}^{d} a_i x^i \tag{15}$$

where:

$$\{a_0, \ldots, a_d\} \leftarrow k \tag{16}$$

denoting that the coefficients of the polynomial encode k. Therefore, computing $P_k(x)$ implies computing exponentiation up to degree d on the secret shared variables, multiplication of the resulting values by the respective constants $a_0, \ldots, a_d$, and addition on the resulting terms $a_i x^i$ for all $i \in [1, \ldots, d]$.

As discussed in Section 2, the bulk of the overhead on MPC comes from multiplication of secret shares, because addition (and consequently multiplication by a constant) can be computed locally by adding the secret shares. Multiplication, on the other hand, requires communication, since the parties must publish (broadcast) intermediate results to all other parties that hold secret shares. Since each multiplication involves network delays this is usually the major source of overhead in the MPC evaluation.

The computation of x^d, with no optimization, requires d multiplication operations, i.e., computing $\prod^d x$. In such approach, computing all terms in Eq. (15) would take $\sum_{i=0}^{d} i$. Therefore, the number of communication rounds to compute the HD shares would be:

$$T = M * \sum_{i=0}^{d} i \tag{17}$$

In terms of asymptotic complexity this *naive* approach yields $\Theta(d \times log(d))$ multiplications, where d is the polynomial degree. Such number of multiplications can be trivially reduced to d if we take into account that $x^n = x \cdot x^{n-1}$. This implies that the result of a lower order polynomial term can be used as an intermediate result for the computation of the subsequent term, reducing the number of communication rounds to compute the HD to:

$$T = M * d \tag{18}$$

resulting in linear asymptotic complexity of $\Theta(d)$ for the number of required multiplications.

To make the process more efficient, we take a different approach. We bring the number of necessary multiplications to zero by pre-computing the powers of x before distributing the shares to the RESs. From the standard BT_U, which is a vector of M data points in the format $\{p_1, \ldots, p_M\}$, we pre-compute the x^i exponents for all $i \in [1, d]$ and secret share each of the pre-computed exponents for each data point. Therefore, a secret share $[BT_U]_j$ of a biometric template with pre-computed exponents becomes a $d \times M$ matrix in the format:

$$[BT_U]_j = \begin{pmatrix} [(p_1)^1]_j & \cdots & [(p_M)^1]_j \\ \vdots & & \vdots \\ [(p_1)^i]_j & & [(p_M)^i]_j \\ \vdots & & \vdots \\ [(p_1)^d]_j & \cdots & [(p_M)^d]_j \end{pmatrix} \tag{19}$$

Each column in $[BT_U]_j$ represents a data point p_i and each line an exponentiation of such data point. For example, line 3, column 4, would contain a secret share of the fourth data point in BT_U raised to the cubic power: $[(p_4)^3]_j$.

In this approach, a secret share is included for each of the exponents of each data point. Thus, the evaluation of the polynomial requires no multiplications of secret shares because all exponents are now individual secret shares in the matrix $[BT_U]_j$. Therefore, the computation of $[HD]_j$ to be done locally. Specifically, let $[BT_U]_j(x, y)$ denote the element in line x and column y of the matrix[6]. Then $[y]_j = P_k([p_i]_j)$ can be computed locally for every i as:

$$P_k([p_i]_j) = a_0 + \sum_{k=1}^{d} a_k \times [BT_U]_j(k, i) \tag{20}$$

[6] In our notation the first row/column of a matrix is indexed by 1 and not 0.

where $\{a_0, \ldots, a_d\}$ are the polynomial coefficients defined in Eq. (15).

This eliminates the need for network communication making the scheme much faster. Such approach is feasible because in practice $d \approx 10$ and because one single entity (B.T. Reader) possesses all data points for enrollment. This is a necessary condition for user enrollment in biometric authentication systems (the BT must be read by a given sensor). We take advantage of that to improve efficiency of `SNUSE` by pre-computing the exponents and including them in the secret shares of the biometric templates. As detailed in the experiments of Appendix C, by pre-computing the exponentiations, `SNUSE` achieves high performance in terms of processing time with reasonable storage requirements that are comfortably within the capacity of modern computers. Algorithm 1 synthesizes

Algorithm 1: FV_{GEN}^{MPC} computation on RES_j

inputs : Secret share matrix $[BT_U]_j$ (Eq. (19)); fresh secret k; random permutation π; number of chaff points s; and polynomial degree d.
output: $[HD]_j$

1 $\{a_0, \ldots, a_d\} \leftarrow$ EncodeAsPolynomialCoeffs(k, d);
2 $L_p \leftarrow$ emptyList()
3 **forall the** $i \in$ *[1, 2, . . . , M]* **do**
4 $\quad pi_x \leftarrow [BT_U]_j(1, i)$
5 $\quad pi_y \leftarrow a_0 + \sum_{k=1}^{d} a_k \times [BT_U]_j(k, i)$ /* *MPC* */
6 $\quad L_p.append([pi_x, pi_y])$
7 **end**
8 $L_s \leftarrow$ emptyList()
9 **forall the** $i \in$ *[1, 2, . . . , s]* **do**
10 $\quad r_x \leftarrow rand_{GF(2^\tau)}()$
11 $\quad r_y \leftarrow rand_{GF(2^\tau)}()$
12 $\quad L_s.append([r_x, r_y])$
13 **end**
14 $L \leftarrow concat(L_p, L_s)$
15 $[HD]_j \leftarrow permute(L, \pi)$
16 **return** $[HD]_j$;

what is discussed in this section with a formalization of the method to compute a share of a HD using MPC on the matrix $[BT_U]_j$ of Eq. (19). AS receives all shares $[HD]_j$ $\forall j \in [1, N]$ and use them to reconstruct $HD \leftarrow \{[HD]_1, \ldots, [HD]_N\}$. Note that the MPC evaluation in line 5 of Algorithm 1 only involves additions and multiplication by constants. Therefore, Algorithm 1 can be computed locally at RES_j not requiring communication.

4 Conclusion and Future Work

We study secure re-enrollment in cryptographically secured biometrics-based identification and authentication (BIA) systems. We argue that addressing this issue is

paramount for real-life deployments of such systems and ensuring long-term confidentiality of biometrics. We develop a new approach for Secure Non-interactive Users at Scale Enrollment (`SNUSE`) and prototype it. Our experimental results (in Appendices B and C) show a high detection accuracy with over 90% Genuine Acceptance Rate (GAR) and less than 5% False Acceptance Rate (FAR) when the right degree of polynomial is used, see Figs. 6(a) and 6(b); and efficient re-enrollment for thousands of users, e.g., in a few seconds, using standard computing servers (see Appendix C).

There remain several avenues for future work. First, our approach could be expanded to support other FE/FV schemes, e.g., computational and reusable ones. Second, it would be interesting to add support to other biometrics in addition to fingerprints and iris, and other types of devices, e.g., smart-phones. The challenge here is that other biometrics have other notions of matching distance which may require development and optimizations of the secure computation circuits to be practical. Finally, our prototype is in the honest-but-curious model, extending it to handle fully malicious, or at least covert adversaries, without significant overhead is an interesting problem.

Acknowledgement

This work was funded by the US Department of Homeland Security (DHS) Science and Technology(S&T) Directorate under contract no. HSHQDC-16-C-00034. The views and conclusions contained herein are the authors' and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DHS or the US government.

References

1. A. Jain, R. Bolle, and S. Pankanti, *Biometrics: personal identification in networked society*. Springer Science & Business Media, 2006, vol. 479.
2. N. K. Ratha, J. H. Connell, and R. M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM systems Journal*, vol. 40, no. 3, pp. 614–634, 2001.
3. Wikipedia, "Office of Personnel Management data breach," https://en.wikipedia.org/wiki/Office_of_Personnel_Management_data_breach (accessed 2017-12-05).
4. A. Juels and M. Sudan, "A fuzzy vault scheme," *Designs, Codes and Cryptography*, vol. 38, no. 2, pp. 237–257, 2006.
5. B. Fuller, X. Meng, and L. Reyzin, "Computational fuzzy extractors," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2013, pp. 174–193.
6. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. Acm, 2006, pp. 89–98.
7. A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
8. A. C. Yao, "Protocols for secure computations," in *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*. IEEE, 1982, pp. 160–164.
9. O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM, 1987, pp. 218–229.

10. I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Proceedings of the 32Nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*. New York, NY, USA: Springer-Verlag New York, Inc., 2012, pp. 643–662. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32009-5_38
11. T. Rabin and M. Ben-Or, "Verifiable secret sharing and multiparty protocols with honest majority," in *Proceedings of the twenty-first annual ACM symposium on Theory of computing*. ACM, 1989, pp. 73–85.
12. D. Beaver, "Efficient multiparty protocols using circuit randomization." in *Crypto*, vol. 576. Springer, 1991, pp. 420–432.
13. R. Cramer, I. Damgård, and U. Maurer, "General secure multi-party computation from any linear secret-sharing scheme," in *Advances in CryptologyEUROCRYPT 2000*. Springer, 2000, pp. 316–334.
14. I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology–CRYPTO 2012*. Springer, 2012, pp. 643–662.
15. A. Kiayias and M. Yung, "Cryptographic hardness based on the decoding of reed-solomon codes," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2002, pp. 232–243.
16. V. Shoup, "Ntl: A library for doing number theory," *www.shoup.net/ntl/*, 2001.
17. K. Ko, "User's guide to nist biometric image software (nbis)," *NIST Interagency/Internal Report (NISTIR)-7392*, 2007.
18. K. Nandakumar, A. K. Jain, and S. Pankanti, "Fingerprint-based fuzzy vault: Implementation and performance," *IEEE transactions on information forensics and security*, vol. 2, no. 4, pp. 744–757, 2007.
19. B. Tams, "Absolute fingerprint pre-alignment in minutiae-based cryptosystems," in *2013 International Conference of the BIOSIG Special Interest Group (BIOSIG)*, Sept 2013, pp. 1–12.
20. N. Othman, B. Dorizzi, and S. Garcia-Salicetti, "Osiris: An open source iris recognition software," *Pattern Recognition Letters*, vol. 82, pp. 124–131, 2016.
21. Y. J. Lee, K. Bae, S. J. Lee, K. R. Park, and J. Kim, "Biometric key binding: Fuzzy vault based on iris images," in *International Conference on Biometrics*. Springer, 2007, pp. 800–808.
22. K. Eldefrawy, G. Tsudik, A. Francillon, and D. Perito, "SMART: Secure and minimal architecture for (establishing dynamic) root of trust," in *NDSS*, vol. 12, 2012, pp. 1–15.
23. D. Apon, C. Cho, K. Eldefrawy, and J. Katz, "Efficient, reusable fuzzy extractors from lwe," in *International Conference on Cyber Security Cryptography and Machine Learning*. Springer, 2017, pp. 1–18.
24. G. Itkis, V. Chandar, B. W. Fuller, J. P. Campbell, and R. K. Cunningham, "Iris biometric security challenges and possible solutions: For your eyes only?using the iris as a key," *IEEE Signal Processing Magazine*, vol. 32, no. 5, pp. 42–53, Sept 2015.
25. A. Juels and M. Sudan, "A fuzzy vault scheme," *Des. Codes Cryptography*, vol. 38, no. 2, pp. 237–257, Feb. 2006. [Online]. Available: http://dx.doi.org/10.1007/s10623-005-6343-z
26. K. Nandakumar, A. K. Jain, and S. Pankanti, "Fingerprint-based fuzzy vault: Implementation and performance," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 4, pp. 744–757, Dec 2007.

27. Y. Dodis, L. Reyzin, and A. Smith, *Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 523–540. [Online]. Available: https://doi.org/10.1007/978-3-540-24676-3_31
28. X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith, "Secure remote authentication using biometric data," in *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*, ser. EUROCRYPT'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 147–163. [Online]. Available: http://dx.doi.org/10.1007/11426639_9
29. B. Fuller, X. Meng, and L. Reyzin, "Computational fuzzy extractors," in *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, 2013, pp. 174–193.
30. X. Boyen, "Reusable cryptographic fuzzy extractors," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*, ser. CCS '04. New York, NY, USA: ACM, 2004, pp. 82–91. [Online]. Available: http://doi.acm.org/10.1145/1030083.1030096
31. M. Blanton and M. Aliasgari, "On the (non-)reusability of fuzzy sketches and extractors and security in the computational setting," in *Proceedings of the International Conference on Security and Cryptography*, July 2011, pp. 68–77.
32. ——, "Analysis of reusability of secure sketches and fuzzy extractors," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 9, pp. 1433–1445, Sept 2013.
33. D. Apon, C. Cho, K. Eldefrawy, and J. Katz, "Efficient, reusable fuzzy extractors from LWE," in *Cyber Security Cryptography and Machine Learning - First International Conference, CSCML 2017, Beer-Sheva, Israel, June 29-30, 2017, Proceedings*, 2017, pp. 1–18.
34. A. C.-C. Yao, "How to generate and exchange secrets," in *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, ser. SFCS '86. Washington, DC, USA: IEEE Computer Society, 1986, pp. 162–167. [Online]. Available: https://doi.org/10.1109/SFCS.1986.25
35. M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. New York, NY, USA: ACM, 1988, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/62212.62213
36. D. Chaum, C. Crépeau, and I. Damgard, "Multiparty unconditionally secure protocols," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. New York, NY, USA: ACM, 1988, pp. 11–19. [Online]. Available: http://doi.acm.org/10.1145/62212.62214
37. D. Beaver, *Foundations of Secure Interactive Computing.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 377–391. [Online]. Available: https://doi.org/10.1007/3-540-46766-1_31
38. G. Asharov, Y. Lindell, and T. Rabin, *Perfectly-Secure Multiplication for Any $t < n/3$.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 240–258. [Online]. Available: https://doi.org/10.1007/978-3-642-22792-9_14
39. Z. Beerliová-Trubíniová and M. Hirt, *Perfectly-Secure MPC with Linear Communication Complexity.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 213–230. [Online]. Available: https://doi.org/10.1007/978-3-540-78524-8_13

40. R. Canetti, U. Feige, O. Goldreich, and M. Naor, "Adaptively secure multi-party computation," in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96. New York, NY, USA: ACM, 1996, pp. 639–648. [Online]. Available: http://doi.acm.org/10.1145/237814.238015
41. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai, "Universally composable two-party and multi-party secure computation," in *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '02. New York, NY, USA: ACM, 2002, pp. 494–503. [Online]. Available: http://doi.acm.org/10.1145/509907.509980
42. I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, *Practical Covertly Secure MPC for Dishonest Majority – Or: Breaking the SPDZ Limits*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–18. [Online]. Available: https://doi.org/10.1007/978-3-642-40203-6_1
43. D. W. Archer, D. Bogdanov, B. Pinkas, and P. Pullonen, "Maturity and performance of programmable secure computation," *IEEE Security & Privacy*, vol. 14, no. 5, pp. 48–56, 2016. [Online]. Available: https://doi.org/10.1109/MSP.2016.97
44. Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," *J. Cryptol.*, vol. 23, no. 2, pp. 281–343, Apr. 2010. [Online]. Available: http://dx.doi.org/10.1007/s00145-009-9040-7
45. Y. Huang, J. Katz, and D. Evans, "Quid-pro-quo-tocols: Strengthening semi-honest protocols with dual execution," in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, ser. SP '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 272–284. [Online]. Available: http://dx.doi.org/10.1109/SP.2012.43
46. C. Hazay and Y. Lindell, "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries," in *Proceedings of the 5th Conference on Theory of Cryptography*, ser. TCC'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 155–175. [Online]. Available: http://dl.acm.org/citation.cfm?id=1802614.1802628
47. J. Baron, K. Eldefrawy, K. Minkovich, R. Ostrovsky, and E. Tressler, "5PM: Secure pattern matching," in *Proceedings of the 8th International Conference on Security and Cryptography for Networks*, ser. SCN'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 222–240. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32928-9_13
48. ——, "5PM: Secure pattern matching," *J. Comput. Secur.*, vol. 21, no. 5, pp. 601–625, Sep. 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2590628.2590630

Appendix

A SNUSE Implementation

To demonstrate practicality of the SNUSE protocol, we implemented a prototype that works with both fingerprints and iris scan biometrics. In this section we discuss the implementation details of the prototype, including software details and security parameters.

We implemented SNUSE in C++, using the Number Theory Library (NTL) [16]. The polynomials used in these schemes were defined over the Galois Field $GF(2^{24})$, which is large enough to securely encode cryptographic keys while ensuring that the scheme

is computationally efficient. The polynomial degree in the FV is a flexible security parameter that allows adjusting authentication accuracy. The number of data points and chaff points and the distance functions used on FV_{OPEN} are specific to the type of biometric used in the scheme. A SHA-256 hash function is used to compute the secret hash that is included in the HD.

The ideal FV polynomial degree that yields the best accuracy results also depends on the type of biometric used. In Appendix B we present accuracy results as a function of the polynomial degree of the FV for both fingerprints and iris scans.

Secret sharing and MPC were also implemented using polynomials defined over $GF(2^{24})$. Therefore, a BT secret share is defined as a collection of shares of each data point in such BT. The number of RESs is configurable and determines the number of shares generated during the first user enrollment phase. Shares are delivered to RESs through TCP sockets. During re-enrollment, AS sends a request to the RESs, which compute their shares of the helper data and send them back to AS which reconstructs the secret by computing polynomial interpolation using the NTL library. This communication is also implemented using TCP sockets.

A.1 Template Extraction with Fingerprints

The first step in `SNUSE` is to extract the biometric template from the biometric reading. In the case of a fingerprint, the biometric reading is an image of the fingerprint and, as discussed in Section 2, each data point p_i in the biometric template is the position and orientation (x_i, y_i, θ) of a fingerprint *minutiae*. In our fingerprint template extraction we use NIST Biometric Image Software (NBIS) [17] to extract the template data points from the fingerprint image. NBIS returns a set of identified *minutiae* points sorted by the confidence in the identified minutiae. From NBIS output we select the 20 points with the highest confidence and encode as data points in $GF(2^{24})$ and then use them as described in `SNUSE` design. In our prototype, an HD is composed of 20 real data points mixed with 200 chaff points.

During user authentication, in the FV_{OPEN}, candidate *minutiae* points are selected from the helper data based on their distance to minutiae points in the new template BT' sampled from the user during authentication. We use a distance function similar to the one introduced in [18], defined as:

$$D(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + \beta \times \Delta(\theta_i, \theta_j) \tag{21}$$

where $\Delta(\theta_i, \theta_j) = min(|\theta_i - \theta_j|, 360 - |\theta_i - \theta_j|)$. The parameter β allows controlling how much importance is given to the *minutiae* orientation in the distance computation as compared to the euclidean distance between the points. A data point p_i is selected from the helper data if $D(p_i, p_j) < w$ for some point in BT'. As described in [18], the parameters β and w must empirically calibrated to yield the best accuracy results. The fingerprint accuracy experiments reported in Appendix B were conducted with $\beta = 0.2$ and $w = 20$.

To improve accuracy results for noisy fingerprint readings, before extracting the template, during the biometric sampling, we compute the fingerprint pre-alignment algorithm proposed in [19] which works by translating the fingerprint core-point to the center of a coordinate system and then rotating the image according fingerprint orientation patterns. This way, two misaligned fingerprints can still be matched in FV_{OPEN} computation, increasing the accuracy of authentication. Fig. 5(a) illustrates the result

of the template extraction for two misaligned impressions of the same fingerprint before and after pre-alignment. White squares show the minutiae points detected in these fingerprints.

(a) Fingerprint biometric template (b) Iris biometric template

Fig. 5. a) Fingerprint pre-processing and feature extraction. On the top, two impressions of the same fingerprint. At the bottom the correspondent feature extraction after pre-alignment. **b)** Iris feature extraction. On the top, raw iris scans of two different users and at the bottom the resulting iris templates as bitmaps.

A.2 Template Extraction with Iris Scans

In the implementation of the iris version of our prototype, we use the Osiris open-source library [20] for biometric template extraction. Osiris receives as input an iris scan image and outputs an iris template. The resulting templates for two iris scans are depicted in Fig. 5(b). OSIRIS start by pre-processing the eye image, using contour detection and masks to remove the eye contours, eye lashes, and pupil from the scan, leaving only the iris, which is the actual biometric data used for recognition. Next, the ring-shaped image, containing only the iris itself is stretched into rectangle, which is referred to as iris texture. Finally, different filters can be applied to the iris texture each of which resulting in a different iris code: bitmaps formed by black and white pixels. The actual template is a bitmap composed by the combination of multiple iris codes generated using different filter parameters. For instance, the templates shown in Fig. 5(b) are formed by three iris codes vertically concatenated.

Standard iris based authentication works by xor-ing the bitmaps of the stored template with the new template sample provided by the user during authentication. However, such methods are not applicable to fuzzy vaults (recall that the Fuzzy Vault requires a biometric template formed by a set of points in a field). Differently from fingerprints, the iris biometric template is not a set of data points that can be directly used to generate a fuzzy vault. Therefore, the bitmap output must be encoded into multiple data points in such a way that these points can be matched together, by using a given distance function, during FV_{OPEN} computation.

To address this issue we use the following simple encoding mechanism. We divide the biometric template into a grid of 36 equally sized squares. Let $S[x, y]$ denote the square in line x column y of the template grid and $N(S[i, j])$ the number of black pixels in such square. The set of data points that compose the biometric template in this encoding scheme are defined as:

$$BT_U = \bigcup_{i=1}^{6} \bigcup_{i=1}^{3} [N(S[i, 7 - j]), N(S[i, j])] \tag{22}$$

Resulting in 18 data points in the format $p = [x, y]$ where x and y are the number of black pixels in a pair of squares in the template. In our implementation we define the template resolution such that x and y can be represented in 12 bits each and therefore each data point can be encoded as an element of $GF(2^{24})$. these 18 points are mixed with 200 chaff points to generate the helper data. Finally, to implement FV_{OPEN} we use the distance function:

$$D(p_i, p_j) = |x_i - x_j| + |y_i - y_j| \tag{23}$$

which measures the absolute difference in the number of black pixels in both squares that compose each data point. A point is considered a chaff point if such distance is larger than a threshold w. In our iris accuracy experiments in Appendix B we set $w = 100$.

The scheme above is one of multiple possibilities for encoding schemes and distance functions. As our focus in this work is not on biometric template extraction, but on non-interactive user re-enrollment using MPC, we use straight-forward encoding mechanism and distance functions. We refer the interested reader to [21] for more sophisticated methods to encode iris scans into templates that are suitable for usage with FVs. While our simple encoding mechanism reaches modest accuracy results of 80% GAR with 5% FAR, these numbers could reach $> 99\%$ GAR with nearly 0 FAR if an encoding scheme such as the one in [21] is used. It is worth emphasizing that the accuracy relates to the biometric template extraction which is completely orthogonal to `SNUSE` and thus not the focus of the present work.

B Authentication Accuracy

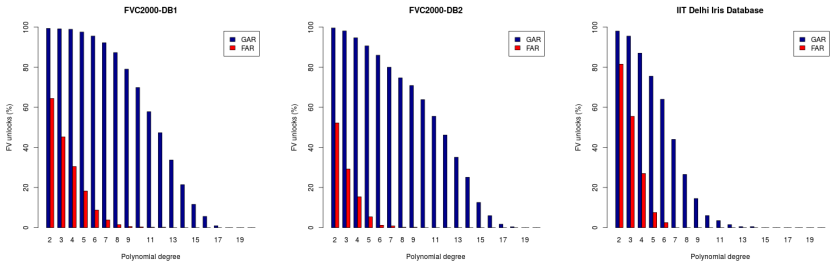
As described above, our prototype supports fingerprints and iris scans. We evaluate the accuracy of these two modes of operation according to the two following metrics:

- **Genuine Acceptance Rate (GAR):** Percentage of rightful users that are successfully authenticated after providing the correct biometric sample.
- **False Acceptance Rate (FAR):** Percentage of users authenticated when providing unauthorized biometric sample.

Our accuracy experiments we conducted using two fingerprint and one iris datasets:

- **FVC2000-DB1:** Fingerprint database containing 10 fingers with 8 impressions each. Samples collected using a low-cost optical sensor[7].

[7] Information and dataset available at: `http://bias.csr.unibo.it/fvc2000/`



(a) Low-cost Optical Sensor (b) Low-cost Capacitive Sensor (c) Iris Scans

Fig. 6. `SNUSE` accuracy: GAR and FAR using fingerprint low-cost optical sensor, fingerprint low-cost capacitive sensor, and iris scans

– **FVC2000-DB2:** Fingerprint database containing 10 fingers with 8 impressions each. Samples collected using a low-cost capacitive sensor[8].
– **IIT Delhi Iris Database (Version 1.0):** Iris scans from 200 eyes with 10 impressions each. Samples collected using JPC1000 digital CMOS camera[9].

For each database we cross-check every possible pair of biometric samples using one of them to compute generate the HD by computing FV_{GEN} and the other to unlock the HD by computing FV_{OPEN}. Therefore, GAR is computed as the fraction of different impressions for the same biometric that are successfully authenticated during FV_{OPEN}. Conversely, FAR is the fraction of biometric samples that succeed in computing FV_{OPEN} in an HD generated using a biometric sample of some other user.

Recall that the polynomial degree in the fuzzy vault algorithm determines the number of data points correctly retrieve from the HD that are necessary to reconstruct the secret. Therefore, a low polynomial degree tends to increase GAR and FAR and a high polynomial degree tends to decrease both. We present the accuracy results for each of the aforementioned databases in Fig. 6, as a function of the polynomial degree. The degree can be calibrated, depending on the application case to yield appropriate accuracy. For example, in the fingerprint databases, a degree of 7 yields a GAR of ∼90% with a FAR of ∼3%. Degree 8 yields ∼80% GAR with 0 FAR. In the iris scan implementation a degree of 5 results in ∼75% GAR with ∼5% FAR.

C `SNUSE` Evaluation

Here we evaluate the computational costs of `SNUSE`. Results in Appendix B justify the choice of polynomial degrees and other parameters used in this evaluation. Such accuracy results show, for example, that for fingerprints, a polynomial with degree 7 yields a GAR of ∼90% with a FAR of ∼3%. A polynomial with degree 8 yields ∼80% GAR with 0 FAR. In our iris scans implementation, a polynomial with degree 5 results

[8] Information and dataset available at: http://bias.csr.unibo.it/fvc2000/
[9] Information and dataset available at: http://www4.comp.polyu.edu.hk/~csajaykr/IITD/Database_Iris.htm

in ∼75% GAR with ∼5% FAR. In the following we proceed with the evaluation of computational costs in terms of time and storage required by SNUSE.

C.1 Computational Requirements

In this section we focus on the computational efficiency of SNUSE. We measure the times required to compute SNUSE protocols with special emphasis to large scale user re-enrollments over a local network. The latter simulates the use case of when an organization wants to re-enroll all its employees/users refreshing their cryptographic keys. The experiments presented throughout this section were executed in an Intel Core i7-3770 octacore CPU @3.40GHz, with 16GB of RAM, running Linux (Ubuntu 14.04LTS). The AS and the RESs were implemented as independent processes communicating though TCP sockets and an artificial delay of 10 milliseconds is introduced in order to simulate a typical communication delay for a local area network. Unless stated otherwise, the fuzzy vault polynomial degree is set to 7 (which has shown to be a reasonable value according to the accuracy experiments in Appendix B) and the number of RESs is set to 5. We use an (N, K) Shamir secret sharing scheme with $N = K$. Therefore, all secret shares are required to reconstruct the secret and, consequently, to compute any function using MPC.

We start the evaluation by measuring the execution times of each of the protocols in SNUSE: Enrollment, Authentication, Re-Enrollment. Table 1 presents the average times and standard deviations (out of 100 independent executions) of each protocol for a single user. The results show that enrollment and authentication phases are considerably more time demanding (in the order of a second) than re-enrollment, which is a consequence of the biometric templates extraction, required in both phases. Since re-enrollment is done based on the secret shares (with pre computed exponentiations) of the biometric template, it does not require extracting the biometric template and only requires one communication round between the AS and the RESs. As a consequence re-enrollment for one user is performed in 13 milliseconds on average.

Protocol	**Avg. Time**	**Std. Dev.**
Enrollment	945.9 ms	24.1 ms
Authentication	848.7 ms	26.2 ms
Re-Enrollment	13.2 ms	0.2 ms

Table 1. Execution times for SNUSE protocols

In our next experiment we evaluate the effect of increasing the polynomial degree on the computational times of each algorithm. We consider reasonable polynomial degrees from 5 to 15 and compute the required time for each case. The results displayed on Fig. 7 show that the variation of polynomial degree has negligible impact on the execution time of the protocols. The only perceptible change is in the authentication protocol, that has a slightly higher execution time with high degrees. FV_{OPEN}, which is computed during authentication, executes multiple polynomial interpolations with the candidate data points selected by comparing the provided biometric template data points with the points in the HD. In this process, subsets of size $d + 1$ (where d is the polynomial degree) are selected from the 20 candidate points, interpolated and

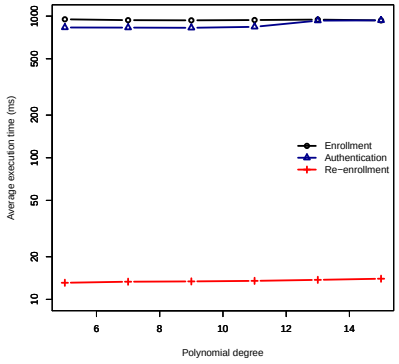


Fig. 7. Execution times as a function of fuzzy vault polynomial degrees

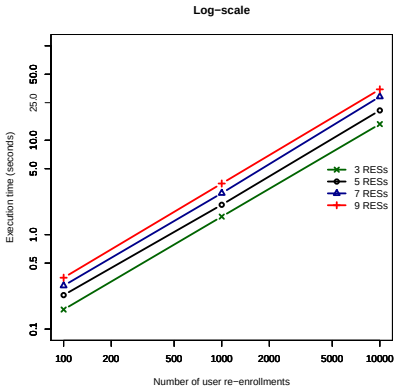


Fig. 8. Processing time for massive user re-enrollment requests for different numbers of RESs

the resulting secret is hashed and compared to the hash in the HD. Increasing d also increases the number of possible combinations, consequently increasing the number of polynomial interpolations and thus the average computation time for authentication. This effect causes a 12% increase in the computational time for a polynomial degree of 15 when compared to a polynomial degree of 5.

Finally, we evaluate how re-enrollment performs at scale, in the order of thousands of user re-enrollments. We run the large scale experiments with 3, 5, 7, and 9 RESs. The results for the large scale experiments are depicted in Fig. 8. We can see that the re-enrollment processing time scales linearly with the number of simultaneous re-enrollment requests. The re-enrollment time for `SNUSE` running with more RESs is also slightly higher than with smaller number of RESs, which is expected due to the higher amount of computation required. Nevertheless, our results indicate that the re-enrollment of massive number of users is affordable. For example, re-enrolling 100 thousand users would using 9 RESs would take less than five minutes and re-enrolling one million users would be possible in less than an hour.

C.2 Storage Requirements

`SNUSE` stores the HD in the AS and the biometric template secret shares in the RESs. The HD public parameters $\Phi = \{F, d, M, H(k)\}$ (recall that $H(k)$ is implemented with a SHA-256 hash function) by themselves require 39 bytes per HD. In our implementation, using $GF(2^{24})$, each element in the HD (data points and chaff points) can be encoded into 6 bytes. Considering an HD with 20 biometric data points and 200 chaff points, and HD for one user requires $220 \times 6 + 39 = 1359$ bytes. Considering that an additional user ID of 30 bytes is associated to each HD entry in the AS database, a massive number of user entries, e.g., 1 million, would require $(1359 + 30) \times 10^6 = 1.39$ GB of storage, which is well within the capacity of modern computers. Increasing the number of users or the number of chaff points used to construct the vault would increase this number linearly. The degree of the polynomial used in the fuzzy vault does not affect the storage requirements on the AS.

Each RES has to store one share of a biometric template per user. Therefore, the storage requirement depends on the size of such shares. In Section 3.4 we discuss three approaches that differ from each other in the number of multiplications required and consequently in the number of network communication rounds. In the first two approaches, each secret share will have the same size of the original biometric template, because no pre-computation of exponentiations takes place before the secret sharing. Therefore, considering a biometric template composed of 20 data points in $GF(2^{24})$, each secret share would require 60 bytes of storage plus 30 bytes for storing the associated user ID. In the case of pre-computing exponentiations before secret sharing, each secret share is in the format of the matrix in Eq. (19) of Section 3.4. In this case, the size of each secret share depends on the polynomial degree used in the fuzzy vault and on the number of data points. If $GF(2^{24})$ is used as the field for the scheme, each element in the secret share matrix can be encoded as 3 bytes. Considering a biometric template with 20 data points and a polynomial of degree 15 (which is a comfortable upper bound, based on the accuracy result we present in Appendix B) /each secret share would require $3 \cdot 20 \cdot 15 = 900$ bytes. With 1 million user this would result in a storage requirement of 900 megabytes per RES.

D Security Analysis

D.1 Confidentiality of Stored Biometrics

In practice, backend servers may store thousands or even hundreds of thousands of BTs. Therefore, an adversary that compromises a backend server which stores biometrics in clear can obtain this massive number of biometrics.

In `SNUSE`, compromising AS does not allow the adversary to reconstruct BT because AS only store HDs, and the impossibility of reconstructing a BT from an HD is guaranteed by security of the underlying FV scheme, which in turn relies on the infeasibility of the polynomial reconstruction problem [15].

On the other hand, the adversary might attempt to retrieve the stored BTs by compromising the RESs. `SNUSE` security, in this case, relies on the security of the secret sharing scheme. In other words, if a (K,N) scheme is used, reconstructing BT in clear implies compromising at least K out of the N RESs. K can be made arbitrarily large according to the desired resilience in a given organization.

D.2 Confidentiality of Biometrics During Execution

The system does not store the BT at any backend server (RES or AS). In addition, the biometric is not reconstructed at the servers during the computation of enrollment, re-enrollment, or during user authentication. The biometric is only visible in clear at the B.T. Reader, during initial enrollment and regular authentication. However, there is no way to work around that, because B.T. Reader is the physical sensor device that samples the biometric. Moreover, since these sensors are simple devices (compared to backend servers), trusted computing techniques such as remote attestation for low-end devices [22] can be efficiently used to ensure that such devices are operating as expected (i.e., not infected by Malware).

If an adversary is able to compromise an RES during the execution of enrollment or re-enrollment protocols, the adversary can retrieve the secret k, because all RESs must know k in order to compute the HD. We argue that this is not as threatening

as reconstructing the BT in clear, because it is relatively easy to revoke and use `SNUSE` to re-assign a fresh secret to a user. However, a user's biometric is usually tied to the individual through its whole life.

Regarding the biometric confidentiality, compromising less than K RESs during the execution of the protocol does not reveal anything about BT. However, compromising at least one RES and the AS, at the same time, during enrollment/re-enrollment execution reveals the BT being computed at that time. This is due to the construction of the fuzzy vault scheme, which allows someone in possession the secret k to tell apart which points in the vault are points in the polynomial encoding k and which ones are chaff points. This limitation can be addressed by using MPC to i) compute a fuzzy vault from randomly generated secret shares $[k]_i$ of the secret k; and ii) compute the permutation π. We consider such implementation as future work. Nevertheless, even in this case, the attack surface is much smaller, because i) it requires compromising both AS plus one RES ii) the attacker must have control of both of them during the protocol computation and iii) even in that case only one BT (the one for which the FV is being computed at that time) is leaked, instead of the whole database.

D.3 Re-Usability of Fuzzy Vaults

Another issue is the re-usability of traditional FVs and FEs. As discussed in [23], the possession of two different HDs generated from the same BT (e.g., a user authenticates with the same biometric in two different organizations), allows an adversary to tell apart chaff points from minutiae points. It would be interesting to implement MPC based re-enrollment with a reusable FE. We defer this direction as future work.

E Related Work

A study [24] of security and privacy challenges facing biometrics, especially iris based ones, investigated suitability and viability of relying on them as the sole method for identification and authentication; the results of the study in terms of accuracy and entropy of extracted keys were both positive and encouraging. The first Fuzzy Vault (FV) scheme was developed in [25]. It was later implemented and tested with actual fingerprint biometrics in [26]. Subsequently, Fuzzy Extractors (FE) were formalized in [27], and further applied to biometrics in [28]. Most FE schemes provided statistical or information-theoretic security, until the scheme of [29] was developed; this computational FE scheme relies on hardness of the Learning with Errors (LWE) problem. Re-usability of FE was first studied in [30], where a reusable FE scheme was first formalized, and then developed for specific attacks. Re-usability enables one to extract multiple helper data from the same biometric without leaking any additional information. Not every FE/FV can be reused and still ensure security as illustrated in subsequent research [31, 32] which analyzed re-usability of multiple FE schemes and demonstrated attacks against them. Finally, new (indistinguishability based) definitions for re-usability were presented [33] and theoretical analysis demonstrated that the computational FE scheme in [29] is not even weakly reusable. In fact, from the helper data, HD1 and HD2, of two instances of the scheme, an attacker can learn the original biometric inputs w_1 and w_2 in their entirety. Fixes to the FE scheme in [29] were then developed by using common public parameters and proven secure in the weak model, and further transformed to ensure string re-usability in the random oracle model.

Secure two/multi-party computation (2/MPC) protocols allow two or more mutually distrusting parties to compute functions of their private inputs, while guaranteeing correctness of the result of the computation, and privacy of inputs and outputs (if desired). 2PC/MPC has been an active area of research for the past three decades [9, 10, 34–42]. Recent models and practical schemes [43] provide a trade off between security and privacy guarantees on one hand, and required computation and communication on the other. For example, the covert model [44] accounts for settings where the involved parties are less likely to cheat if they get caught with a high probability (e.g., 0.5) and the work in [45] proposes protocols in which a malicious adversary may learn a single (arbitrary) bit of additional information about the honest party's input. Generic 2/MPC protocols can be utilized as is in cryptographically secured BIA systems but may incur higher overhead. If performance of generic protocols is unsatisfying, one can design function specific secure protocols for the generate function of FE/FV; several function-specific two and multiparty protocols for pattern matching were also developed in [46–48].