Biometric-Authenticated Searchable Encryption

Daniel Gardham, Mark Manulis and Constantin Cătălin Drăgan

Surrey Centre for Cyber Security
University of Surrey
Guildford, United Kingdom

d.gardham@surrey.ac.uk, mark@manulis.eu, c.dragan@surrey.ac.uk

Abstract. We introduce *Biometric-Authenticated Keyword Search (BAKS)*, a novel searchable encryption scheme that relieves clients from managing cryptographic keys and relies purely on client's biometric data for authenticated outsourcing and retrieval of files indexed by encrypted keywords.

BAKS utilises *distributed trust* across two servers and the *liveness assumption* which models physical presence of the client; in particular, BAKS security is guaranteed even if clients' biometric data, which often has low entropy, becomes public. We formalise two security properties, Authentication and Indistinguishability against Chosen Keyword Attacks, which ensure that only a client with a biometric input sufficiently close to the registered template is considered legitimate and that neither of the two servers involved can learn any information about the encrypted keywords.

Our BAKS construction further supports outsourcing and retrieval of files using multiple keywords and flexible search queries (e.g., conjunction, disjunction and subset-type queries). An additional update mechanism allows clients to replace their registered biometrics without requiring re-encryption of outsourced keywords, which enables smooth user migration across devices supporting different types of biometrics.

Keywords: Searchable Encryption · Biometric Authentication · Secret Sharing.

1 Introduction

Searchable Encryption. *Searchable Encryption* addresses the need for clients with computation or storage limitations to outsource encrypted data to (potentially multiple) servers. Retrieval processes typically operate on encrypted keywords that preserve the privacy of search queries of the client from malicious servers. Most of existing solutions can be generally split into two approaches: *Symmetric Searchable Encryption (SSE)*, e.g., [4, 16, 35], where a high-entropy key is shared between the client and the server, and *Public Key Encryption with Keyword Search (PEKS)*, e.g., [2, 6, 7, 10, 15, 30, 35, 36].

From the practical perspective, both of these approaches require the client to store and manage cryptographic keys, which can be generally considered error-prone and brings further limitations for users who wish to use multiple devices to outsource and perform search over encrypted data. *Password-Authenticated Keyword Search (PAKS)*

[24] was recently introduced to alleviate this problem by basing security of searchable encryption on a human-memorisable password and adopting a two-server architecture [11, 44] to account for dictionary attacks on low-entropy passwords and keywords which arise in such case. The high-level idea used in [24] is to use a password-authenticated key-recovery protocol in combination with an SSE scheme.

Precisely due to their nature of being memorable, passwords are often re-used, in different contexts than their initial purpose [34]. Moreover, strict policies that require the user to frequently update his password, can further exacerbate this problem. A security breach on a particular server may impact all other servers for which the user has re-used some form of his password. The current trend, observed in the domain of web authentication (e.g. [20]), is to move away from passwords to other forms of usable authentication factors. In this context, biometric data (e.g., fingerprints, iris recognition, facial imagery, etc.) can be seen as an alternative to passwords, especially since support for different types of biometrics on commodity user devices such as laptops, smartphones, and wearables is on the rise.

Challenges for Biometric-based Searchable Encryption. Crucially, biometric data used for authentication is inherently noisy (e.g., as opposed to passwords), which introduces unique challenges for the design and security analysis of biometric-based protocols. In the context of searchable encryption, a general approach for a purely biometric-based solution could be to adopt fuzzy extractors (FE) [8,12,18,19] and combine them with SSE schemes [4, 16, 35], i.e., user's biometric input would be processed by a fuzzy extractor to obtain a symmetric key, which will then be used to outsource encrypted keywords to the server and perform the search. We insist on requiring the fuzzy extractor to be *reusable*, that is, the biometric can be used to source keys in other cryptographic schemes. Whilst traditional notions of FE necessarily allow for sourcing more bits of entropy [22], they do not allow the same biometric to be used elsewhere without potentially compromising security, which is impractical in the modern world. However, to practically use reusable fuzzy extractors to source high-entropy (symmetric) keys, the biometric data needs to be of a certain "quality" (i.e., the min-entropy greater than a fixed threshold). Techniques to utilize distributions with low-entropy have been studied in [12], but require some additional structure, in particular, that random subsequences of the biometric data source still have sufficient min-entropy. It is shown that this property is necessary to support low-entropy source material [12]. Finding biometric data that satisfies these requirements is particularly challenging, as even the best source of biometric data, i.e., iris recognition [39], ran by the state-of-the-art iris recognition protocol IrisCode [17] falls short of the minimum bound when processed by a (reusable) fuzzy extractor. The resulting key would have approximately 32 bits security [22]. Additionally, it is unclear how to apply the techniques in [12] to arbitrary types of biometric data, and enforce the structure required by low entropy source distributions. Therefore, the use of fuzzy extractors imposes restrictions over the selection of biometric data, that makes them unsuitable for our construction.

Another aspect for the unsuitability of fuzzy-extractors for our model relates to their security assumptions. The security of this general approach would rely on the secrecy of the biometric data, and if this data is leaked the server would be able to break the privacy of the outsourced keywords. However, the assumption that biometric data is secret, although widely used in academia [9,18,27], is often too strong for the real world. Indeed, in many practical applications biometric authentication often relies on mechanisms ensuring physical presence of the user, e.g., in applications of e-passports and in recent FIDO standards for web authentication [20], etc. This so-called *liveness assumption* is also part of industrial standardization efforts, e.g., ISO/IEC WD 30107 [1], to detect and prevent impersonation attacks based on digital copies of biometric data or other fake artefacts, and has been also considered in the academic literature [21, 38]. Under this assumption biometric data used as input to the cryptographic protocol can be considered public as long as it remains fresh and stems from a living subject, in which case security of the protocol can still be guaranteed. The liveness assumption, however, is only valid and enforced on the client side, and does not prevent malicious servers from running brute-force attacks against the biometric template, thus enabling dictionary-type attacks on the outsourced (low-entropy) keywords [15]. Hence, designing a searchable encryption scheme that would rely only on biometric data and remain secure is particularly challenging and requires a new approach.

We note that access pattern attacks, when paired with auxiliary information about the files in the database, could allow an adversary to recover the keyword of a search query (e.g. [25]). A variety of works [13, 28, 33, 45] have shown that it may also be impossible to protect against access pattern leakage attacks, and that SSE as a primitive leaks too much information to feasibly mitigate against these attacks. Some of these known attacks [13, 45] require an adversary to perform active insertion of files, which could be mitigated against by enforcing an authentication property for at least the outsourcing protocol. We briefly note that oblivious RAM (ORAM), or other techniques [31], can be used to achieve minimal information leakage, that is, the server only learns the number of files in the database. However, ORAM typically requires high bandwidth, but where low bandwidth is sufficient, much more client storage is needed [43]. With this in mind, understanding when pattern leakage is acceptable is an important direction for future research, but is not considered further in this work.

Our Contribution: Biometric-Authenticated Keyword Search (BAKS). We propose a novel searchable encryption scheme, BAKS, that relies purely on a client's biometric data for authenticated outsourcing and retrieval of files indexed by encrypted keywords and relieves clients from managing cryptographic keys. The scheme relies on the two-server architecture and the liveness assumption on the biometric inputs. BAKS makes use of symmetric searchable encryption techniques with the symmetric key being linked to the biometric data and protected by secret-sharing techniques involving both servers. BAKS accounts for the noise in biometric measurements and consists of a protocol that allows the client to reconstruct this key following an interaction with both servers as long as the measured biometric input is "similar" to the template that was used during the initial registration phase. The degree of similarity, potentially influenced by the accuracy of the measuring device, is defined by the client during the registration. The use of a two-server architecture not only is crucial for the secrecy of the keywords but also improves the availability of outsourced files, since each server stores a copy. Our security model for BAKS defines two main security properties: (i) indistinguishability against chosen keyword attacks guaranteeing that keywords remain secret against an active adversary in control of at most one server and (ii) authentication ensuring that only legitimate clients can outsource the encrypted keywords. Security of our construction is proven under standard cryptographic assumptions and its efficiency analysis shows that the scheme has linear overhead in the length of the biometric template. In addition, BAKS supports update of the registered biometric without requiring any re-encryption of the outsourced keywords, thus making it easier for the users to migrate across devices.

2 Preliminaries and Building Blocks

We introduce the assumptions and building blocks underlying our BAKS scheme.

2.1 Cryptographic Building Blocks

Pedersen Commitment [37]. Let $\mathbb{G}$ be a multiplicative cyclic group of order q, with $g, h \leftarrow_{\$} \mathbb{G}$ two generators such that $\log_g h$ is unknown. A Pedersen commitment is computed as $c \leftarrow g^r h^m$, for some message m, with $r \leftarrow_{\$} \mathbb{Z}_q$. It can be opened by providing (r, m). Pedersen commitments provide perfect hiding and computational binding of the message under the Discrete Logarithm (DL) assumption in $\mathbb{G}$.

Pseudorandom Function [23, 32]. A pseudorandom function ($\mathtt{PRF}$) takes as input a high entropy key k and a message m, and produces an output $\mathtt{PRF}(k, m)$ that should be indistinguishable from a uniformly random bit string of the same length, for any efficient adversary $\mathcal{A}$. In this paper, we restrict the key space, message space, and output to be a cyclic group $\mathbb{G}$ of order q, $\mathtt{PRF} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}$.

Key Derivation Function [29]. A key derivation function ($\mathtt{KDF}$) takes as input a source of key material σ and a context variable c, to produce a key k. Any efficient adversary $\mathcal{A}$ can distinguish with a negligible probability between the output of $\mathtt{KDF}(\sigma, c)$ and a uniformly sampled binary string of the same length, for some σ, c. Let $\mathbb{G}$ be a cyclic group of order q, we consider $\mathtt{KDF} : \mathbb{G} \times \{0, 1\}^* \rightarrow \mathbb{G}$.

Message Authentication Code [5]. A message authentication code ($\mathtt{MAC}$) is defined as $(\mathtt{KGen}, \mathtt{Tag}, \mathtt{Ver})$, where the secret key $\mathtt{mk} \leftarrow \mathtt{KGen}(1^\lambda)$, for some security parameter λ. Algorithm $\mathtt{Tag}(\mathtt{mk}, m)$ outputs a code (or a tag) μ, for any key $\mathtt{mk}$ and any message m. The verification algorithm $\mathtt{Ver}(\mathtt{mk}, m, \mu)$ evaluates if the tag μ is valid w.r.t. $\mathtt{mk}$ and a message m. The $\mathtt{MAC}$ satisfies a *correctness* property: $\mathtt{true} \leftarrow \mathtt{Ver}(\mathtt{mk}, m, \mathtt{Tag}(\mathtt{mk}, m))$, for some m and any $\mathtt{mk} \leftarrow \mathtt{KGen}(1^\lambda)$. The $\mathtt{MAC}$ is *unforgeable* if any efficient adversary $\mathcal{A}$ has a negligible advantage to create a tag μ^* for a message m^*, without access to the key $\mathtt{mk}$. The adversary has access to an oracle $\mathcal{O}_{\text{tag}}(\cdot)$ that outputs $\mu \leftarrow \mathtt{Tag}(\mathtt{mk}, m)$ for any message $m \neq m^*$. In this paper, we use the output of $\mathtt{KDF}$ as the key space, and consider messages of arbitrary length $\{0, 1\}^*$. There is no restriction on the code space, but for uniformity with the previous cryptographic primitives we take it as $\mathbb{G}$.

Secret Sharing of Group Elements [37, 40, 41]. We consider a threshold secret-sharing scheme $\mathtt{SS} = (\mathtt{Setup}, \mathtt{Shr}, \mathtt{Rec})$ that shares group elements in $\mathbb{G}$ for which we know the discrete logarithm. The setup algorithm $\mathtt{Setup}(1^\lambda)$ sets the public parameters $\mathtt{pp} = (t, N, g, \mathbb{G}, q)$, with N as the number of shares, $0 < t \leq N$ as the threshold, the group $\mathbb{G}$ of order q as the secret space and share space, and $g \in \mathbb{G}$ as the generator. Algorithm $\mathtt{Shr}(\mathtt{pp}, k)$ shares the secret $K = g^k$ by returning the shares $\{K_i\}_{i=1}^N$. This is done by first applying Shamir's threshold secret sharing scheme [41] to create the shares $k_1, \ldots, k_N \in \mathbb{Z}_q$ from k, before returning $\{K_i\}_{i=1}^N$ with $K_i = g^{k_i}$. The reconstruction $\mathtt{Rec}(\mathtt{pp}, T)$ returns K for any set of shares $|T| \geq t$, by replicating Shamir's reconstruction in the exponent [37]. The scheme $\mathtt{SS}$ is *private*, if any efficient adversary $\mathcal{A}$ has an negligible advantage to distinguish between given shares of $K \in \mathbb{G}$ or shares of $K' \in \mathbb{G}$, when he can see at most $(t-1)$ shares [40], for any $\mathsf{pp} = (t, N, g, \mathbb{G}, q)$. This scheme trivially satisfies privacy, as the underlying Shamir scheme is perfect [41].

2.2 Biometric Sampling and Liveness Assumption

A user's biometric data is modelled as a distribution $\mathcal{D}$, and an instance of a user submitting a biometric reading is captured by sampling $\mathcal{W} \leftarrow_{\$} \mathcal{D}$. Let M be a metric space with distance function $d : M \times M \to \mathbb{R}^{+}$. Our protocol is constructed for d being the Hamming distance over $M := \{0,1\}^N$, however, we note that the *model* fits a generic instance to encompass other metrics suitable for different types of biometric data. To relate the biometric sampling to the security properties, it is necessary to define two error-probabilities.

The first error we consider is *false rejection.* That is, the distance between any two samples from the same biometric distribution is bounded by a constant τ_1 with probability $1 - \varepsilon_{fr}$. Here we capture the event a legitimate user submits a biometric sample that is sufficiently noisy and is therefore rejected. This probability will be necessary in the definition for the correctness of the protocol. Formally, we have:

$$\Pr[\mathcal{W} \leftarrow \mathcal{D}, \mathcal{W}' \leftarrow \mathcal{D} : d(\mathcal{W}, \mathcal{W}') \leq \tau_1] \geq 1 - \varepsilon_{fr}$$

Secondly, we introduce *false acceptance.* This states that any two biometrics sampled from two different distributions (which corresponds to different users submitting biometric data) have a minimum distance τ_2 with probability $1 - \varepsilon_{fa}$. We use this to model an adversary, who cannot sample from the user's biometric distribution, that is able to falsely authenticate himself as that user. We define false acceptance formally as:

$$\Pr[\mathcal{W} \leftarrow \mathcal{D}, \mathcal{W}' \leftarrow \mathcal{D}' : d(\mathcal{W}, \mathcal{W}') > \tau_2] \geq 1 - \varepsilon_{fa}$$

Finally, we assume that the biometric distribution $\mathcal{D}$ for user $\mathtt{U}$ is public and consequently rely on the *liveness assumption* [1,17] which states that any input of biometric data is sampled fresh from a user. It ensures the physical presence of the user and thus prevents an attacker from mounting replay attacks, or altering the output of the biometric sensor. We model it through an oracle $\mathcal{O}_{\text{bio}}$ that takes as input some biometric distribution $\mathcal{D}$, chosen by the adversary, and outputs results of the computation using a fresh sample $\mathcal{W} \leftarrow_{\$} \mathcal{D}$. Any step that involves computation on the biometric sample directly must be computed and output by the oracle.

An implementation of liveness assumption in practice would require some form of trusted processing of biometric measurements, which is not in the focus of this paper. We note, however, there exist a number of methods for enforcing liveness detection such as *software enhancement* (e.g. pupil and eye movement for iris recognition), *hardware enhancement* (e.g. temperature sensing, pulse, electrical conductivity, ECG for fingerprints) and *challenge-response techniques* (e.g. expressions in face recognition) [3].

3 Biometric-Authenticated Keyword Search: Syntax and Definitions

In this section we model BAKS and its security properties. Our model is inspired by the recent model for password-authenticated keyword search from [24], which in turn addresses main security requirements that have been previously formulated for other flavours of searchable encryption, e.g. [2, 6, 15]. The main differences to [24] is that we need to account for the inherent errors in the imperfectness of the measurement process, and also application of the liveness assumption to this protocol.

3.1 Syntax of BAKS

Definition 1 (BAKS). *The* $\mathtt{BAKS} = (\mathtt{Setup}, \mathtt{Register}, \mathtt{Outsource}, \mathtt{Retrieve})$ *protocol consists of the following algorithms:*

• $\mathtt{Setup}(1^\lambda) : \mathtt{pp}$, takes as input a security parameter λ and outputs public parameters $\mathtt{pp}$, that include a description for the structure of the biometric data.
• $\mathtt{Register}(\mathtt{pp}, \mathtt{U}, t, \mathcal{W}, \mathtt{S}_0, \mathtt{S}_1)$ is executed between user $\mathtt{U}$ and two servers $\mathtt{S}_0$ and $\mathtt{S}_1$, and consists of the following two interactive algorithms:

- $\mathtt{RegisterU}(\mathtt{pp}, t, \mathcal{W}, \mathtt{S}_0, \mathtt{S}_1) : \{\mathtt{succ}, \mathtt{fail}\}$ performed by $\mathtt{U}$, takes an input a description of the biometric data $\mathcal{W}$, public parameters $\mathtt{pp}$, and the identities of the two servers $\mathtt{S}_0$ and $\mathtt{S}_1$. It interacts with $\mathtt{RegisterS}_d$ for $d \in \{0, 1\}$ and outputs $\mathtt{succ}$ if registration was successful and $\mathtt{fail}$ otherwise.
- $\mathtt{RegisterS}_d(\mathtt{pp}, \mathtt{U}, \mathtt{S}_{1\text{-}d}) : \mathtt{info}_d$, performed by $\mathtt{S}_d$, $d \in \{0, 1\}$, takes as input the users identity $\mathtt{U}$ and the identity of the other server $\mathtt{S}_{1\text{-}d}$. It interacts with $\mathtt{RegisterU}$ and $\mathtt{RegisterS}_{1-d}$. At the end of the protocol, it stores $\mathtt{info}_d$ associated with user $\mathtt{U}$ on $\mathtt{S}_d$.

• $\mathtt{Outsource}(\mathtt{pp}, \mathtt{U}, \mathcal{W}', w, \mathtt{f}, \mathtt{S}_0, \mathtt{info}_0, \mathtt{S}_1, \mathtt{info}_1)$ is executed between a user $\mathtt{U}$ and two servers $\mathtt{S}_0$ and $\mathtt{S}_1$ that follow the interactive algorithms:

- $\mathtt{OutsourceU}(\mathtt{pp}, \mathtt{U}, \mathcal{W}', w, \mathtt{f}, \mathtt{S}_0, \mathtt{S}_1) : \{\mathtt{succ}, \mathtt{fail}\}$ executed by $\mathtt{U}$, takes as input a sampled biometric $\mathcal{W}'$, a keyword w, and a file descriptor $\mathtt{f}$, and finally the identity of two servers $\mathtt{S}_0$ and $\mathtt{S}_1$. The algorithm interacts with $\mathtt{OutsourceS}_d$ for $d \in \{0, 1\}$, it outputs $\mathtt{succ}$ if successful, and $\mathtt{fail}$ otherwise.
- $\mathtt{OutsourceS}_d(\mathtt{pp}, \mathtt{U}, \mathtt{S}_{1\text{-}d}, \mathtt{info}_d) : (C, \mathtt{f})$ is performed by server $\mathtt{S}_d$ for $d \in \{0, 1\}$. It takes as input the identity of the user $\mathtt{U}$, identity of the other server $\mathtt{S}_{1\text{-}d}$ and information $\mathtt{info}_d$. This protocol interacts with $\mathtt{OutsourceU}$ and $\mathtt{OutsourceS}_{1-d}$, and upon completion, stores $(C, \mathtt{f})$ in a database $\boldsymbol{C}_d$.

• $\mathtt{Retrieve}(\mathtt{pp}, \mathtt{U}, \mathcal{W}', w, \mathtt{S}_0, \mathtt{info}_0, \mathtt{S}_1, \mathtt{info}_1)$ is executed between user $\mathtt{U}$ and the servers $\mathtt{S}_0$ and $\mathtt{S}_1$ that follow the two interactive algorithms below.

- $\mathtt{RetrieveU}(\mathtt{pp}, \mathcal{W}', w, \mathtt{S}_0, \mathtt{S}_1) : \boldsymbol{F}$, executed by the user $\mathtt{U}$, takes as input a sampled biometric $\mathcal{W}'$, a keyword w, and the identity of the servers $\mathtt{S}_0$ and $\mathtt{S}_1$. It interacts with $\mathtt{RetrieveS}_d$ for $d \in \{0, 1\}$. It outputs a set $\boldsymbol{F}$ containing all file descriptors $\mathtt{f}$ associated with keyword w.
- $\mathtt{RetrieveS}_d(\mathtt{pp}, \mathtt{U}, \mathtt{S}_{1\text{-}d}, \mathtt{info}_d) : \{\mathtt{succ}, \mathtt{fail}\}$, executed by server $\mathtt{S}_d$ for $d \in \{0, 1\}$. It takes as input user identity $\mathtt{U}$, server identity $\mathtt{S}_{1\text{-}d}$ and information $\mathtt{info}_d$, it interacts with $\mathtt{RetrieveU}$ and $\mathtt{RetrieveS}_{1-d}$, and outputs a flag in $\{\mathtt{succ}, \mathtt{fail}\}$.

We store the outsourced data on both servers for redundancy and increased availability. In particular, a user suffers minimal data loss in the event a server is compromised and mounts a denial of service attack.

Correctness. Intuitively, correctness ensures that a file $\mathtt{f} \in \mathcal{F}$ outsourced under a keyword w will be retrieved (i.e., $\mathtt{f} \in \boldsymbol{F}$) with probability of $1 - \varepsilon_{fr}$, where ε_{fr} is the probability of false rejection, provided the user presents a biometric sample $\mathcal{W}'$ which is sufficiently close to the registered template $\mathcal{W}$. Formally, we say that the BAKS scheme is *correct* if $\forall \lambda \in \mathbb{N}, \mathtt{f} \in \mathcal{F}, w \in wd, \mathcal{W}, \mathcal{W}' \in \mathcal{D}, \mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda)$ we get $\Pr[\mathtt{f} \in \boldsymbol{F}] = 1 - \varepsilon_{fr}$ iff:

$$\begin{aligned}\langle \mathtt{succ}, \mathtt{info}_0, \mathtt{info}_1 \rangle &\leftarrow \mathtt{Register}(\mathtt{pp}, \mathtt{U}, t, \mathcal{W}, \mathtt{S}_0, \mathtt{S}_1)\\ \langle \mathtt{succ}, (C, \mathtt{f}), (C, \mathtt{f}) \rangle &\leftarrow \mathtt{Outsource}(\mathtt{pp}, \mathtt{U}, \mathcal{W}', w, \mathtt{f}, \mathtt{S}_0, \mathtt{info}_0, \mathtt{S}_1, \mathtt{info}_1)\\ \langle \boldsymbol{F}, \mathtt{succ}, \mathtt{succ} \rangle &\leftarrow \mathtt{Retrieve}(\mathtt{pp}, \mathtt{U}, \mathcal{W}', w, \mathtt{S}_0, \mathtt{info}_0, \mathtt{S}_1, \mathtt{info}_1)\end{aligned}$$

3.2 Security Definitions

We define two notions of security for BAKS, *Indistinguishability against Chosen Keyword Attacks* (`IND-CKA`) and *Authentication* (`Auth`), inspired by the recent model from [24]. We model the adversary $\mathcal{A}$ as a probabilistic polynomial-time algorithm (PPT), and we define security through experiments in Figure 1.

Oracles. We consider a PPT adversary $\mathcal{A}$ that interacts with the BAKS functionality through the following set of oracles and can possibly corrupt one of the two servers, $\mathtt{S}_0$ or $\mathtt{S}_1$, involved in the protocol. During each user registration, the adversary gets access to the information assigned to server $\mathtt{S}_{1\text{-}d}$, for any one d of his choice. Then, the adversary can *only* play an active role in the retrieval and outsourcing protocols by assuming the role of server $\mathtt{S}_{1\text{-}d}$, for d fixed during registration. We further give the adversary the capability to impersonate the user, together with his control over server $\mathtt{S}_{1\text{-}d}$, and interact with an honest server $\mathtt{S}_d$ either during retrieval or outsourcing.

To manage multiple registration sessions, even for the same user (with potentially different biometric distributions), we use a unique session identifier j. For simplicity, this session identifier starts at 0, and with each registration it is incremented. Each time the adversary wants to retrieve or outsource files for a particular user, he needs to provide the corresponding session identifier i, s.t. $0 \leq i < j$. Due to the *liveness assumption* each oracle call that handles (adversarial given) biometric distributions must first call the biometric sampling oracle to extract a biometric sample.

Internal to the oracles, we use a table E (initially empty) to store and access the tuples $(d, \mathcal{D}, \mathtt{info}_d)$ assigned to the honest servers $\mathtt{S}_d$ during registration, i.e. $E[j] \leftarrow (d, \mathcal{D}, \mathtt{info}_d)$. We store all biometric distributions for which the adversary has requested a sample in a list B; that is initially empty. We also log all keyword requests by the adversary in the sets `ASet` (during outsourcing) and `ISet` (during retrieval); that are initialised at the beginning of both security experiments. The variables $i^* \in \mathbb{Z}, \mathtt{f}^* \in \boldsymbol{F}$ and user distribution $\mathcal{D}^*$ are used to store the challenge values during the challenge outsource oracle in the `IND-CKA` experiment.

– $\mathcal{O}_{\mathrm{ch}}(b, \cdot)$ is a challenge oracle, which on input $(\cdot) = (i, w_0, w_1, \mathtt{f})$ aborts if $((i^* \geq 0) \vee (i \geq j) \vee ((i, w_0) \in \mathtt{ISet}) \vee ((i, w_1) \in \mathtt{ISet}))$. Otherwise, it sets $i^* \leftarrow i$, $\mathtt{f}^* \leftarrow \mathtt{f}$, and takes $\mathcal{D}^* \leftarrow \mathcal{D}$ from $E[i] \leftarrow (d, \mathcal{D}, \mathtt{info}_d)$. Then, it invokes oracle $\mathcal{O}_{\mathrm{outU}}(i^*, w_b, \mathtt{f}^*)$. It is used as the indistinguishably challenge in the `IND-CKA` proof for `BAKS`, where bit b is defined.

- $\mathcal{O}_{\text{reg}}(\cdot)$, with $(\cdot) = (d, \mathcal{D})$, is the oracle that registers user $\mathtt{U}$ and its biometric distribution $\mathcal{D}$. First, a biometric sample $\mathcal{W}$ is obtained from the $\mathcal{O}_{\text{bio}}(\mathcal{D})$. The registration protocol $\mathtt{Register}$ is run between the adversary $\mathcal{A}$ playing the role of server $\mathtt{S}_{1\text{-}d}$, and the oracle running $\mathtt{RegisterU}$ and $\mathtt{RegisterS}_d$. Then, the map E gets updated for the current session identifier j as $E[j] \leftarrow (d, \mathcal{D}, \mathtt{info}_d)$, and j is incremented $j \leftarrow j + 1$.
- $\mathcal{O}_{\text{outU}}(\cdot)$ on input $(\cdot) = (i, w, \mathtt{f})$ the oracle aborts if $(i \geq j)$, otherwise it obtains $(d, \mathcal{D}, \mathtt{info}_d) \leftarrow E[i]$. The biometric sample is taken from $\mathcal{W} \leftarrow \mathcal{O}_{\text{bio}}(\mathcal{D})$. Then, the protocol $\mathtt{Outsource}$ is then executed, with the oracle running $\mathtt{OutsourceU}$ and $\mathtt{OutsourceS}_d$, and the adversary $\mathcal{A}$ playing as $\mathtt{S}_{1\text{-}d}$. During the authentication game, the oracle performs an additional step $\mathtt{ASet} \leftarrow \mathtt{ASet} \cup (i, w, \mathtt{f})$.
- $\mathcal{O}_{\text{outS}}(\cdot)$ on input $(\cdot) = (i)$. If $i < j$ the oracle parses list E and obtains $(d, \mathcal{D}, \mathtt{info}_d)$; otherwise it aborts. Then, the protocol $\mathtt{Outsource}$ is executed with $\mathcal{A}$ playing the roles of $\mathtt{S}_{1\text{-}d}$ and $\mathtt{U}$, and the oracle running $\mathtt{OutsourceS}_d$. This oracle is used in the $\mathtt{AUTH}$ experiment for $\mathtt{BAKS}$.
- $\mathcal{O}_{\text{retU}}(\cdot)$ on input $(\cdot) = (i, w)$. If $i \geq j$ and $((i = i^*) \wedge (w \in \{w_0, w_1\}))$ the oracle aborts; otherwise it parses list E and obtains $(d, \mathcal{D}, \mathtt{info}_d)$. A biometric sample is taken from $\mathcal{W} \leftarrow \mathcal{O}_{\text{bio}}(\mathcal{D})$. Then, the protocol $\mathtt{Retrieve}$ is then executed with the adversary $\mathcal{A}$ in role of $\mathtt{S}_{1\text{-}d}$, and the oracle honestly running $\mathtt{RetrieveU}$ and $\mathtt{RetrieveS}_d$. If $(i^* = -1)$ then the oracle further computes $\mathtt{ISet} \leftarrow \mathtt{ISet} \cup (i, w)$.
- $\mathcal{O}_{\text{retS}}(\cdot)$ takes as input $(\cdot) = (i)$, and aborts only if $i \geq j$. Otherwise, it obtains $(d, \mathcal{D}, \mathtt{info}_d) \leftarrow E[i]$. The retrieve protocol $\mathtt{Retrieve}$ is then executed, with the oracle running $\mathtt{RetrieveS}_d$ honestly, and the adversary $\mathcal{A}$ playing the roles of user $\mathtt{U}$ and server $\mathtt{S}_{1\text{-}d}$. This oracle is used in the $\mathtt{IND\text{-}CKA}$ property of $\mathtt{BAKS}$.
- $\mathcal{O}_{\text{bio}}(\cdot)$ takes as input a biometric distribution $(\cdot) = (\mathcal{D})$. It samples $\mathcal{W} \leftarrow_{\$} \mathcal{D}$, possibly performs computations on $\mathcal{W}$ and outputs the result. Additionally, it updates the list of all queried biometric distributions $B \leftarrow B \cup \{\mathcal{D}\}$.

Indistinguishability against Chosen Keyword Attacks ($\mathtt{IND\text{-}CKA}$). This security property is closely related to [4] and [24], with the extension that our definition considers authentication with regard to the retrieval phase based on a user's biometric data. It is formally defined through the experiment $\mathbf{Exp}^{\mathtt{IND\text{-}CKA}-b}_{\mathtt{BAKS},\mathcal{A}}$ in Figure 1. The experiment is initialised and the public parameters $\mathtt{pp}$ are set, the adversary is given access to the oracles $\mathcal{O}_{\text{ch}}$, $\mathcal{O}_{\text{reg}}$, $\mathcal{O}_{\text{out}}$,$\mathcal{O}_{\text{retU}}$, $\mathcal{O}_{\text{retS}}$ and $\mathcal{O}_{\text{bio}}$ where it makes $1, q_r, q_o, q_t, q_s$ and q_b queries respectively. The adversary wins the game if, when presented with a challenge, is unable to distinguish which keyword w_b was used in the execution of the protocol except in the case an (illegitimate) biometric sample is falsely accepted. We capture the two scenarios for $\mathcal{A}$ in the $\mathtt{IND\text{-}CKA}$ experiment. Firstly, an adversary who has control of a corrupt server $\mathtt{S}_{1\text{-}d}$ interacts with the honest user $\mathtt{U}$ and server $\mathtt{S}_d$, or secondly, where the adversary controls an illegitimate user communicating with the honest servers $\mathtt{S}_d$ and $\mathtt{S}_{1\text{-}d}$. We must also consider the case $\mathcal{A}$ samples a biometric sufficiently close to the registered template of a user. This is incorporated by utilising the false acceptance probability ε_{fa} defined in Section 2.2. We say that a $\mathtt{BAKS}$ scheme is $\mathtt{IND\text{-}CKA}$-secure if the following advantage is bounded by $\varepsilon_{fa} + \mathtt{negl}$, where $\mathtt{negl}$ is negligible in λ:

$$\mathbf{Adv}^{\mathtt{IND-CKA}}_{\mathtt{BAKS},\mathcal{A}}(1^\lambda) := \left|\Pr[\mathbf{Exp}^{\mathtt{IND\text{-}CKA}-1}_{\mathtt{BAKS},\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathtt{IND\text{-}CKA}-0}_{\mathtt{BAKS},\mathcal{A}}(1^\lambda) = 1]\right|$$

Authentication ($\mathtt{AUTH}$). Motivated by a similar property in the password-based setting [24], the authentication property of $\mathtt{BAKS}$ captures two attack scenarios for an

```
$\mathbf{Exp}^{\mathtt{AUTH}}_{\mathtt{BAKS},\mathcal{A}}$
$E \leftarrow \emptyset; B \leftarrow \emptyset; j \leftarrow 0;$
$\mathtt{ASet} \leftarrow \emptyset; \mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda);$
$(i^*, w^*, \mathtt{f}^*, \mathcal{D}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{reg}},\mathcal{O}_{\text{outU}},\mathcal{O}_{\text{outS}},\mathcal{O}_{\text{retU}},\mathcal{O}_{\text{bio}}}(\mathtt{pp})$
$\mathcal{W}' \leftarrow_{\$} \mathcal{D}^*$
$\langle \boldsymbol{F}, \mathtt{succ}, \mathtt{succ} \rangle \leftarrow \mathtt{Retrieve}(\mathtt{pp}, \mathtt{U}, \mathcal{W}', w^*, \hat{S})$
return $(((i^*, w^*, \mathtt{f}^*) \notin \mathtt{ASet}) \wedge$
        $(\mathtt{f}^* \in \boldsymbol{F}) \wedge (\mathcal{D}^* \notin B))$
```

```
$\mathbf{Exp}^{\mathtt{IND\text{-}CKA}-b}_{\mathtt{BAKS},\mathcal{A}}$
$E \leftarrow \emptyset; B \leftarrow \emptyset; i^* \leftarrow (-1); j \leftarrow 0;$
$\mathtt{ISet} \leftarrow \emptyset; \mathtt{pp} \leftarrow \mathtt{Setup}(1^\lambda);$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{ch}},\mathcal{O}_{\text{reg}},\mathcal{O}_{\text{outU}},\mathcal{O}_{\text{retU}},\mathcal{O}_{\text{retS}},\mathcal{O}_{\text{bio}}}(\mathtt{pp})$
return $b' \wedge (\mathcal{D}^* \notin B)$
```

Fig. 1. Security Experiments for BAKS, where $\hat{S} = (\mathtt{S}_0, \mathtt{info}_0, \mathtt{S}_1, \mathtt{info}_1)$.

adversary. In the first case, $\mathcal{A}$ wins if he is able to outsource a file on behalf of some user without sampling the user's biometric. Secondly, if it is able to retrieve an honestly outsourced, file again without sampling a biometric from the user's biometric distribution. We define this property with the experiment $\mathbf{Exp}^{\mathtt{AUTH}}_{\mathtt{BAKS},\mathcal{A}}$ in Figure 1. In the game, variables are initialised, and the adversary is given access to the oracles $\mathcal{O}_{\text{reg}}$, $\mathcal{O}_{\text{outU}}$, $\mathcal{O}_{\text{outS}}$, $\mathcal{O}_{\text{retU}}$ and $\mathcal{O}_{\text{bio}}$ where each oracle is queried q_r, q_o, q_s, q_t and q_b times, respectively. The adversary is allowed to interact with the user U with biometric distribution $\mathcal{D}$, as well as an honest server $\mathtt{S}_d$. We allow the adversary to control server $\mathtt{S}_{1\text{-}d}$ as well as interrupt and alter all communication in sessions between the honest server $\mathtt{S}_d$ and the user U. The probability the adversary samples a biometric (from a distinct distribution $\mathcal{D}' \neq \mathcal{D}$) sufficiently close to the registered template and thus is able to trivially break security by including the false acceptance probability ε_{fa} from Section 2.2. The adversary loses the game if it invokes the $\mathcal{O}_{\text{bio}}$ on the user's U distribution $\mathcal{D}$.

A BAKS scheme provides *authentication* if the following advantage is bounded by $\varepsilon_{fa} + \mathtt{negl}$, where negl is negligible in λ:

$$\mathbf{Adv}^{\mathtt{AUTH}}_{\mathtt{BAKS},\mathcal{A}}(1^\lambda) := \left|\Pr[\mathbf{Exp}^{\mathtt{AUTH}}_{\mathtt{BAKS},\mathcal{A}}(1^\lambda) = 1]\right|$$

4 Construction

In this section we present our Biometric-Authenticated Keyword Search (BAKS) protocol. We start with a high-level description of the protocol, and emphasise the functionality for the users. Then, we detail the protocol, and provide explicit description for the setup and registration algorithm, and refer to Figures 3 and 4 for the outsource and retrieval algorithms. To ease presentation, we illustrate in Figure 2 the method for key reconstruction that both outsource and retrieval algorithms use. Additionally, we provide an efficiency analysis and show how our protocol can be extended to handle multiple keywords and update of user's biometric data.

High-level Overview. Our construction is based on the password-authenticated keyword search [24], that we modify significantly to work with (noisy) biometric templates. We account for the imperfections of biometric data, where at each scan some bits may

```
Key reconstruction sub-routine KRec

User U(pp, $\mathcal{W}$)                          Server $\mathtt{S}_d(\mathtt{pp}, \mathtt{info}_d)$, with $\mathtt{info}_d = (x_d,$
  with $\mathcal{W} = \{W_i\}_{i=1}^N$                  $g^{r_1}, g^{r_2}, \{C_W^{(i)}, K_d^{(i)}, \mathtt{mk}_d^{(i)}\}_{i=1}^N, \mathtt{mk}_d)$
1 : for $1 \le i \le N$ do
      $a_i \leftarrow_\$ \mathbb{Z}_q^*,\ A_i \leftarrow g^{a_i} h_i^{W_i}$
2 :                  $\xrightarrow{\{A_i\}_{i=1}^N}$  $s_d, y_d \leftarrow_\$ \mathbb{Z}_q^*,\ Y_d \leftarrow g^{y_d}$   3
                          $R_d \leftarrow (g^{r_2})^{y_d},\ \mathtt{c}_d \leftarrow g^{s_d} h^{H(Y_d, R_d)}$   4
                          $(\mathtt{c}_{1\text{-}d}, s_{1\text{-}d}, Y_{1\text{-}d}, R_{1\text{-}d}) \xleftrightarrow{\mathtt{StS}}$   5
                          if $\mathtt{c}_{1\text{-}d} \ne g^{s_{1\text{-}d}} h^{H(Y_{1\text{-}d}, R_{1\text{-}d})}$ then abort   6
                          $Y \leftarrow Y_0 Y_1,\ R \leftarrow R_0 R_1$   7
                          for $1 \le i \le N$ do   8
                            $Z_d^{(i)} \leftarrow K_d^{(i)} (C_W^{(i)} A_i^{-1})^{y_d} (g^{r_1} R)^{-x_d}$
                            $\mu_d^{(i)} \leftarrow \mathtt{Tag}(\mathtt{mk}_d^{(i)}, (A_i, Y, Z_d^{(i)}))$
9 :   $\xleftarrow{Y, \{Z_d^{(i)}, \mu_d^{(i)}\}_{i=1}^N}$  return $(\{A_i\}_{i=1}^N, Y)$
10 : $T \leftarrow \emptyset$      $\xleftarrow{Y, \{Z_{1\text{-}d}^{(i)}, \mu_{1\text{-}d}^{(i)}\}_{i=1}^N}$
11 : for $1 \le i \le N$ do
       $T_i \leftarrow Z_0^{(i)} Z_1^{(i)} Y^{a_i}$
       $\mathtt{mk}_0^{(i)} \leftarrow \mathtt{KDF}_1(T_i, \mathtt{S}_0, \text{‘1’}),\ \mathtt{mk}_1^{(i)} \leftarrow \mathtt{KDF}_1(T_i, \mathtt{S}_1, \text{‘1’})$
       if $\mathtt{Ver}(\mathtt{mk}_0^{(i)}, (A_i, Y, Z_0^{(i)}), \mu_0^{(i)}) \wedge \mathtt{Ver}(\mathtt{mk}_1^{(i)}, (A_i, Y, Z_1^{(i)}), \mu_1^{(i)})$ then
         $T \leftarrow T \cup \{T_i\}$
12 : $K \leftarrow \mathtt{SS.Rec}(T)$, return $(K, \{A_i\}_{i=1}^N, Y)$
```

Fig. 2. This is a partial view of subroutine KRec that construct the key K from the biometric input W. The complete routine KRec involves the user U sending the message in Step 2 to both servers $\mathtt{S}_0$ and $\mathtt{S}_1$, then at Steps 9 and 10 receiving messages from both servers. We illustrate only the computation done by $\mathtt{S}_d$, as $\mathtt{S}_{1\text{-}d}$ performs the exact same steps. Moreover, we use $\xleftrightarrow{\mathtt{StS}}$ to model the communication between $\mathtt{S}_d$ and $\mathtt{S}_{1\text{-}d}$; at a high level $\mathtt{S}_d$ sends the commitment $\mathtt{c}_d$, waits for $\mathtt{c}_{1\text{-}d}$, then sends the opening s_d, Y_d, R_d and receives $s_{1\text{-}d}, Y_{1\text{-}d}, R_{1\text{-}d}$. SS.Rec returns either a valid key or an error symbol $\perp$.

differ from the initial template, by considering a threshold secret-sharing scheme [41]. We fix as N the total number of bits that can be extracted and used by users, but trivial extension can consider an arbitrary number, dependingent the user's biometric device.

When each user U registers in our protocol, he calls Register and creates a high entropy key $K = g^k$, for $k \leftarrow_\$ \mathbb{Z}_q^*$. All future computations between the user U and the servers $\mathtt{S}_0$ and $\mathtt{S}_1$ are done with respect to this key K. One core property of our protocol is that we do not require the user to store this key, and devise a method where the user's biometric data $\mathcal{W} = \{W_i\}_{i=1}^N$ is used to reconstruct this key. More precisely, the user applies $\mathtt{SS.Shr}(t, N, g, \mathbb{G}, q, K)$ to create the shares $\{K^{(i)}\}_{i=1}^N$ that can later be

```
Outsource Protocol Outsource

User $\mathtt{U}(\mathtt{pp}, \mathcal{W}, w, \mathtt{f})$          Server $\mathtt{S}_d(\mathtt{pp}, \mathtt{info}_d)$ with $\mathtt{info}_d = (x_d,$
                                                   $g^{r_1}, g^{r_2}, \{C_W^{(i)}, K_d^{(i)}, \mathtt{mk}_d^{(i)}\}_{i=1}^N, \mathtt{mk}_d)$

 1 : $(K, \{A_i\}_{i=1}^N, Y) \longleftarrow \boxed{\mathtt{KRec}} \longrightarrow (\{A_i\}_{i=1}^N, Y)$
 2 : if $K = \perp$ then abort
 3 : $z \leftarrow \mathtt{KDF}_2(K, w)$
 4 : $e \leftarrow_\$ \mathbb{G}, v \leftarrow \mathtt{PRF}(z, e)$
 5 : $\mathtt{mk}_u \leftarrow \mathtt{KDF}_1(K, \mathtt{U}, \text{‘0’})$
 6 : $\mu_c \leftarrow \mathtt{Tag}(\mathtt{mk}_u, (e, v, \mathtt{f}))$
 7 : $C \leftarrow (e, v, \mu_c)$
 8 : $\mathtt{mk}_d \leftarrow \mathtt{KDF}_1(K, \mathtt{S}_d, \text{‘1’})$,
 9 : $\mathtt{sk}_d \leftarrow \mathtt{KDF}_1(\mathtt{mk}_d, \{A_i\}_{i=1}^N, Y, \text{‘2’})$      $\mathtt{sk}_d \leftarrow \mathtt{KDF}_1(\mathtt{mk}_d, \{A_i\}_{i=1}^N, Y, \text{‘2’})$   10
11 : $\mu_{\mathtt{sk}_d} \leftarrow \mathtt{Tag}(\mathtt{sk}_d, (C, \mathtt{f}))$  $\xrightarrow{(C,\mathtt{f}),\mu_{\mathtt{sk}_d}}$  if $\mathtt{Ver}(\mathtt{sk}_d, (C, \mathtt{f}), \mu_{\mathtt{sk}_d})$ then   12
                                                       store $(C, \mathtt{f})$ on $\boldsymbol{C}_d$
                                                   else abort;
```

Fig. 3. This is a partial view of the interactive protocol Outsource, and captures only the interactions between user U that outsources file f with keyword w and biometric data W to server $\mathtt{S}_d$. The complete algorithm requires U to replicate Steps 9 and 11 for $\mathtt{S}_{1\text{-}d}$, and send $((C, \mathtt{f}), \mu_{\mathtt{sk}_{1\text{-}d}})$ to $\mathtt{S}_{1\text{-}d}$. Step 1 executes the routine KRec from Figure 2.

used to directly recover K. (See Remark 1.) Further, each share $K^{(i)}$ is split into $K_0^{(i)}$ and $K_1^{(i)}$, such that the reconstruction of $K^{(i)}$ can done only when the shares $K_0^{(i)}, K_1^{(i)}$ are processed together with an encryption $C_W^{(i)}$ of the bit W_i. The value $K_d^{(i)}$ is sent to $\mathtt{S}_d$, for $d \in \{0, 1\}$ and all $1 \leq i \leq N$. We also account for the fact that some of the inputs W_i the user is submitting at later stage may differ from the ones in the initial template $\mathcal{W}$ that has used when sharing K. As pointed by the steps in the subroutine KRec in Figure 2 we use MAC tags $\mu_d^{(i)}$ to identify the shares $T_i = K^{(i)}$ that have been correctly recovered before applying SS.Rec to reconstruct K.

Whenever the user U wants to outsource a file f characterized by the keyword w, he does so by calling Outsource. The first step is for the user to recover the key K using his current scanned biometric data $\mathcal{W}$, and establish some session identifiers $(\{A_i\}_{i=1}^N, Y)$ with the two servers $\mathtt{S}_0$, $\mathtt{S}_1$. Then, the user constructs a ciphertext C that should be stored together with the file f by both servers. The ciphertext C plays a fundamental role during retrieval, as it contains a MAC tag μ_c that can be used to authenticate the file provenance, as could have been produced only by the user with the knowledge of K and w. We assume the communication between U and server $\mathtt{S}_d$ during outsourcing and retrieval is performed over insecure channels, and in addition to $(C, \mathtt{f})$ it contains a MAC that uses a fresh KDF key composed of the long term key $\mathtt{mk}_d$ and the current session identifiers. If the MAC tag verifies, server $\mathtt{S}_d$ stores $(C, \mathtt{f})$ in an internal database $\boldsymbol{C}_d$. Notice, that each $\mathtt{S}_d$ stores a copy of the file f to achieve better availability.

At any point of time, the user U can retrieve all outsourced files that match the keyword w, by running Retrieve with servers $\mathtt{S}_0, \mathtt{S}_1$. Similar to the outsource algo-

```
Retrieve Protocol Retrieve

User $U(pp, \mathcal{W}, w)$                                   Server $S_d(pp, info_d)$ with $info_d = (x_d,$
                                                                 $g^{r_1}, g^{r_2}, \{C_W^{(i)}, K_d^{(i)}, mk_d^{(i)}\}_{i=1}^N, mk_d)$
1 : $(K, \{A_i\}_{i=1}^N, Y)$        $\longleftarrow \boxed{\texttt{KRec}} \longrightarrow$  $(\{A_i\}_{i=1}^N, Y)$
2 : if $K = \perp$ then abort                                  $L_d \leftarrow \emptyset$                                   3
4 : $mk_d \leftarrow KDF_1(K, S_d, '1')$
5 : $sk_d \leftarrow KDF_1(mk_d, \{A_i\}_{i=1}^N, Y, '2')$           $sk_d \leftarrow KDF_1(mk_d, \{A_i\}_{i=1}^N, Y, '2')$           6
7 : $z \leftarrow KDF_2(K, w), \mu_{sk_d} \leftarrow Tag(sk_d, z)$ $\xrightarrow{z, \mu_{sk_d}}$  if $Ver(sk_d, z, \mu_{sk_d})$ then     8
                                                                   for $(C, f) \in C_d$ do
                                                                     $(e, v, \mu_c) \leftarrow C$
                                                                     if $v = PRF(z, e)$ then
9 : $mk_u \leftarrow KDF_1(K, U, '0')$                                     $L_d \leftarrow L_d \cup (C, f)$
                                                                 else abort;
10 :                                    $\xleftarrow{L_d}$
11 : $L \leftarrow L_0 \cup L_1, F \leftarrow \emptyset$          $\xleftarrow{L_{1-d}}$
12 : for $(C, f) \in L$ with $C = (e, v, \mu_c)$ do
       if $(v = PRF(z, e) \wedge Ver(mk_u, (e, v, f), \mu_c))$ then
         $F \leftarrow F \cup f$
13 : return $F$
```

Fig. 4. This is a partial view of the interactive protocol Retrieve, and captures the interaction between user U, authenticated by the biometric data W, and server S_d, such that the user retrieves all files f that have keyword w. The complete algorithm requires U to replicate Steps 5 and 7 for S_{1-d}, then send $(t, \mu_{sk_{1-d}})$ to S_{1-d}. The set $\boldsymbol{L}$ does not contain duplicate elements.

rithm, the user inputs his biometric data $\mathcal{W}$, recovers the key K, and creates the session identifiers $(\{A_i\}_{i=1}^N, Y)$. Then, the user defines a search query $z \leftarrow \texttt{KDF}(K, w)$ that depends on the key K and keyword w. As the communication between U and S_d is over an insecure channel, z is sent together with a MAC that is built and verified using the knowledge of mk_d and the session identifiers $(\{A_i\}_{i=1}^N, Y)$. Finally, the user collects the responses from the two servers in the list $\boldsymbol{L}$, and discards the responses that do not match his query or have an invalid MAC tag μ_c to produce the final output list $\boldsymbol{F}$.

Detailed Description. We provide the complete specification of our protocol only for the setup and registration algorithms, and illustrate the outsource and retrieval steps in Figure 3 and 4. Both algorithms contain a key reconstruction phase, that for simplicity is presented separately in Figure 2.

– Setup(1^λ). Based on the type of biometric data considered, we define the number of bits N that can be extracted. For simplicity, we provide a uniform value for all users, but trivial extensions of our protocol can allow users to define their own number of bits during registration. Moreover, we generate a cyclic group $\mathbb{G}$ of order q with the following $(N+2)$ random generators: $g, h, \{h_i\}_{i=1}^{N} \leftarrow_{\$} \mathbb{G}$. Additionally, we introduce the building blocks $\mathtt{SS}, \mathtt{MAC}, \mathtt{KDF}_1, \mathtt{KDF}_2$, and $\mathtt{PRF}$ using the notations from Section 2. The algorithm outputs $\mathtt{pp} = (N, \mathbb{G}, q, g, h, \{h_i\}_{i=1}^{N}, \mathtt{SS}, \mathtt{MAC}, \mathtt{KDF}_1, \mathtt{KDF}_2, \mathtt{PRF})$.

– $\mathtt{Register}(\mathtt{pp}, \mathtt{U}, t, \mathcal{W}, \mathtt{S}_0, \mathtt{S}_1)$. Let $\mathcal{W} = \{W_i\}_{i=1}^{N}$. User $\mathtt{U}$ randomly selects a secret key $k \leftarrow_{\$} \mathbb{Z}_q^*$, sets $K = g^k$, and builds the shares $\{K^{(i)}\}_{i=1}^{N} \leftarrow \mathtt{SS.Shr}(t, N, g, \mathbb{G}, q, k)$. Additionally, the user generates $r_1, r_2, x_0, x_1 \leftarrow_{\$} \mathbb{Z}_q^*$ and $\{K_0^{(i)}\}_{i=1}^{N} \leftarrow_{\$} \mathbb{G}$, then sets $X = g^{x_0+x_1}$, $K_1^{(i)} = X^{r_1} K^{(i)} (K_0^{(i)})^{-1}$ and $C_W^{(i)} = X^{r_2} h_i^{W_i}$ for all $1 \leq i \leq N$. For each server $\mathtt{S}_d \in \{\mathtt{S}_0, \mathtt{S}_1\}$, the values $\mathtt{mk}_d \leftarrow \mathtt{KDF}_1(K, \mathtt{S}_d, \text{‘1’})$ and $\mathtt{mk}_d^{(i)} \leftarrow \mathtt{KDF}_1(K^{(i)}, \mathtt{S}_d, \text{‘1’})$ for all $1 \leq i \leq N$, are derived and used to define $\mathtt{info}_d = (x_d, g^{r_1}, g^{r_2}, \{C_W^{(i)}, K_d^{(i)}, \mathtt{mk}_d^{(i)}\}_{i=1}^{N}, \mathtt{mk}_d)$ before $\mathtt{info}_d$ is sent over an authenticated and secure channel to $\mathtt{S}_d$.

Correctness. Correctness of each key share $K^{(i)}$ can be observed by computing the following:

$$\begin{aligned}
Z_0^{(i)} Z_1^{(i)} Y^{a_i} =& K_0^{(i)} (C_W^{(i)} A_i^{-1})^{y_0} (g^{r_1} R)^{-x_0} \cdot K_1^{(i)} (C_W^{(i)} A_i^{-1})^{y_1} (g^{r_1} R)^{-x_1} \cdot g^{a_i(y_0+y_1)} \\
=& K_0^{(i)} K_1^{(i)} (C_W^{(i)} A_i^{-1} g^{a_i})^{(y_0+y_1)} (g^{r_1} R)^{-(x_0+x_1)} \\
=& X^{r_1} K^{(i)} (X^{r_2} h_i^{W_i} (g^{a_i} h_i^{W_i})^{-1} g^{a_i})^{(y_0+y_1)} (g^{r_1} g^{r_2(y_0+y_1)})^{-(x_0+x_1)} \\
=& g^{(x_0+x_1)r_1} K^{(i)} g^{(x_0+x_1)r_2(y_0+y_1)} (g^{r_1} g^{r_2(y_0+y_1)})^{-(x_0+x_1)} = K^{(i)}
\end{aligned}$$

The correctness of $\mathtt{SS}$ ensures that they key K can be recovered from t or more correct shares.

Remark 1. The desired level of accuracy for the biometric measurement is defined by the $\mathtt{SS}$ threshold $t \leq N$, such that any t bits W_i that are correct would lead to the recovery of the key K. In particular, the false rejection parameter τ_1 is proportional to the value of t and similarly τ_2, the false accept parameter is inversely proportional to t. However, the true values for τ_1 and τ_2 cannot be derived from the threshold parameter alone, as they also depend on the accuracy of the biometric sensor. We assume that selection of t can be made based on the sensors used such that the scheme adheres to appropriate security guarantees.

4.1 Efficiency Analysis and Improvements

We compare the efficiency of our $\mathtt{BAKS}$ scheme with the PAKS construction from [24], which relies on a password and by this also supports searchable encryption without requiring users to store and manage private keys. We immediately note that $\mathtt{BAKS}$ has the same number of rounds for the outsourcing and retrieval as PAKS. This leads to $\mathtt{BAKS}$ having the same server-side storage $\mathcal{O}(D)$ as PAKS, for a database of size D, but without any user-side storage. The computational cost of PAKS for the user and each server is 3, resp. 8, exponentiations, while in our $\mathtt{BAKS}$ scheme it amounts to $3N$, resp. $6 + 2N$, exponentiations, for the noisy biometric sample of N bits. During the key reconstruction phase, the communication overheads are 8 group elements, and 2 MACs for PAKS, and $2 + 6N$ group elements along with 2 MAC tags for $\mathtt{BAKS}$. The linear growth in $\mathtt{BAKS}$ complexity is unavoidable due to existence of noise and hence bitwise processing of the biometric template.

We remark that it is possible to reduce the value of N by splitting the biometric sample W into k blocks of N/k bits, which would reduce the computational and communication costs to $O(N/k)$. This could be used, for example, where the biometric is a fingerprint, the data stored represents k features (or *minutiae* [26]) and the data stored in each block is the feature-type and location. While we would not loose security by supporting this mechanism, the corresponding false-rejection rate would be significantly affected by the accuracy of the biometric scanner. In particular, even one bit difference in the feature description would render the share invalid, whereas the presented BAKS scheme offers more flexible tolerance to errors.

BAKS with sublinear search complexities. BAKS achieves $\mathcal{O}(D)$ search complexity (as highlighted in Step. 7 of Figure 4), for a database DB of size D, whilst state-of-the-art schemes achieve a bound of $\mathcal{O}(\log D)$. We can improve BAKS search complexity by employing some of the general ideas from [14,16,43]. However, this would introduce security and/or functionality limitations. One first approach would be to restrict only to static databases [14, 16], and loose the dynamic aspect of our solution. Another approach would be to use ORAM and maintain the dynamic aspects of our database, but at the cost of requiring periodic (costly, i.e., $\mathcal{O}(D \log D)$ comparisons) oblivious sorting [43]. The final method, and the one we highlight in the rest of this section, is to consider dynamic databases with limited update (i.e., handles only adding entries) [16].

At a high level, [16] does each outsourcing over batches of documents, and treats each update as an outsource of a static database DB. Their optimisation relies on a look-up table T that stores pointers to location of documents in DB, such that the table inputs depend on the document keyword. In the setup phase, the look-up table T is initialized to empty. We extend the outsource protocol from Figure 3 to upload the database $DB = \{(w_i, \mathtt{f}_i) | 1 \leq i \leq N\}$ instead of a single file $\mathtt{f}$ and keyword w. First, for all unique keywords w_i, we create the list $L_i = \{(w_i, \mathtt{ind}(\mathtt{f}_{i_j}), \mathtt{f}_{i_j}) | (w_i, \mathtt{f}_{i_j}) \in DB\}$ with $\mathtt{ind}(\mathtt{f}_{i_j})$ the index in DB where file $\mathtt{f}_{i_j}$ can be found. Then, for each entry $(w_i, \mathtt{ind}(\mathtt{f}_{i_j}), \mathtt{f}_{i_j})$ steps 1-9 are performed, followed by the generation of the key $o_{i_j} \leftarrow \mathtt{KDF}_2(z_i, j)$ and setting the look-up table $T[o_{i_j}] = \mathtt{ind}(\mathtt{f}_{i_j})$. Finally, Step 11 is executed for all entries and the entire encrypted database $\boldsymbol{C}_d$ is sent with the look-up table. Notice that $\boldsymbol{C}_d$ preserves the same order of elements from DB, and $\mathtt{ind}(\cdot)$ should give the same location for both $\boldsymbol{C}_d$ and DB. The retrieval protocol is performed in the same way as in Figure 4, except the server receives $(z, \mu_{\mathtt{sk}_d}, \{o_{i_j}\}_{j=1}^{n_i})$ in step 7, for $|L_i| = n_i$. Then, the values $\{o_{i_j}\}_{j=1}^{n_i}$ are used to identify the entries $\{(C_{i_j}, \mathtt{f}_{i_j})\}_{j=1}^{n_i}$ from $\boldsymbol{C}_d$, based on $\mathtt{ind}(\mathtt{f}_{i_j})$ from the look-up table T. To stop adversaries from trivially differentiating based on the list size, we can use the techniques in [16] to extend DB to DB^* with dummy files, such that all lists have the same size $|L_i| = n$, for $n = \max_i\{n_i\}$. This version of BAKS with sublinear search complexity would satisfy the same security guarantees as the initial version. The main limitation is that now we are restricted to dynamic databases that only allow addition of entries, and not removal. The security intuition is based on the fact that each outsource operation is treated as an outsource of a new independent static database.

4.2 Extensions with multiple keywords

The construction we have presented in this paper is limited to only allowing a user to search for a single keyword. In practise, it is likely that a user would want to outsource a file with multiple keywords in a single execution, or search for multiple keywords in one running of the retrieve protocol. We briefly show that our BAKS construction allows for a natural extension to support this functionality.

Outsourcing with multiple keywords. Here, we allow a user to outsource a file $\mathtt{f}$ with keywords $\overline{w} := (w_1, ..., w_k)$. This can be achieved by computing $t_j \leftarrow \mathtt{KDF}_2(K, w_j)$, $v_j \leftarrow \mathtt{PRF}(t_j, e)$ and finally $\mu_c \leftarrow \mathtt{tag}(\mathtt{mk}_u(e, \overline{v}))$ as part of $\mathtt{out}$. Each server receives the ciphertext $C = (e, \overline{v}, \mu_c)$ where $\overline{v} = (v_1, ..., v_k)$.

Retrieval based on complex queries. A user submits the keywords $\overline{w} = (w_1, ..., w_k)$ to a single execution of the retrieve protocol. It computes search queries $t_j \leftarrow \mathtt{KDF}_2(K, w_j)$ for each keyword. Then for every entry $(e, \overline{v}, \mu_c, \mathtt{f})$ in the database $\boldsymbol{C}_d$, the server computes $v'_j = \mathtt{PRF}(t_j, e)$ $(j = 1, ..., k)$ and updates the output list $\mathcal{A}_d$ according to the search query. We note this approach supports conjunctive, disjunctive and subset type queries which can be enforced by line 7 in the retrieve protocol.

4.3 Biometric Update

Our BAKS construction supports a user who may wish to update their registered biometric template $\mathcal{W}$ to a new biometric template $\mathcal{W}^*$. This could be the case when a user changes their biometric device (e.g., from fingerprint to iris recognition), or does not have access to the initial biometric feature they registered (e.g., vision loss). To prevent the need for re-encryption of all keywords for all outsourced files, we do not change the key K and simply update only the commitments to the biometric template. If the user cannot use their already registered biometric template for authentication, we can use an alternative authenticated channel to each of the servers $\mathtt{S}_0$ and $\mathtt{S}_1$ to update the biometric sample. This process is executed independent of the previous biometric sample, and has the following steps:

- Servers $\mathtt{S}_d$, for $d \in \{0, 1\}$, compute $X_d \leftarrow g^{x_d}$ and send this to the user $\mathtt{U}$ over a secure and authenticated channel.
- User $\mathtt{U}$ then samples $r_2^* \leftarrow_{\$} \mathbb{Z}_q^*$, computes $C_{W^*}^{(i)} \leftarrow (X_0 X_1)^{r_2^*} h_i^{W_i^*}$ for each bit W_i^* in the new biometric, and forwards $(g^{r_2^*}, \{C_{W^*}^{(i)}\}_{i=1}^N)$ to both servers, on the same authenticated channel.
- Each server $\mathtt{S}_d$ updates their information $\mathtt{info}_d$ to reflect the new biometric sample, from $(g^{r_2}, \{C_W^{(i)}\}_{i=1}^N)$ to $(g^{r_2^*}, \{C_{W^*}^{(i)}\}_{i=1}^N)$, for $d \in \{0, 1\}$.

5 Security Analysis

In this section, we give proof that our construction for BAKS in Figures 2, 3 and 4 meets the security definitions from Section 3.2. We extend the table E to also include some secret values, s.t. $E[j] \leftarrow (d, \mathcal{D}, \mathtt{info}_d, r_2, r_1, x_{1-d})$ for some session identifier j. This is possible due to the oracle playing the role of the user and generating the values r_2, r_1, x_{1-d}.

Theorem 1. *Our* BAKS *construction is* IND-CKA *secure given* $\mathtt{KDF}_1$ *and* $\mathtt{KDF}_2$ *are secure,* PRF *is pseudorandom,* MAC *is unforgeable,* SS *offers privacy, the discrete logarithm (DL) assumption is hard in* $\mathbb{G}$ *and the liveness assumption holds.*

Proof. $\boldsymbol{\mathcal{G}_0}$: Game $\boldsymbol{\mathcal{G}_0}$ is defined exactly by the experiment $\mathbf{Exp}^{\texttt{IND-CKA}-b}_{\texttt{BAKS},\mathcal{A}}$. Thus:

$$\Pr[\mathcal{G}_0 = 1] = \Pr[\mathbf{Exp}^{\texttt{IND-CKA}-b}_{\texttt{BAKS},\mathcal{A}} = 1].$$

$\boldsymbol{\mathcal{G}_1}$: This game is obtained from $\boldsymbol{\mathcal{G}_0}$ by aborting if y_d is used in more than one protocol session in oracles $\mathcal{O}_{\text{out}}$, $\mathcal{O}_{\text{retU}}$ or $\mathcal{O}_{\text{retS}}$, the rest remains the same. The probability of winning $\boldsymbol{\mathcal{G}_1}$ over $\boldsymbol{\mathcal{G}_0}$ is bounded by the probability of sampling y_d twice or more from j samples of a uniform distribution over $\mathbb{Z}_q^*$. Thus we have:

$$\left| \Pr[\boldsymbol{\mathcal{G}_1} = 1] - \Pr[\boldsymbol{\mathcal{G}_0} = 1] \right| \leq \frac{j^2}{q}$$

$\boldsymbol{\mathcal{G}_2}$: This game is identical to $\boldsymbol{\mathcal{G}_1}$ except that the game aborts if Y appears in two different protocol sessions through oracles $\mathcal{O}_{\text{out}}$, $\mathcal{O}_{\text{retU}}$ or $\mathcal{O}_{\text{retS}}$. We note because the Pedersen commitment c_d is perfectly hiding, Y_{1-d} must be independent of Y_d. Additionally, the binding property (based on the DL assumption) ensures that it is computationally hard to open c_{1-d} to a $Y'_d \neq Y_d$. Hence, since Y_d and Y_{1-d} are fresh and independent, Y is also fresh based on the hardness of DL assumption. Thus we have:

$$\left| \Pr[\boldsymbol{\mathcal{G}_2} = 1] - \Pr[\boldsymbol{\mathcal{G}_1} = 1] \right| \leq \mathbf{Adv}^{DL}_{\mathcal{B}}$$

$\boldsymbol{\mathcal{G}_3}$: This game is defined by $\boldsymbol{\mathcal{G}_2}$ where PRF is replaced with a function. In the outsourcing protocol, when the adversary has control of U, it is able to provide a μ'_c such that $\texttt{Ver}(\texttt{mk}_d, (e, v, \texttt{f}), \mu'_c) = 1$. If $\mathcal{A}$ is able to do so without knowledge of $\texttt{mk}_d$, he breaks the unforgeability of MAC.
Hence, the games are bounded by:

$$\left| \Pr[\boldsymbol{\mathcal{G}_3} = 1] - \Pr[\boldsymbol{\mathcal{G}_2} = 1] \right| \leq \mathbf{Adv}^{unforge}_{\mathcal{B}}$$

We now split the games conditioned on the value of some components. In $\boldsymbol{\mathcal{G}_4}$, the adversary does not have knowledge of $\texttt{mk}_d$ and is therefore forced to try to break unforgeability of the message authentication code. We let prime-counterpart $\texttt{mk}'_d$ denote the adversary's guess at $\texttt{mk}_d$ and capture the case the adversary *does* know $\texttt{mk}_d$ in game $\boldsymbol{\mathcal{G}_5}$ by letting $\texttt{mk}'_d = \texttt{mk}_d$. The law of total probability gives us:

$$\Pr[\boldsymbol{\mathcal{G}_3} = 1] = \underbrace{(\Pr[\boldsymbol{\mathcal{G}_3} = 1] \wedge \texttt{mk}'_u \neq \texttt{mk}_u)}_{\boldsymbol{\mathcal{G}_4}} + \underbrace{(\Pr[\boldsymbol{\mathcal{G}_3} = 1] \wedge \texttt{mk}'_u = \texttt{mk}_u)}_{\boldsymbol{\mathcal{G}_5}}$$

$\boldsymbol{\mathcal{G}_4}$: This game is defined by $\boldsymbol{\mathcal{G}_3}$ where PRF is replaced with a function. In the outsourcing protocol, when the adversary has control of U, it is able to provide a μ'_c such that $\texttt{Ver}(\texttt{mk}_d, (e, v, \texttt{f}), \mu'_c) = 1$. If $\mathcal{A}$ is able to do so without knowledge of $\texttt{mk}_d$, he breaks the unforgeability of MAC. Hence, the game is bounded by:

$$\Pr[\boldsymbol{\mathcal{G}_4} = 1] \leq (q_o + q_t)\mathbf{Adv}^{PRF}_{\mathcal{B}}$$

$\boldsymbol{\mathcal{G}_5}$: We now split the games conditioned on the value of some components. In $\boldsymbol{\mathcal{G}_6}$, the adversary does not have knowledge of K and is therefore forced to try to break the security of KDF, we capture the case the adversary *does* know K in game $\boldsymbol{\mathcal{G}_7}$. The law of total probability gives:

$$\Pr[\boldsymbol{\mathcal{G}_5} = 1] = \underbrace{(\Pr[\boldsymbol{\mathcal{G}_5} = 1] \wedge K' \neq K)}_{\boldsymbol{\mathcal{G}_6}} + \underbrace{(\Pr[\boldsymbol{\mathcal{G}_5} = 1] \wedge K' = K)}_{\boldsymbol{\mathcal{G}_7}}$$

$\boldsymbol{\mathcal{G}_6}$: In this game, the adversary does not know the key which we denote $K' \neq K$, where K' is his guess at the key. Thus, the adversary is only able to win this game if he can break the security of the key derivation function. Hence we have the following bound its success probability:

$$\Pr[\boldsymbol{\mathcal{G}_6} = 1] \leq \mathbf{Adv}_{\mathcal{B}}^{KDF}$$

$\boldsymbol{\mathcal{G}_7}$: In this game, we necessarily have $K' = K$ and we analyse the adversary's probability of being able to compute K. We define game $\boldsymbol{\mathcal{G}_7}$ to be $\boldsymbol{\mathcal{G}_5}$ except that `SS.share` and `SS.rec` are replaced with functions F_s and F_r. T_s is initialised as an empty table at the beginning of the experiment, on input K, and returns $\{K_i\}_{i=1}^N$ if $\exists(K, \{K_i\}_{i=1}^N) \in T_s$.

We aim to show that the success probability of this game is bounded by an adversary against the secret sharing scheme. To start, the adversary to SS, $\mathcal{B}$, guesses a user i for which $\mathcal{A}$ is going to break the authentication property, he does with probability $1/U$ for U total users. For $\mathtt{U}_j$ where $j \neq i$, he runs the protocol honestly. Upon registration and when $j = 1$, he invokes his challenge oracle from the privacy game against `SS` by inputting two challenge secrets S_0 and S_1, the game returns the shares $\{S_b^{(i)}\}_{i=1}^N$ for some $b \in \{0, 1\}$. The adversary $\mathcal{B}$ wins the game if he can correctly guess the challenge bit b. It invokes $\boldsymbol{\mathcal{G}_5}$ with $\mathcal{A}$ in the role of `U` and $\mathtt{S}_{1\text{-}d}$.

If $\mathcal{A}$ submits his biometric reading via the $\mathcal{O}_{\text{bio}}$ oracle, with probability ε_{fa}, for a sampled biometric $\mathcal{W}'$ (sampled from a distribution $\mathcal{D}'$) we have $d(\mathcal{W}', \mathcal{W}) \leq \tau_2$ (equivalent to more than $t-1$ bits matching). Thus, with probability $1-\varepsilon_{fa}$, $\mathcal{A}$ is only able to compute at most $t-1$ correct shares of K (for biometric $\mathcal{W}$) given $\{Z_0^{(i)'}, Z_1^{(i)'}, Y^{a'_i}\}_{i=1}^N$ from $\mathcal{W}'$. The adversary has a negligible probability of reconstructing the t^{th} share by either guessing $Z_d^{(t)}$ such that $T_t = Z_0^{(t)} Z_1^{(t)} Y^{a_t}$, or finding another user (in a different session) with common features to the challenge user, breaking liveness assumption. In particular, the liveness assumption prevents an adversary from submitting $\mathcal{W}'$ sampled from the user's distribution $\mathcal{D}$. The `SS` adversary $\mathcal{B}$ waits for $\mathcal{A}$ to finish the run of the user. If it outputs a response $(t, \mu_{\mathtt{sk}_d})$ then adversary $\mathcal{B}$ answers its `SS` challenge with $b = 0$. We note that if $b = 1$, then $\mathcal{A}$ would not have been able to complete the protocol due to the security properties of `KDF` and `MAC` that we have ruled out in games $\boldsymbol{\mathcal{G}_3}$ and $\boldsymbol{\mathcal{G}_4}$, respectively. Thus, any correct response from $\mathcal{A}$ implies the challenge bit $b = 0$. Hence, the probability of winning game $\boldsymbol{\mathcal{G}_5}$ is bound by the probability that $\mathcal{B}$ can break the privacy property of the secret sharing scheme.

$$\Pr[\boldsymbol{\mathcal{G}_7} = 1] \leq \frac{1}{|U|} \mathbf{Adv}_{\mathcal{B}}^{priv} + \varepsilon_{fa}$$

By the sequence of games $\boldsymbol{\mathcal{G}_0}$ to $\boldsymbol{\mathcal{G}_7}$, we have shown that the probability of a `BAKS` adversary against the `IND-CKA` property is bounded by $\varepsilon_{fa} + \mathtt{negl}$ where `negl` is negligible in the security parameter λ. □

Theorem 2. *Our* `BAKS` *construction provides authentication given* $\mathtt{KDF}_1$ *and* $\mathtt{KDF}_2$ *are secure,* `MAC` *is unforgeable,* `SS` *offers privacy, the discrete logarithm (DL) assumption is hard in* $\mathbb{G}$ *and the liveness assumption holds.*

Proof. $\boldsymbol{\mathcal{G}_0}$: Game $\boldsymbol{\mathcal{G}_0}$ is defined exactly by the experiment $\mathbf{Exp}_{\mathtt{BAKS},\mathcal{A}}^{\mathtt{AUTH}}$, where the oracles in the experiment are instantiated by the following:

The success probability of an adversary winning game $\mathcal{G}_0$ is identical to adversary against the experiment $\mathbf{Exp}^{\mathtt{AUTH}}_{\mathtt{BAKS},\mathcal{A}}$.

$$\Pr[\mathcal{G}_0 = 1] = \Pr[\mathbf{Exp}^{\mathtt{AUTH}}_{\mathtt{BAKS},\mathcal{A}} = 1].$$

$\mathcal{G}_1$: This game is obtained from $\mathcal{G}_0$ by aborting if y_d is used in more than one protocol session in oracles $\mathcal{O}_{\text{out}}$, $\mathcal{O}_{\text{retU}}$ or $\mathcal{O}_{\text{retS}}$, the rest remains the same. The probability of winning $\mathcal{G}_1$ over $\mathcal{G}_0$ is bounded by the probability of sampling y_d twice or more from j samples of a uniform distribution over $\mathbb{Z}_q^*$. Thus we have:

$$\left| \Pr[\mathcal{G}_1 = 1] - \Pr[\mathcal{G}_0 = 1] \right| \leq \frac{j^2}{q}$$

$\mathcal{G}_2$: This game is identical to $\mathcal{G}_1$ except that the game aborts if Y appears in two different protocol sessions through oracles $\mathcal{O}_{\text{out}}$, $\mathcal{O}_{\text{retU}}$ or $\mathcal{O}_{\text{retS}}$. We note because the Pedersen commitment c_d is perfectly hiding, Y_{1-d} must be independent of Y_d. Additionally, the binding property (based on the DL assumption) ensures that it is computationally hard to open c_{1-d} to a $Y'_d \neq Y_d$. Hence, since Y_d and Y_{1-d} are fresh and independent, Y is also fresh based on the hardness of DL assumption. Thus we have:

$$\left| \Pr[\mathcal{G}_2 = 1] - \Pr[\mathcal{G}_1 = 1] \right| \leq \mathbf{Adv}^{DL}_{\mathcal{B}}$$

We now split the games conditioned on the value of some components. In $\mathcal{G}_3$, the adversary does not have knowledge of $\mathtt{sk}_d$ and is therefore forced to try to break unforgeability. We capture the case the adversary *does* know $\mathtt{sk}_d$ in game $\mathcal{G}_4$. The law of total probability gives us:

$$\Pr[\mathcal{G}_2 = 1] = \underbrace{(\Pr[\mathcal{G}_2 = 1] \wedge \mathtt{sk}'_d \neq \mathtt{sk}_d)}_{\mathcal{G}_3} + \underbrace{(\Pr[\mathcal{G}_2 = 1] \wedge \mathtt{sk}'_d = \mathtt{sk}_d)}_{\mathcal{G}_4}$$

$\mathcal{G}_3$: This game is defined by $\mathcal{G}_2$ where we restrict the adversary to not have knowledge of $\mathtt{sk}_d$. In the outsourcing protocol, when the adversary has control of $\mathtt{U}$, it is able to provide a $\mu'_{\mathtt{sk}_d}$ such that $\mathtt{Ver}(\mathtt{sk}_d, (e, v, \mathtt{f}), \mu'_{\mathtt{sk}_d}) = 1$. If $\mathcal{A}$ is able to do so without knowledge of $\mathtt{sk}_d$, he breaks the unforgeability of $\mathtt{MAC}$.
Hence, the games are bounded by:

$$\Pr[\mathcal{G}_3 = 1] \leq \mathbf{Adv}^{unforge}_{\mathcal{B}}$$

$\mathcal{G}_4$: We now split the games conditioned on the value of some components. In $\mathcal{G}_5$, the adversary does not have knowledge of K and is therefore forced to try to break the security of $\mathtt{KDF}$. We capture the case the adversary *does* know K in game $\mathcal{G}_6$. The law of total probability gives:

$$\Pr[\mathcal{G}_4 = 1] = \underbrace{(\Pr[\mathcal{G}_4 = 1] \wedge K' \neq K)}_{\mathcal{G}_5} + \underbrace{(\Pr[\mathcal{G}_4 = 1] \wedge K' = K)}_{\mathcal{G}_6}$$

$\mathcal{G}_5$: We define game $\mathcal{G}_5$ to be $\mathcal{G}_4$ except we make the restriction the adversary does know the key K. For each session i of the oracles $\mathcal{O}_{\text{out}}$ and $\mathcal{O}_{\text{retU}}$, the value $t \leftarrow \mathtt{KDF}_2(K, w)$ is replaced with $t \leftarrow F_2(i, w)$. T_2 is initialised as an empty table at the beginning of $\mathcal{G}_4$ and define F_2 as follows. If $\exists (i, w, t) \in T_2$, then it returns t, otherwise it samples a fresh $t \leftarrow_{\$} K_{\texttt{PRF}}$ and stores (i, w, t) in T_2.

We also replace the values $\texttt{mk}_u \leftarrow \texttt{KDF}_1(K, \texttt{U}, \text{'0'}), \texttt{mk}_d \leftarrow \texttt{KDF}_1(K, \texttt{S}_d, \text{'1'})$ and $\texttt{mk}_{1\text{-}d} \leftarrow \texttt{KDF}_1(K, \texttt{S}_{1\text{-}d}, \text{'1'})$ with the function F_1 such that $\texttt{mk}_u \leftarrow F_1(i, \texttt{U}, \text{'0'}), \texttt{mk}_d \leftarrow F_1(i, \texttt{S}_d, \text{'1'})$ and $\texttt{mk}_{1\text{-}d} \leftarrow F_1(i, \texttt{S}_{1\text{-}d}, \text{'1'})$. The game $\boldsymbol{\mathcal{G}_5}$ initialises a table T_1, and for $F_1 : \{0,1\}^* \rightarrow \texttt{MAC}_{keyspace}$ define as follows. If $\exists(i, id, k, mk) \in T_1$ then $F_1(i, id, k)$ returns mk, otherwise it samples $mk \leftarrow_{\$} \texttt{MAC}_{keyspace}$ and stores (i, id, k, mk) in T_1 and returns mk.
By the uniform distribution of K, and the indistinguishability property of $\texttt{KDF}_1$ and $\texttt{KDF}_2$, we have the following bound:

$$\Pr[\boldsymbol{\mathcal{G}_5} = 1] \leq (q_o + q_t + q_r)\mathbf{Adv}_{\mathcal{B}}^{KDF}$$

$\boldsymbol{\mathcal{G}_6}$: We follow the same strategy as that in game $\boldsymbol{\mathcal{G}_7}$ in the proof for Theorem 1. Thus, the advantage of an adversary against $\boldsymbol{\mathcal{G}_6}$ is bounded by the success probability of an adversary against the privacy of the secret sharing scheme.

$$\Pr[\boldsymbol{\mathcal{G}_6} = 1] \leq \mathbf{Adv}_{\mathcal{B}}^{priv} + \varepsilon_{fa}$$

By the sequence of games $\boldsymbol{\mathcal{G}_0}$ to $\boldsymbol{\mathcal{G}_6}$, we have shown that the probability of a $\texttt{BAKS}$ adversary against the $\texttt{Auth}$ property is bounded by $\varepsilon_{fa} + \texttt{negl}$ where $\texttt{negl}$ is negligible in λ. □

6 Conclusion

We introduced Biometric-Authenticated Keyword Search (BAKS), a novel searchable encryption scheme that uses biometric data as authentication mechanism for outsourcing and retrieval of files indexed by encrypted keywords. We accounted for the imperfections in the biometric measurements, and designed BAKS such that users can authenticate via biometric data that is "similar" to their template. This degree of similarity is modeled using threshold secret sharing techniques, and selecting the threshold involves at least considering the accuracy of the measuring device and the nature of used biometric factors, e.g. fingerprint, iris, face, voice.

BAKS employs a two-server architecture together with the liveness assumption to provide security guarantees and doesn't assume that biometric data remains private. Moreover, BAKS allows users to update their biometric template without the need to re-encrypt the outsourced keywords and provides support for multi-keyword outsourcing and search. BAKS ensures that only legitimate users can outsource and retrieve files (via authentication property) and confidentiality of the outsourced keywords (via indistinguishability against chosen keyword attacks) in presence of at most one compromised server.

Acknowledgements. Daniel Gardham was supported by the UK Government PhD studentship scheme. Mark Manulis and Constantin Cătălin Drăgan were supported by the EPSRC project TAPESTRY (EP/N02799X).

References

1. Biometric presentation attack detection. Standard ISO/IEC WD 30107-3:2017, International Organization for Standardization, Geneva, CH, 2017.

2. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. *J. Cryptology*, 21(3):350–391, 2008.
3. Z. Akhtar, C. Micheloni, and G. L. Foresti. Biometric liveness detection: Challenges and research opportunities. *IEEE Security & Privacy*, 13(5):63–72, 2015.
4. L. Ballard, S. Kamara, and F. Monrose. Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. In *ICICS 2005*, pages 414–426, 2005. LNCS 3783.
5. M. Bellare, R. Canetti, and H. Krawczyk. Keying Hash Functions for Message Authentication. In *CRYPTO 1996*, pages 1–15, 1996. LNCS 1109.
6. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. In *EUROCRYPT 2004*, pages 506–522, 2004. LNCS 3027.
7. D. Boneh and B. Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. In *TCC 2007*, pages 535–554, 2007. LNCS 4392.
8. X. Boyen. Reusable cryptographic fuzzy extractors. In *ACM CCS 2004*, pages 82–91. ACM, 2004.
9. X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. D. Smith. Secure Remote Authentication Using Biometric Data. In *EUROCRYPT 2005*, pages 147–163, 2005. LNCS 3494.
10. J. Bringer, H. Chabanne, and B. Kindarji. Error-tolerant searchable encryption. In *IEEE ICC 2009*, pages 1–6, 2009.
11. J. Camenisch, A. Lehmann, A. Lysyanskaya, and G. Neven. Memento: How to reconstruct your secrets from a single password in a hostile environment. In *CRYPTO 2014*, volume 8617 of *LNCS*, pages 256–275. Springer, 2014.
12. R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. D. Smith. Reusable fuzzy extractors for low-entropy distributions. In *EUROCRYPT 2016*, pages 117–146, 2016. LNCS 9665.
13. D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 668–679. ACM, 2015.
14. D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *CRYPTO 2013, Part I*, pages 353–373, 2013.
15. R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang. Dual-server public-key encryption with keyword search for secure cloud storage. *Trans. Info. For. Sec.*, 11(4):789–798, 2016.
16. R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. *J. of Computer Security*, 19(5):895–934, 2011. A preliminary version appeared in ACM CCS 2006.
17. J. Daugman. How iris recognition works. *IEEE Trans. Circuits Syst. Vid. Techn.*, 14(1):21–30, 2004.
18. Y. Dodis, J. Katz, L. Reyzin, and A. D. Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. In *CRYPTO 2006*, pages 232–250, 2006. LNCS 4117.
19. Y. Dodis, L. Reyzin, and A. D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT 2004*, pages 523–540, 2004. LNCS 3017.
20. FIDO Alliance. FIDO 2.0. Technical report, 2015. `https://fidoalliance.org/specifications`, Accessed 03-March-2019.
21. N. Fleischhacker, M. Manulis, and A. Azodi. A Modular Framework for Multi-Factor Authentication and Key Exchange. In *SSR 2014*, pages 190–214, 2014. LNCS 8893.
22. B. Fuller, S. Simhadri, and J. Steel. Reusable authentication from the iris. Cryptology ePrint Archive, Report 2017/1177, 2017. `https://eprint.iacr.org/2017/1177`.
23. J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.
24. K. Huang, M. Manulis, and L. Chen. Password authenticated keyword search. In *PAC 2017*, pages 129–140, 2017.

25. M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS 2012*, 2012.
26. A. K. Jain, A. Ross, and S. Prabhakar. Fingerprint matching using minutiae and texture features. In *ICIP 2001*, pages 282–285, 2001.
27. A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *ACM CCS 1999*, pages 28–36, 1999.
28. G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill. Generic attacks on secure outsourced databases. In *ACM CCS 2016*, pages 1329–1340, 2016.
29. H. Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In *CRYPTO 2010*, volume 6223 of *LNCS*, pages 631–648. Springer, 2010.
30. V. Kuchta and M. Manulis. Public Key Encryption with Distributed Keyword Search. In *INTRUST 2015*, volume 9565 of *LNCS*, pages 62–83. Springer, 2016.
31. C. Liu, L. Zhu, M. Wang, and Y. Tan. Search pattern leakage in searchable encryption: Attacks and new construction. *Inf. Sci.*, 265:176–188, 2014.
32. M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988.
33. M. Naveed, S. Kamara, and C. V. Wright. Inference attacks on property-preserving encrypted databases. In *ACM SIGSAC 2015*, pages 644–655, 2015.
34. L. O'Gorman. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*, 91(12):2021–2040, 2003.
35. C. Örencik, A. Selcuk, E. Savas, and M. Kantarcioglu. Multi-keyword search over encrypted data with scoring and search pattern obfuscation. *Int. J. Inf. Sec.*, 15(3):251–269, 2016.
36. D. J. Park, K. Kim, and P. J. Lee. Public key encryption with conjunctive field keyword search. In *WISA 2004*, pages 73–86, 2004. LNCS 3325.
37. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO 1991*, pages 129–140, 1991. LNCS 576.
38. D. Pointcheval and S. Zimmer. Multi-factor Authenticated Key Exchange. In *ACNS 2008*, volume 5037 of *LNCS*, pages 277–295, 2008.
39. S. Prabhakar, S. Pankanti, and A. K. Jain. Biometric recognition: Security and privacy concerns. *IEEE Security & Privacy*, 1(2):33–42, 2003.
40. P. Rogaway and M. Bellare. Robust computational secret sharing and a unified account of classical secret-sharing goals. In *ACM CCS 2007*, pages 172–184, 2007.
41. A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
42. K. Simoens, P. Tuyls, and B. Preneel. Privacy weaknesses in biometric sketches. In *2009 30th IEEE Symposium on Security and Privacy*, pages 188–203, 2009.
43. E. Stefanov, C. Papamanthou, and E. Shi. Practical dynamic searchable encryption with small leakage. In *NDSS 2014*. The Internet Society, 2014.
44. X. Yi, F. Hao, L. Chen, and J. K. Liu. Practical threshold password-authenticated secret sharing protocol. In *ESORICS 2015*, volume 9326 of *LNCS*, pages 347–365. Springer, 2015.
45. Y. Zhang, J. Katz, and C. Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In *USENIX Security 2016*, pages 707–720, 2016.