# Secure Remote Authentication Using Biometrics

Xavier Boyen* Yevgeniy Dodis† Jonathan Katz‡ Rafail Ostrovsky§
Adam Smith¶

**Abstract**

Biometrics offer a potential source of high-entropy, secret information. Before such data can be used in cryptographic protocols, however, two issues must be addressed: biometric data (1) are not uniformly distributed, and (2) are not exactly reproducible. Recent work, most notably that of Dodis, Reyzin, and Smith, has shown how these obstacles may be overcome using public information which is reliably sent from a server to the (human) user. Subsequent work of Boyen has shown how to extend these techniques — in the random oracle model — to enable unidirectional authentication from the user to the server without the assumption of a reliable channel.

Here, we show two efficient techniques enabling the use of biometric data to achieve *mutual* authentication/authenticated key exchange over a completely insecure (i.e., adversarially controlled) channel. In addition to achieving stronger security guarantees than the above-mentioned work of Boyen, we improve upon his solution in a number of other respects: we tolerate a broader class of errors and (in one case) improve upon the parameters of his solution and give a proof of security in the standard model.

## 1 Using Biometric Data for Secure Authentication

Biometric data — which offers a potential source of high-entropy, secret information — has been suggested as a way to enable strong, cryptographically secure authentication of human users without requiring them to remember or store traditional cryptographic keys.[1] Before such data can be used in existing cryptographic protocols, however, two issues must be addressed: first, biometric data are *not uniformly distributed* and hence will not guarantee "security" (at least not in any provable sense) if used as-is, say, as a key for a pseudorandom function. While the problem of non-uniformity can be addressed using a hash function (viewed either as a random oracle [2] or as a strong extractor [19]), the second and more difficult problem is that biometric data are *not exactly reproducible* (as two biometric scans of the same feature are rarely identical); hence, traditional protocols will not even guarantee correctness when the parties use a shared secret generated from biometric data.

Much work has focused on addressing the aforementioned problems in an attempt to develop secure techniques for biometric authentication [8, 15, 18, 14, 20]. Most recently, Dodis, Reyzin, and

*Voltage Security. xb@boyen.org.

†Department of Computer Science, New York University. dodis@cs.nyu.edu

‡Department of Computer Science, University of Maryland. jkatz@cs.umd.edu. Work supported by NSF Trusted Computing grant #0310751.

§Department of Computer Science, UCLA. rafail@cs.ucla.edu

¶Wiezmann Institute of Science. adam.smith@weizmann.ac.il.

[1] Although other cryptographic applications of biometric data are certainly possible, the application to user authentication seems most natural and is the one on which we focus here.

Smith [9] studied how to use biometric data to securely derive cryptographic keys for use in a general context, and thus, in particular, for the purposes of authentication. Roughly speaking (see Section 2 for formal definitions), they introduce two primitives: a *secure sketch* which allows recovery of a shared secret from any value "close" to this secret, and a *fuzzy extractor* which extracts a uniformly distributed random string $s$ from this shared secret in an error-tolerant manner. Both primitives rely on a "public" string $\mathsf{pub}$ which is stored by the server and transmitted to the user; loosely speaking, $\mathsf{pub}$ encodes the redundancy needed for error-tolerant reconstruction. The primitives are designed so as to be "secure" even when an adversary learns the value of this public string.

Unfortunately, although these primitives suffice to obtain security in the presence of an eavesdropping adversary who may learn $\mathsf{pub}$ when it is sent to the user — or by passively monitoring the server's storage — the work of Dodis, *et al.* does not address the issue of malicious adversarial modification of $\mathsf{pub}$, either in the case when $\mathsf{pub}$ is transmitted over an insecure network or when an adversary might tamper with the server's storage. As a consequence, their work does not provide a method for secure authentication in the presence of an *active* adversary who may modify the messages sent between the two parties. Indeed, depending on the specific construction used, attempting to use a maliciously-altered public string with one's biometric could result in the exposure of information related to one's biometric secret. A "solution" is for the user to store $\mathsf{pub}$ himself rather than obtain it from the server (or to authenticate $\mathsf{pub}$ using a certificate chain), but this defeats the purpose of using biometrics in the first place: namely, to avoid the need for the user to store *any* additional cryptographic information (even if it need not be kept secret).

Boyen [5], *inter alia*, partially addresses the issue of adversarial modification of $\mathsf{pub}$ (although the main focus of his work is the orthogonal issue of re-using biometrics with multiple servers, which we do not explicitly address here). The main disadvantage of applying his technique in our context is that it provides only *unidirectional* authentication from the user to the server. Indeed, his approach cannot be used to achieve authentication of the server to the user since his definition of "insider security" (cf. [5, Section 5.2]) does not preclude an adversary from knowing the (incorrect) value $s'$ recovered by the user when the adversary forwards a modified $\mathsf{pub}'$ to this user; if the adversary knows $s'$, then from the viewpoint of the user the adversary can do anything the server could do, and hence authentication of the server to the user is impossible. The lack of mutual authentication implies that — when communicating over an insecure network — the user and server cannot securely establish a shared session key with which to encrypt and authenticate future messages: the user may unwittingly share a key with an adversary who can then decrypt any data sent by the user as well as authenticate arbitrary data of the adversary's choice.

### 1.1 Our Contributions

Here, we provide the first full solution to the problem of secure remote authentication using biometric data: in particular, we show techniques allowing for mutual authentication and/or authenticated key exchange over a completely insecure channel. We show two constructions: the first is a generic solution which protects against modification of the public value $\mathsf{pub}$ in any context in which secure sketches/fuzzy extractors are used. Thus, this solution serves as a "drop-in" replacement which "compiles" any protocol which is secure when $\mathsf{pub}$ is assumed to be transmitted reliably into one which is secure even when $\mathsf{pub}$ might be tampered with (we do not formalize this notion, but rather view it as an intuitive way to understand our results). Our second construction is specific to the setting of remote authentication/key exchange, and improves upon our first solution in this case. In addition to enabling mutual authentication, our constructions enjoy the following additional

advantages compared to the work of Boyen [5]:

- Our solutions tolerate a stronger class of errors than those considered by Boyen. In particular, Boyen's work only allows for *data-independent* errors, whereas our analysis handles *arbitrary* (bounded) errors. We remark that small yet data-dependent errors seem natural in the context of biometrics.
- Our second solution is proven secure in the standard model.
- Our second solution achieves improved bounds on the entropy loss. For practical choices of the parameters, this results in an improvement of roughly 128 bits of entropy. This is particularly important since biometrics have relatively low entropy to begin with.[2]

**Organization.** We review some basic definitions as well as the primitives of Dodis, *et al.* in Section 2. In Section 3 we introduce the notion of *robust* sketches/fuzzy extractors which are resilient — in a very strong way — to any modification of the public value, and can be used as a generic replacement for the sketches/fuzzy extractors of [9]. Our second solution, which is specific to the problem of using biometrics for authentication and offers some advantages with respect to our generic construction, is described in Section 4.

## 2 Definitions

Unless explicitly stated otherwise, all logarithms are base 2. We let $U_\ell$ denote the uniform distribution over $\ell$-bit strings. A (discrete) *metric space* is a finite set $\mathcal{M}$ equipped with a symmetric distance function $d : \mathcal{M} \times \mathcal{M} \to \mathbb{Z}^+ \cup \{0\}$ satisfying the triangle inequality and such that $d(x, y) = 0 \Leftrightarrow x = y$. (All metric spaces considered in this work will be discrete.) For the application to biometrics, we assume that the format of the biometric data is such that it forms a metric space under some appropriate distance function. We will not need to specify any particular metric space in our work, as our constructions build in a generic way on any sketches/fuzzy extractors constructed over any such space (e.g., those constructed in [9] for a variety of metrics). If $(\Omega, P)$ is a probability space over which random variables $W, W'$ (taking values in a metric space $\mathcal{M}$) are defined, then we say $d(W, W') \le t$ if for all $\omega \in \Omega$ it holds that $d(W(\omega), W'(\omega)) \le t$.

Given a metric space $(\mathcal{M}, d)$ and a point $x \in \mathcal{M}$ we define $\mathsf{Vol}_t^{\mathcal{M}}(x) \stackrel{\text{def}}{=} |\{x' \in \mathcal{M} \mid d(x, x') \le t\}|$ and $\mathsf{Vol}_t^{\mathcal{M}} \stackrel{\text{def}}{=} \max_{x \in \mathcal{M}}\{\mathsf{Vol}_t^{\mathcal{M}}(x)\}$. The latter is simply the maximum number of points in any "ball" of radius $t$ in the given metric space.

The *min-entropy* $H_\infty(A)$ of a random variable $A$ is defined as $-\log(\max_a \Pr[A = a])$. Following [9], for a pair of random variables $A$ and $B$, we define the *average min-entropy of $A$ given $B$* as $\bar{H}_\infty(A|B) \stackrel{\text{def}}{=} -\log\left(\mathrm{Exp}_{b \leftarrow B}\left[2^{-H_\infty(A|B=b)}\right]\right)$. The *statistical difference* between $A$ and $B$ over the same domain $D$ is defined as $\mathbf{SD}(A, B) \stackrel{\text{def}}{=} \frac{1}{2}\sum_{v \in D} |\Pr[A = d] - \Pr[B = d]|$.

### 2.1 Secure Sketches and Fuzzy Extractors

We review the definitions from [9] using slightly different terminology. Recall from the introduction that a secure sketch provides a way to recover a shared secret $w$ from any value $w'$ which is "close" to $w$. More formally:

[2] For instance, estimates for iris scans range from 173 bits [8] to 250 bits [13] of entropy per eye.

**Definition 1** An $(m, m', t)$-*secure sketch* over a metric space $(\mathcal{M}, d)$ is a *sketching procedure* $\mathsf{SS} : \mathcal{M} \to \{0,1\}^*$ along with a *recovery procedure* $\mathsf{Rec}$, such that:

*Security:* For all random variables $W$ over $\mathcal{M}$ with $H_\infty(W) \geq m$, we have $\bar{H}_\infty(W \mid SS(W)) \geq m'$.

*Error tolerance:* For all $w, w' \in \mathcal{M}$ with $d(w, w') \leq t$ we have $\mathsf{Rec}(w', SS(w)) = w$. ♢

While secure sketches address the issue of error correction, they do not address the issue of the possible non-uniformity of $W$. Fuzzy extractors, defined next, correct for this.

**Definition 2** An $(m, \ell, t, \delta)$-*fuzzy extractor* over a metric space $(\mathcal{M}, d)$ consists of an *extraction algorithm* $\mathsf{Ext} : \mathcal{M} \to \{0,1\}^\ell \times \{0,1\}^*$ and a *recovery procedure* $\mathsf{Rec}$ such that:

*Security:* For all random variables $W$ over $\mathcal{M}$ with $H_\infty(W) \geq m$, if $\langle R, \mathsf{pub}\rangle \leftarrow \mathsf{Ext}(W)$ then $\mathbf{SD}(\langle R, \mathsf{pub}\rangle, \langle U_\ell, \mathsf{pub}\rangle) \leq \delta$.

*Error tolerance:* For all $w, w' \in \mathcal{M}$ with $d(w, w') \leq t$, if $\langle R, \mathsf{pub}\rangle \leftarrow \mathsf{Ext}(w)$ then it is the case that $\mathsf{Rec}(w', \mathsf{pub}) = R$. ♢

As shown in [9, Lemma 3.1], it is easy to construct a fuzzy extractor over a metric space $(\mathcal{M}, d)$ given any secure sketch defined over the same space, by applying a (standard) strong randomness extractor [19] and including the (randomly chosen) "key" of the extractor as part of $\mathsf{pub}$. Starting with an $(m, m', t)$-secure sketch and with an appropriate choice of extractor, this yields an $(m, m' - 2\log(\frac{1}{\delta}), t, \delta)$-fuzzy extractor.

## 2.2 Modeling Error in Biometric Applications

As error correction is a key motivation for our work, it is necessary to develop some formal model of the types of errors that may occur. In prior work by Boyen [5], the error in various biometric readings was assumed to be under adversarial control but with the restriction that the adversary could only specify data-*independent* errors (e.g., constant shifts, permutations, etc.). It is not clear that this is a realistic model in practice, as one certainly expects, say, portions of the biometric where "features" are present to be more susceptible to error.

Here, we consider a much more general error model where the errors may be data-*dependent* and hence correlated with each other and even with the (secret) biometric itself. Furthermore, as we are ultimately interested in modeling "nature" (as manifested in the physical processes that cause fluctuations in the biometric measurements), we do not even require that the errors be efficiently computable. The only restriction we make is that the errors are "small" and, in particular, less than the desired error-correction bound; since the error-correction bound in any real-world application should ensure correctness with high probability, this restriction seems reasonable. Formally:

**Definition 3** A $t$-*bounded distortion ensemble* $\mathcal{W} = \{W_i\}_{i=0,\ldots}$ is a sequence of random variables $W_i : \Omega \to \mathcal{M}$ such that for all $i$ we have $d(W_0, W_i) \leq t$. We refer to $W_0$ as the *original* variable. ♢

For our application, $W_0$ will represent the biometric reading obtained when a user initially registers with a server, and $W_i$ will represent the biometric reading upon subsequent authentication attempts by this user. Note that mutual authentication fails if an adversary can guess $W_i$ for some $i > 0$. Luckily, the following lemmas give bounds on this probability. First, we show that the min-entropy of each $W_i$ is, at worst, $\log \mathsf{Vol}_t^{\mathcal{M}}$ bits less than that of $W_0$. Moreover, when publishing $SS(W_0)$ using a secure sketch, we show that $W_i$ is in fact no easier to guess than $W_0$.

**Lemma 1** *Let $W_0$ and $W_1$ be two random variables over $\mathcal{M}$ satisfying $d(W_0, W_1) \leq t$, and let $B$ be an arbitrary random variable. Then*

$$\bar{H}_\infty(W_1 \mid B) \geq \bar{H}_\infty(W_0 \mid B) - \log \mathsf{Vol}_t^{\mathcal{M}}.$$

**Proof** Fix $x \in \mathcal{M}$ and any outcome $B = b$. Since $d(W_0, W_1) \leq t$, we have $\Pr[W_1 = x \mid B = b] \leq \sum_{x' | d(x,x') \leq t} \Pr[W_0 = x' \mid B = b] \leq \mathsf{Vol}_t^{\mathcal{M}} \cdot 2^{-H_\infty(W_0 | B=b)}$, which means $H_\infty(W_1 \mid B = b) \geq H_\infty(W_0 \mid B = b) - \log \mathsf{Vol}_t^{\mathcal{M}}$. Since this holds for every $b$, the lemma follows. ■

Secure sketches imply the following, stronger form of Lemma 1. It states that points close to $W_0$ cannot be easier to guess than $W_0$ if the adversary knows the value of the sketch.

**Lemma 2** *Let $W_0$ and $W_1$ be two random variables over $\mathcal{M}$ satisfying $d(W_0, W_1) \leq t$, and let $B$ be an arbitrary random variable. Then*

$$\bar{H}_\infty(W_1 \mid SS(W_0), B) \geq \bar{H}_\infty(W_0 \mid SS(W_0), B).$$

**Proof** Notice that since $d(W_0, W_1) \leq t$, we have $\mathsf{Rec}(W_1, SS(W_0)) = W_0$, which means that if for some $x, b, \mathsf{pub}$ we have $\Pr(W_1 = x \mid SS(W_0) = \mathsf{pub}, B = b) \geq \alpha$, then $\Pr(W_0 = \mathsf{Rec}(x, \mathsf{pub}) \mid SS(W_0) = \mathsf{pub}, B = b) \geq \alpha$ as well. Since this holds for all $x, b$ and $\mathsf{pub}$, the lemma follows. ■

The analogue of Lemma 2 for fuzzy extractors holds as well.

## 3 Robust Sketches and Fuzzy Extractors

Recall that a secure sketch, informally, takes a secret $w$ and returns some value $\mathsf{pub}$ which allows recovery of $w$ given any $w'$ "close" to $w$. When $\mathsf{pub}$ is transmitted to a user over an insecure network, however, an adversary might modify $\mathsf{pub}$ in transit. In this section, we define the notion of a *robust* sketch, which protects against this sort of attack in a very strong way: with high probability, the user will detect whether $\mathsf{pub}$ has been modified and can thus immediately abort in this case. A robust fuzzy extractor is defined similarly. We then show: (1) a construction of a robust sketch in the random oracle model (starting from any secure sketch), and (2) a conversion from any robust sketch to a robust fuzzy extractor (this conversion works in the standard model). We conclude this section by showing the immediate application of robust fuzzy extractors to the problem of mutual authentication.

We first define a slightly stronger notion of a secure sketch:

**Definition 4** An $(m, m', t)$-secure sketch $(\mathsf{SS}, \mathsf{Rec})$ is said to be *well-formed* if it satisfies the conditions of Definition 1 except for the following modifications: (1) $\mathsf{Rec}$ may now return either an element in $\mathcal{M}$ or the distinguished symbol $\perp$, and (2) for all $w' \in \mathcal{M}$ and arbitrary $\mathsf{pub}'$, if $\mathsf{Rec}(w', \mathsf{pub}') \neq \perp$ then $d(w', \mathsf{Rec}(w', \mathsf{pub}')) \leq t$. ♢

It is straightforward to transform any secure sketch $(\mathsf{SS}, \mathsf{Rec})$ into a secure sketch $(\mathsf{SS}, \mathsf{Rec}')$ which is well-formed: $\mathsf{Rec}'$ runs $\mathsf{Rec}$ and then verifies that its output is within distance $t$ of the input value. If yes, it returns this value; otherwise, it outputs $\perp$.

We now define the notion of a *robust* sketch:

**Definition 5** Given algorithms $(\mathsf{SS}, \mathsf{Rec})$ and random variables $\mathcal{W} = \{W_0, W_1, \ldots, W_n\}$ over metric space $(\mathcal{M}, d)$, consider the following game between an adversary $\mathcal{A}$ and a challenger: Let

$w_0$ (resp., $w_i$) be the value assumed by $W_0$ (resp., $W_i$). The challenger computes $\mathsf{pub} \leftarrow \mathsf{SS}(w_0)$ and gives $\mathsf{pub}$ to $\mathcal{A}$. Next, for $i = 1, \ldots, n$, the challenger and $\mathcal{A}$ proceed as follows: $\mathcal{A}$ outputs $\mathsf{pub}_i \neq \mathsf{pub}$ and is given $\mathsf{Rec}(w_i, \mathsf{pub}_i)$ in return. If there exists an $i$ such that $\mathsf{Rec}(w_i, \mathsf{pub}_i) \neq \perp$ we say the adversary *succeeds* and this event is denoted by $\mathsf{Succ}$.

We say $(\mathsf{SS}, \mathsf{Rec})$ is an $(m, m'', n, \varepsilon, t)$-*robust sketch* (over $(\mathcal{M}, d)$) if it is a well-formed $(m, \cdot, t)$-secure sketch and: (1) for all $t$-bounded distortion ensembles $\mathcal{W}$ with $H_\infty(W_0) \geq m$ and all adversaries $\mathcal{A}$ we have $\Pr[\mathsf{Succ}] \leq \varepsilon$; and (2) the average min-entropy of $W_0$ — conditioned on the entire view of $\mathcal{A}$ throughout the above game — is at least $m''$.[3] $\diamondsuit$

We remark that a simpler definition would be to consider only random variables $\{W_0, W_1\}$ and to have $\mathcal{A}$ only output a single value $\mathsf{pub}_1 \neq \mathsf{pub}$. A standard hybrid argument would then imply the above definition with $\varepsilon$ increased by a multiplicative factor of $n$. We have chosen to work with the more general definition above as it allows for a tighter concrete security analysis. Also, although the above definition considers all-powerful adversaries, we will focus our attention on security against (computationally unbounded) adversaries whose queries to a random oracle are limited.

We now construct the robust sketch $(\mathsf{SS}, \mathsf{Rec})$ from any well-formed secure sketch $(\mathsf{SS}^*, \mathsf{Rec}^*)$. In what follows, $H : \{0,1\}^* \to \{0,1\}^k$ is modeled as a random oracle.

| $\underline{\mathsf{SS}(w)}$ | $\underline{\mathsf{Rec}(w, \mathsf{pub} = \langle \mathsf{pub}^*, h \rangle)}$ |
|---|---|
| $\mathsf{pub}^* \leftarrow \mathsf{SS}^*(w)$ | $w' = \mathsf{Rec}^*(w, \mathsf{pub}^*)$ |
| $h = H(w, \mathsf{pub}^*)$ | if $w' = \perp$ output $\perp$ |
| return $\mathsf{pub} = \langle \mathsf{pub}^*, h \rangle$ | if $H(w', \mathsf{pub}^*) \neq h$ output $\perp$ |
| | otherwise, output $w'$ |

**Theorem 1** *If $(\mathsf{SS}^*, \mathsf{Rec}^*)$ is a well-formed $(m, m', t)$-secure sketch over metric space $(\mathcal{M}, d)$ and $H$ is a random oracle with $k$ bits of output, then $(\mathsf{SS}, \mathsf{Rec})$ is an $(m, m'', n, \varepsilon, t)$-robust sketch over $(\mathcal{M}, d)$ for any adversary making at most $q_h$ queries to the random oracle, where*

$$\begin{aligned} \varepsilon &= (q_h^2 + n) \cdot 2^{-k} + (3q_h + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \cdot 2^{-m'} \\ m'' &= m' - \log\left((q_h^2 + n) \cdot 2^{m'-k} + (3q_h + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}})\right) \end{aligned}$$

*When $k \geq m' + \log q_h$ (which can be enforced in practice), this simplifies to $\varepsilon \leq (4q_h + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \cdot 2^{-m'}$ and $m'' \geq m' - \log(4q_h + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}})$.*

**Proof (Sketch)** It is easy to see that $(\mathsf{SS}, \mathsf{Rec})$ is an $(m, \cdot, t)$-secure sketch and thus we only need to prove the latter two conditions of Definition 5. In order to provide intuition, the following proof is somewhat informal; however, the arguments given here can easily be formalized. Let $\mathsf{pub} = \langle \mathsf{pub}^*, h \rangle$ denote the value output by $\mathsf{SS}$ in an execution of the game described in Definition 5. Note that if $\mathcal{A}$ ever outputs $\mathsf{pub}_i = \langle \mathsf{pub}_i^*, h_i \rangle$ with $\mathsf{pub}_i^* = \mathsf{pub}^*$ then the response is always $\perp$ (since then we must have $h_i \neq h$ and so $\mathsf{Rec}$ will output $\perp$). Thus, we simply assume that $\mathsf{pub}_i^* \neq \mathsf{pub}^*$.

Fix a $t$-bounded distortion ensemble $\{W_0, W_1, \ldots, W_n\}$ with $H_\infty(W_0) \geq m$. For any output $\mathsf{pub}_i = \langle \mathsf{pub}_i^*, h_i \rangle$ of $\mathcal{A}$, define the random variable $W_i' \stackrel{\text{def}}{=} \mathsf{Rec}^*(W_i, \mathsf{pub}_i^*)$. In order not to complicate notation, we let $H_\infty(W_i') \stackrel{\text{def}}{=} -\log\left(\max_{x \in \mathcal{M}} \Pr[W_i' = x]\right)$; i.e., we ignore the probability that $W_i' = \perp$ since $\mathcal{A}$ does not succeed in such a case. $\bar{H}_\infty(W_i' \mid X)$, for a random variable $X$, is defined similarly. Let $w_0, w_i$, and $w_i'$ denote the values taken by the random variables $W_0, W_i, W_i'$, respectively.

[3] In particular, this implies that $(\mathsf{SS}, \mathsf{Rec})$ is an $(m, m'', t)$-secure sketch.

We classify the random oracle queries of $\mathcal{A}$ into two types: *type 1* queries are those of the form $H(\cdot, \mathsf{pub}^*)$, and *type 2* queries are all the others. Informally, type 1 queries represent attempts by $\mathcal{A}$ to learn the value of $w_0$ (in particular, if $\mathcal{A}$ finds $w$ such that $H(w, \mathsf{pub}^*) = h$ then it is "likely" that $w_0 = w$), while type 2 queries represent attempts by $\mathcal{A}$ to determine an appropriate value for some $h_i$ (i.e., if $\mathcal{A}$ "guesses" that $w_i' = w$ for a particular choice of $\mathsf{pub}_i^*$ then a "winning" strategy is for $\mathcal{A}$ to obtain $h_i = H(w, \mathsf{pub}_i^*)$ and output $\mathsf{pub}_i = \langle \mathsf{pub}_i^*, h_i \rangle$).

Without loss of generality, we assume that $\mathcal{A}$ makes all its type 1 queries first, then makes all its type 2 queries, and finishes by making all its $n$ recovery queries non-adaptively. The first assumption is legitimate since the oracle answers to type 1 and type 2 queries (as well as the responses to them) are independent from each other, and can thus be safely re-ordered. Further, the adversary should not expect to gain any information whatsoever from the challenger responses $\mathsf{Rec}(W_i, \mathsf{pub}_i)$ — i.e., it must expect all of these to take the value $\perp$ — since as soon as this condition fails the adversary succeeds and thus never needs to actually *use* that information. This also justifies why the recovery queries can be made in parallel, and after all the random oracle queries.

Let $\mathcal{Q}_1$ (resp., $\mathcal{Q}_2$) be a random variable denoting the sequence of type 1 (resp., type 2) queries made by $\mathcal{A}$, and let $q_1$ (resp., $q_2$) denote the value it assumes. For some fixed value of $\mathsf{pub}$, define $\gamma_{\mathsf{pub}} \stackrel{\text{def}}{=} H_\infty(W_0|\mathsf{pub})$. Notice, since $(\mathsf{SS}^*, \mathsf{Rec}^*)$ is an $(m, m', t)$-secure sketch, we have $\mathrm{Exp}_{\mathsf{pub}}[2^{-\gamma_{\mathsf{pub}}}] \leq 2^{-m'}$. Now, define $\gamma'_{\mathsf{pub},q_1} \stackrel{\text{def}}{=} H_\infty(W_0 \mid \mathsf{pub}, q_1)$, and let us call the value $q_1$ "bad" if $\gamma'_{\mathsf{pub},q_1} \leq \gamma_{\mathsf{pub}} - 1$. We consider two cases: If $2^{\gamma_{\mathsf{pub}}} \leq 2q_h$ we will not have any guarantees, but luckily Markov's inequality implies that $\Pr[2^{\gamma_{\mathsf{pub}}} \leq 2q_h] = \Pr[2^{-\gamma_{\mathsf{pub}}} \geq 2^{-m'} \cdot (2^{m'}/2q_h)] \leq 2q_h \cdot 2^{-m'}$. Otherwise, if $2^{\gamma_{\mathsf{pub}}} \geq 2q_h$, we observe that the type 1 queries of $\mathcal{A}$ may be viewed as guesses of $w_0$. In fact, it is easy to see that we only improve the success probability of $\mathcal{A}$ if in response to a type 1 query $H(w, \mathsf{pub}^*)$ we simply tell $\mathcal{A}$ whether $w_0 = w$ or not.[4] It is immediate that $\mathcal{A}$ learns the correct value of $w_0$ with probability at most $q_h \cdot 2^{-\gamma_{\mathsf{pub}}}$. Moreover, when this does *not* happen, $\mathcal{A}$ has eliminated at most $q_h \leq 2^{\gamma_{\mathsf{pub}}}/2$ (out of at least $2^{\gamma_{\mathsf{pub}}}$) possibilities for $w_0$, which means that $\gamma'_{\mathsf{pub},q_1} \geq \gamma_{\mathsf{pub}} - 1$, i.e. that $q_1$ is "good". Therefore, the probability that $q_1$ is "bad" in this second case is at most $q_h \cdot 2^{-\gamma_{\mathsf{pub}}}$.

Combining the above two arguments, we see that

$$\begin{aligned} \mathrm{Exp}_{\mathsf{pub}}[\Pr[q_1 \text{ bad}]] &\leq \Pr_{\mathsf{pub}}[2^{\gamma_{\mathsf{pub}}} \leq 2q_h] + \mathrm{Exp}_{\mathsf{pub}}[q_h \cdot 2^{-\gamma_{\mathsf{pub}}}] \\ &\leq 2q_h \cdot 2^{-m'} + q_h \cdot 2^{-m'} \;=\; 3q_h \cdot 2^{-m'}. \end{aligned} \tag{1}$$

Next, define $\gamma''_{\mathsf{pub},q_1} \stackrel{\text{def}}{=} \max_i(H_\infty(W_i' \mid \mathsf{pub}, q_1))$. Recall that $\{W_0, W_1, \ldots\}$ is a $t$-bounded distortion ensemble which means $d(W_0, W_i) \leq t$. Furthermore, since $(\mathsf{SS}^*, \mathsf{Rec}^*)$ is well-formed, $\{W_i, W_i'\}$ is also a $t$-bounded distortion ensemble[5] regardless of $\mathsf{pub}_i^*$, which means $d(W_i, W_i') \leq t$. Applying Lemma 2 on $(W_0, W_i)$ (and noticing that $\mathsf{pub}$ contains $\mathsf{pub}^*$) followed by Lemma 1 on $(W_i, W_i')$, we find that

$$\gamma''_{\mathsf{pub},q_1} \;\geq\; \max_i(H_\infty(W_i \mid \mathsf{pub}, q_1)) - \log \mathsf{Vol}_t^{\mathcal{M}} \;\geq\; \gamma'_{\mathsf{pub},q_1} - \log \mathsf{Vol}_t^{\mathcal{M}}. \tag{2}$$

We now consider the type 2 queries of $\mathcal{A}$. Clearly, the answers to these queries do not affect the conditional min-entropies of $W_i'$ (since these queries do not include $\mathsf{pub}^*$), so the best probability

[4] This has no effect when $H(w, \mathsf{pub}^*) \neq h$ as then $\mathcal{A}$ learns anyway that $w \neq w_0$. The modification has a small (but positive) effect on the success probability of $\mathcal{A}$ when $H(w, \mathsf{pub}^*) = h$ since this fact by itself does not definitively guarantee that $w = w_0$.

[5] Ignoring the case when $W_i' = \perp$; see the definition of $H_\infty(W_i')$ given earlier.

for the attacker to predict any of the $W'_i$ is still given by $2^{-\gamma''_{\mathsf{pub},q_1}}$ (for fixed $\mathsf{pub}$ and $q_1$). Assume now for a second that there are no collisions in the outputs of type 2 queries, and consider the recovery query $\langle \mathsf{pub}^*_i, h_i \rangle$. The chance that this query will be accepted is at most the probability that $\mathcal{A}$ asked some type 2 query $H(w'_i, \cdot)$ for the correct $w'_i$ (to which the answer was $h_i$) plus the probability that such query was not asked yet $\mathcal{A}$ nevertheless managed to predict the value $H(w'_i, \mathsf{pub}^*_i)$ by sheer luck. Clearly, the second case happens with probability at most $2^{-k}$. As for the first case, every $h_i$ will have at most one value $w$ for which the adversary got $H(w, \mathsf{pub}^*) = h_i$. Thus, the best chance of the adversary is to hope that this single $w$ is equal to the correct value $w'_i$. And we just argued that irrespective of $\mathsf{pub}^*_i$, this probability is at most $2^{-\gamma''_{\mathsf{pub},q_1}}$. Therefore, assuming no collisions happened in type 2 queries, the success probability of $\mathcal{A}$ in any one of the $n$ (non-adaptive) queries is at most $n \cdot (2^{-\gamma''_{\mathsf{pub},q_1}} + 2^{-k})$. Furthermore, by the birthday bound the probability of a collision is at most $q_h^2/2^k$. Therefore, conditionally on $\mathsf{pub}$ and $q_1$, and for the corresponding value of $\gamma''_{\mathsf{pub},q_1}$, we find that $\Pr[\mathsf{Succ} \mid \mathsf{pub}, q_1] \le n \cdot 2^{-\gamma''_{\mathsf{pub},q_1}} + (q_h^2 + n) \cdot 2^{-k}$.

The adversary's overall probability of success is thus bounded by the expectation, over $\mathsf{pub}$ and $q_1$, of this previous quantity; that is:

$$\begin{aligned}
\Pr[\mathsf{Succ}] &= \mathrm{Exp}_{\mathsf{pub},q_1} \Pr[\mathsf{Succ} \mid \mathsf{pub}, q_1] \\
&\le (q_h^2 + n) \cdot 2^{-k} + \mathrm{Exp}_{\mathsf{pub}} \left[ \Pr_{q_1 \leftarrow \mathcal{Q}_1}[q_1 \text{ bad} \mid \mathsf{pub}] + \sum_{q_1 \text{ good}} n \cdot 2^{-\gamma''_{\mathsf{pub},q_1}} \cdot \Pr[Q_1 = q_1 \mid \mathsf{pub}] \right].
\end{aligned}$$

Using Equation (2), we see that $2^{-\gamma''_{\mathsf{pub},q_1}} \le \mathsf{Vol}_t^{\mathcal{M}} \cdot 2^{-\gamma'_{\mathsf{pub},q_1}}$. Moreover, for good $q_1$ we have $\gamma'_{\mathsf{pub},q_1} \ge \gamma_{\mathsf{pub}} - 1$, which means that $2^{-\gamma''_{\mathsf{pub},q_1}} \le 2\mathsf{Vol}_t^{\mathcal{M}} \cdot 2^{-\gamma_{\mathsf{pub}}}$. Finally, using Equation (1), we have $\mathrm{Exp}_{\mathsf{pub}}[\Pr[q_1 \text{ bad} \mid \mathsf{pub}]] \le 3q_h \cdot 2^{-m'}$. Combining all these, we successively derive:

$$\begin{aligned}
\Pr[\mathsf{Succ}] &\le (q_h^2 + n) \cdot 2^{-k} + 3q_h \cdot 2^{-m'} + \mathrm{Exp}_{\mathsf{pub}} \left[ 2n \cdot \mathsf{Vol}_t^{\mathcal{M}} \cdot 2^{-\gamma_{\mathsf{pub}}} \cdot \Pr_{q_1 \leftarrow \mathcal{Q}_1}[q_1 \text{ good}] \right] \\
&\le (q_h^2 + n) \cdot 2^{-k} + 3q_h \cdot 2^{-m'} + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}} \cdot \mathrm{Exp}_{\mathsf{pub}} \left[ 2^{-\gamma_{\mathsf{pub}}} \right] \\
&\le (q_h^2 + n) \cdot 2^{-k} + (3q_h + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \cdot 2^{-m'} \quad = \quad \varepsilon.
\end{aligned}$$

As for the claimed value of $m''$, we omit the details (since they follow almost the same argument as above), only outlining the main argument. As argued above, assuming $q_1$ is good, no collisions happen in type 2 queries, and the adversary did not manage to guess any of the values $H(w'_i, \mathsf{pub}^*_i)$, the conditional min-entropy of $W_0$ is at least $\gamma'_{\mathsf{pub},q_1} - \log(n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \ge \gamma_{\mathsf{pub}} - 1 - \log(n \cdot \mathsf{Vol}_t^{\mathcal{M}})$. On the other hand, all these bad event leading to a possibly smaller min-entropy of $W_0$ happen with (expected) probability (over $\mathsf{pub}$) at most $(q_h^2 + n) \cdot 2^{-k} + 3q_h \cdot 2^{-m'}$. From this, it is easy to see that if $\mathsf{View}$ represents the adversary's view in the experiment, then

$$\begin{aligned}
\bar{H}_\infty(W_0 | \mathsf{View}) &\ge -\log\left( ((q_h^2 + n) \cdot 2^{-k} + 3q_h \cdot 2^{-m'}) \cdot 1 + 1 \cdot \mathrm{Exp}_{\mathsf{pub}} \left[ 2n \cdot \mathsf{Vol}_t^{\mathcal{M}} \cdot 2^{-\gamma_{\mathsf{pub}}} \right] \right) \\
&\ge -\log\left( (q_h^2 + n) \cdot 2^{-k} + (3q_h + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \cdot 2^{-m'} \right) \\
&= m' - \log\left( (q_h^2 + n) \cdot 2^{m'-k} + (3q_h + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \right) \quad = \quad m''.
\end{aligned}$$

■

The bounds $\varepsilon$ and $m''$ that we derive in the above proof have a nice interpretation. The sub-expression $\left(q_h + n \cdot \mathsf{Vol}_t^{\mathcal{M}}\right)$ that appears (up to a small constant factor due to the analysis) in both expressions can be viewed as the number of points in the space $\mathcal{M}$ about which the adversary has obtained some information. The $q_h$ contribution is due to the type 1 oracle queries, each of which only reveals information about the queried point itself. Each of the $n$ queries to the challenger may cover no more than $\mathsf{Vol}_t^{\mathcal{M}}$ candidates for $w_0$, since in the worst case each such query eliminated one guess for $w_i'$ (unless collisions in type 2 queries happened), which in turn eliminated up to $\mathsf{Vol}_t^{\mathcal{M}}$ candidates for $w_i$, each of which can only eliminate one candidate $\mathsf{Rec}(w_i, \mathsf{pub}^*)$ for $w_0$. From there, $\varepsilon$ and $m''$ are easily interpreted as a combination of the above with a usual birthday collision bound arising from the random oracle and a small factor to account for the possibility that the adversary could guess the output to the random oracle.

In practice, it is easy to pick a hash function with sufficiently many output bits so that the expressions become simpler. In particular, the quantity $\max(q_h, n \cdot \mathsf{Vol}_t^{\mathcal{M}})$ will become the dominant factor determining the amount of the "loss" we get as compared with "non-robust" sketches.

### 3.1 From Robust Sketches to Robust Fuzzy Extractors

In a manner exactly analogous to the above, we may define the notion of a robust fuzzy extractor. We include the definition here since we will refer to it in the next subsection:

**Definition 6** Given algorithms $(\mathsf{Ext}, \mathsf{Rec})$ and random variables $\mathcal{W} = \{W_0, W_1, \ldots, W_n\}$ over a metric space $(\mathcal{M}, d)$, consider the following game between an adversary $\mathcal{A}$ and a challenger: Let $w_0$ (resp., $w_i$) be the value assumed by $W_0$ (resp., $W_i$). The challenger computes $(R, \mathsf{pub}) \leftarrow \mathsf{Ext}(w_0)$ and gives $\mathsf{pub}$ to $\mathcal{A}$. Next, for $i = 1, \ldots, n$, $\mathcal{A}$ outputs $\mathsf{pub}_i \neq \mathsf{pub}$ and is given $\mathsf{Rec}(w_i, \mathsf{pub}_i)$ in return. If there exists an $i$ such that $\mathsf{Rec}(w_i, \mathsf{pub}_i) \neq \perp$ we say the adversary *succeeds* and this event is denoted by $\mathsf{Succ}$.

We say $(\mathsf{Ext}, \mathsf{Rec})$ is an $(m, \ell, n, \varepsilon, t, \delta)$-*robust fuzzy extractor* (over $(\mathcal{M}, d)$) if the following hold for all $t$-bounded distortion ensembles $\mathcal{W}$ with $H_\infty(W_0) \geq m$:

*Robustness:* For all adversaries $\mathcal{A}$, it holds that $\Pr[\mathsf{Succ}] \leq \varepsilon$.

*Security:* Let $\mathsf{View}$ denote the entire view of $\mathcal{A}$ at the conclusion of the above game. Then, $\mathbf{SD}(\langle R, \mathsf{View}\rangle, \langle U_\ell, \mathsf{View}\rangle) \leq \delta$.

*Error-tolerance:* For all $w'$ with $d(w_0, w') \leq t$, we have $\mathsf{Rec}(w', \mathsf{pub}) = R$. ♢

By applying techniques almost exactly as in [9, Lemma 3.1] (with one slight subtlety; see Appendix A), we show a conversion from any robust sketch to a robust fuzzy extractor in the standard model. We include the details in Appendix A.

### 3.2 Application to Secure Authentication

The application of any robust fuzzy extractor to the problem of mutual authentication (or authenticated key exchange) over an insecure channel is immediate. Given any secure protocol $\Pi$ based on a uniformly distributed random shared key of length $\ell$, any $(m, \ell, n, \varepsilon, t, \delta)$-robust fuzzy extractor $(\mathsf{Ext}, \mathsf{Rec})$, and any source $W_0$ with $H_\infty(W_0) \geq m$, consider protocol $\Pi'$ constructed as follows:

**Initialization** The user samples $w_0$ according to $W_0$ (i.e., takes a scan of his biometric data) and computes $(R, \mathsf{pub}) \leftarrow \mathsf{Ext}(w_0)$. The user registers $(R, \mathsf{pub})$ at the server.

**Protocol execution** The $i^{\text{th}}$ time the user wants to run the protocol, the user will sample $w_i$ according to some distribution $W_i$ (i.e., the user re-scans his biometric data). The server sends $\mathsf{pub}$ to the user, who then computes $\hat{R} = \mathsf{Ext}(w_i, \mathsf{pub})$. If $\hat{R} = \perp$, the user immediately aborts. Otherwise, the user and server execute protocol $\Pi$, with the server and the user respectively using the keys $R$ and $\hat{R}$.

Assume that $\mathcal{W} = \{W_0, W_1, \ldots\}$ is a $t$-bounded distortion ensemble. Correctness of the above protocol is easily seen to hold: if the user obtains the correct value of $\mathsf{pub}$ from the server then, because $d(w_0, w_i) \leq t$, the user will recover $\hat{R} = R$ and thus both user and server will end up using the same key $R$ in the underlying protocol $\Pi$. Security of $\Pi'$ against an active adversary who may control all messages sent between the user and the server (see Appendix B for formal definitions of security) follows from the following observations:

- If the adversary forwards $\mathsf{pub}' \neq \mathsf{pub}$ to at most $n$ different user-instances, these instances will all abort immediately (without running $\Pi$) except with probability at most $\varepsilon$. Thus, roughly speaking, the adversary is essentially limited to forwarding the correct value of $\mathsf{pub}$.
- When the adversary forwards $\mathsf{pub}$ unchanged, the user and server run an execution of $\Pi$ using a key $R$ which is within statistical difference $\delta$ from a uniformly distributed $\ell$-bit key. Note that this is true even when conditioned on the view of the adversary in sessions when it does *not* forward $\mathsf{pub}$ unchanged (cf. Definition 6). Thus, assuming $\Pi$ is secure, the adversary will not succeed in "breaking" $\Pi'$ in this case either.

In terms of concrete security (informally), if the security of $\Pi$ against an adversary who executes at most $n$ sessions with the user and the server is $\varepsilon_\Pi$, then the security of $\Pi'$ is $\varepsilon + \delta + \varepsilon_\Pi$. A formal proof following the above intuition is straightforward, and will appear in the full version of this work.

# 4 An Improved Solution Tailored for Mutual Authentication

As discussed in the introduction, the robust sketches/fuzzy extractors described in the previous section provide a general mechanism for dealing with adversarial modification of the public value $\mathsf{pub}$ used in the constructions of Dodis, et al. [9]. In particular, taking any protocol based on secure sketches/fuzzy extractors which is secure when this public value is assumed not to be tampered with, and plugging in a *robust* sketch/fuzzy extractor, yields a protocol secure against an adversary who may either modify the contents of the server (e.g., if the server itself is malicious) or else modify the value of $\mathsf{pub}$ when it is sent to the user.

For specific problems of interest, however, it remains important to explore solutions which might improve upon the "general-purpose" solution described above. In this section, we show that for the case of mutual authentication/authenticated key exchange an improved solution is indeed possible. As compared to the generic solution based on robust fuzzy extractors (cf. Section 3.2 and Appendix A), the solution described here has the advantages that: (1) it is provably secure in the standard model, and (2) it achieves improved bounds on the "effective entropy loss". We provide an overview of our solution now.

Given the proof of Theorem 1, the intuition behind our current solution is actually quite straightforward. As in that proof, let $\mathcal{W} = \{W_0, \ldots\}$ be a sequence of random variables where $W_0$ represents

the initial recorded value of the user's biometric and $W_i$ denotes the $i^{\text{th}}$ scanned value of the biometric. Given a well-formed secure sketch $(\mathsf{SS}^*, \mathsf{Rec}^*)$ and a value $\mathsf{pub}_i^* \neq \mathsf{pub}^* = \mathsf{SS}^*(W_0)$ chosen by the adversary, let $W_i' \stackrel{\text{def}}{=} \mathsf{Rec}(W_i, \mathsf{pub}_i^*)$ and define the min-entropy of $W_i'$ as in the proof of Theorem 1. At a high level, Theorem 1 follows from the observations that (1) the average min-entropy of $W_i'$ is "high" for *any* value $\mathsf{pub}_i^*$; and (2) since the adversary succeeds only if it can also output a value $h_i = H(W_i', \mathsf{pub}_i^*)$ and $H$ is a random oracle, the adversary is (essentially) unable to succeed with probability better than $2^{-H_\infty(W_i')}$ in the $i^{\text{th}}$ iteration. Essential to the proof also is the fact that, except with "small" probability, the value $h = H(W_0, \mathsf{pub}^*)$ does not reduce the entropy of $W_0$ "very much" (again using the fact that $H$ is a random oracle with a limited number of queries).

The above suggests that another way to ensure that the adversary does not succeed with probability better than $2^{-H_\infty(W_i')}$ in any given iteration would be to have the user run an "equality test" using its recovered value $W_i'$. If this equality test is "secure" (in some appropriate sense we have not yet defined) then the adversary will effectively be reduced to simply guessing the value of $W_i'$, and hence its success probability in that iteration will be as claimed. Since we have already noted above that the average min-entropy of $W_i'$ is "high" (regardless of the value $\mathsf{pub}_i^*$ chosen by the adversary) when any well-formed secure sketch is used, this will be sufficient to ensure security of the protocol overall.

Thinking about what notion of security this "equality test" should satisfy, one realizes that it must be secure for arbitrary distributions on the user's secret value, and not just uniform ones. Also, the protocol must ensure that each interaction by the adversary corresponds to a guess of (at most) one possible value for $W_i'$. Finally, since the protocol is meant to be run over an insecure network, it must be "non-malleable" in some sense so that the adversary cannot execute a man-in-the-middle attack when the user and server are both executing the protocol. Finally, the adversary should not gain any information about the user's true secret $W_0$ (at least in a computational sense) after passively eavesdropping on multiple executions of the protocol. With the problem laid out in this way, it becomes clear that one possibility is to use a password-only authenticated key exchange (PAK) protocol [4, 1, 6] as the underlying "equality test".

Although the above intuition is appealing, we remark that a number of subtleties arise when trying to apply this idea to obtain a provably secure solution. In particular, we will require the PAK protocol to satisfy a slightly stronger definition of security than that usually considered for PAK (cf. [1, 6, 12]); informally, the PAK protocol should remain "secure" even when: (1) the adversary can dynamically add clients to the system, with (unique) identities chosen by the adversary; (2) the adversary can specify *non-uniform* and *dependent* password distributions for these clients; and (3) the adversary can specify such distributions *adaptively* at the time the client is added to the system. Luckily, it is not difficult to verify that existing solutions (e.g., [1, 16, 11]) satisfy[6] a definition of this sort. We provide a brief review of definitions for PAK, as well as the stronger definition of security required for our application, in Appendix C. Our definition may be of independent interest.

[6] In fact, it is already stated explicitly in [16, 11] that the given protocol(s) remain secure even under conditions (1) and (2), and it is not hard to see that they remain secure under condition (3) as well. We have not verified whether the protocols of, e.g., [6, 12] remain secure under the stated conditions but we expect that they do.

### 4.1 Our Construction

With the above in mind, we now describe our construction. Let $\Pi$ be a PAK protocol and let $(\mathsf{SS}, \mathsf{Rec})$ be a well-formed secure sketch. Construct a modified protocol $\Pi'$ as follows:

**Initialization** A user $U$ samples $w_0$ according to $W_0$ (i.e., takes a scan of his biometric data) and computes $\mathsf{pub} \leftarrow \mathsf{SS}(w_0)$. The user registers $(w_0, \mathsf{pub})$ at the server $S$.

**Protocol execution (server)** The server sends $\mathsf{pub}$ to the user. It then executes protocol $\Pi$ under the following parameters: it sets its own "identity" (within $\Pi$) to be $S \| \mathsf{pub}$, its "partner identity" to be $\mathsf{pid} = U \| \mathsf{pub}$, and the "password" to be $w_0$.

**Protocol execution (user)** The $i^{\text{th}}$ time the user executes the protocol, the user first samples $w_i$ according to distribution $W_i$ (i.e., the user re-scans his biometric data). The user also obtains a value $\mathsf{pub}'$ in the initial message it receives, and computes $w' = \mathsf{Rec}(w_i, \mathsf{pub}')$. If $w' = \perp$ then the user simply aborts. Otherwise, the user executes protocol $\Pi$, setting its own "identity" to $U \| \mathsf{pub}'$, its "partner identity" to $S \| \mathsf{pub}'$, and using the "password" $w'$.

It is easy to see that correctness holds, since if the user and the server interact without any interference from the adversary then: (1) the identity used by the server is equal to the partner ID of the user; (2) the identity of the user is the same as the partner ID of the server; and (3) the passwords $w_0$ and $w'$ are identical.

Before discussing the security of this protocol, we need to introduce a slight relaxation of the notion of a $t$-bounded distortion ensemble in which the various random variables in the ensemble are (efficiently) computable:

**Definition 7** Let $(\mathcal{M}, d)$ be a metric space. An *explicitly computable $t$-bounded distortion ensemble* is a sequence of boolean circuits $\mathcal{W} = \{W_0, \ldots\}$ and a parameter $\ell$ such that, for all $i$, the circuit $W_i$ computes a function from $\{0,1\}^\ell$ to $\mathcal{M}$ and, furthermore, for all $r \in \{0,1\}^\ell$ we have $d(W_0(r), W_i(r)) \leq t$. ♢

In our application $\mathcal{W}$ will be output by a PPT adversary, ensuring both that the ensemble contains only a polynomial number of circuits and that each such circuit is of polynomial size (and hence may be evaluated efficiently). We remark that it is *not* necessary for our proof that it be possible to efficiently verify whether a given $\mathcal{W}$ satisfies the "$t$-bounded" property or whether the min-entropy of $W_0$ is as claimed, although the security guarantee stated below only holds if $\mathcal{W}$ does indeed satisfy these properties.[7] With the above in mind, we now state the security achieved by our protocol (see Appendix B for a review of definitions of security for mutual authentication/authenticated key exchange protocols):

**Theorem 2** *Let $\Pi$ be a secure PAK protocol (with respect to the definition given in Appendix C) and let $\mathcal{A}$ be a* PPT *adversary. If $(\mathsf{SS}, \mathsf{Rec})$ is a well-formed $(m, m', t)$-secure sketch over a metric space $(\mathcal{M}, d)$, and $\mathcal{W} = \{W_0, \ldots\}$ is an explicitly-computable $t$-bounded distortion ensemble (output by $\mathcal{A}$) with $H_\infty(W_0) \geq m$, then the success probability of $\mathcal{A}$ in attacking protocol $\Pi'$ is at most $q_s \cdot 2^{-m''} + \mathsf{negl}(\kappa)$, where $q_s$ is the number of* Send *queries made by the adversary (cf. Appendix B) and $m'' = m' - \log \mathsf{Vol}_t^{\mathcal{M}}$.*

[7] As to whether the adversary can be "trusted" to output a $\mathcal{W}$ satisfying these properties, recall that $\mathcal{W}$ anyway is meant to model naturally occurring errors. Clearly, if a real-world adversary has the ability to, e.g., introduce arbitrarily large errors then only weaker security guarantees can be expected to hold.

Due to space limitations, the proof is given in Appendix D.

**Specific instantiations.** As noted earlier, a number of PAK protocols satisfying the required definition of security are known. If one is content to work in the random oracle model then the protocol of [1] — which is among the most efficient — may be used (note that this still represents an improvement over the solution based on robust fuzzy extractors since the "effective key size" will be larger, as we discuss in the next paragraph). To obtain a solution in the standard model which is only slightly less efficient, the PAK protocols of [16, 11] could be used.[8] Note that although these protocols were designed for use with "short" passwords, they can be easily modified to handle "large" passwords without much loss of efficiency; we discuss the specific case of the protocol by Katz, et al. [16] in Appendix E.

**Comparing our two solutions.** It is somewhat difficult to compare the security offered by our two solutions (i.e., the one based on robust fuzzy extractors and the one described in this section) since an exact comparison depends on a number of assumptions and design decisions. However, the most significant advantage of the present construction is that it avoids the need to apply a (standard) randomness extractor, and thus does not lose an additional $2 \log \delta^{-1}$ bits of entropy. Since a likely value in practice is $\delta \leq 2^{-64}$, this results in a "savings" of at least 128 bits of entropy. Further discussion, along with a more detailed comparison, will appear in the full version.

## References


[1] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. *Adv. in Cryptology — Eurocrypt 2000*, LNCS vol. 1807, Springer-Verlag, pp. 139–155, 2000.

[2] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. ACM CCS 1993.

[3] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. *Adv. in Cryptology — Crypto '93*, LNCS vol. 773, D.R. Stinson ed., Springer-Verlag, 1993, pp. 232–249.

[4] S. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. *IEEE Symposium on Research in Security and Privacy*, IEEE, pp. 72–84, 1992.

[5] X. Boyen. Reusable Cryptographic Fuzzy Extractors. ACM CCS 2004.

[6] V. Boyko, P. MacKenzie, and S. Patel. Provably-Secure Password-Authenticated Key Exchange Using Diffie-Hellman. *Adv. in Cryptology — Eurocrypt 2000*, LNCS vol. 1807, Springer-Verlag, pp. 156–171, 2000.

[7] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. Crypto 1998.


[8] Although these protocols require public parameters, such parameters can be "hard coded" into the implementation of the protocol and are fixed for all users of the system; thus, users are not required to remember or store these values. The difference is akin to the difference between PAK protocols in a "hybrid" PKI model (where clients store their server's public key) and PAK protocols (including [16, 11]) in which clients need remember only a short password.

[8] G. Davida, Y. Frankel, and B. Matt. On Enabling Secure Applications Through Off-Line Biometric Identification. IEEE Security and Privacy '98.

[9] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. Eurocrypt 2004.

[10] N. Frykholm and A. Juels. Error-Tolerant Password Recovery. ACM CCS 2001.

[11] R. Gennaro and Y. Lindell. A Framework for Password-Based Authenticated Key Exchange. *Adv. in Cryptology — Eurocrypt 2003*, LNCS vol. 2656, Springer-Verlag, pp. 524–543, 2003.

[12] O. Goldreich and Y. Lindell. Session-Key Generation Using Human Passwords Only. *Adv. in Cryptology — Crypto 2001*, LNCS vol. 2139, Springer-Verlag, pp. 408–432, 2001.

[13] A. Juels. Fuzzy Commitment. Slides from a presentation at the DIMACS workshop on *Cryptography: Theory Meets Practice*, 2004. Available at `http://dimacs.rutgers.edu/Workshops/Practice/slides/juels.ppt`

[14] A. Juels and M. Sudan. A Fuzzy Vault Scheme. IEEE Intl. Symp. on Info. Theory, 2002.

[15] A. Juels and M. Wattenberg. A Fuzzy Commitment Scheme. ACM CCS 1999.

[16] J. Katz, R. Ostrovsky, and M. Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. *Adv. in Cryptology — Eurocrypt 2001*, LNCS vol. 2045, Springer-Verlag, pp. 475–494, 2001.

[17] J. Katz, R. Ostrovsky, and M. Yung. Forward Secrecy in Password-Only Key-Exchange Protocols. *Security in Communication Networks: SCN 2002*, LNCS vol. 2576, Springer-Verlag, pp. 29–44, 2002.

[18] F. Monrose, M. Reiter, and S. Wetzel. Password Hardening Based on Keystroke Dynamics. ACM CCS 1999.

[19] N. Nisan and A. Ta-Shma. Extracting Randomness: A Survey and New Constructions. J. Computer and System Sciences 58(1): 148–173, 1999.

[20] E. Verbitskiy, P. Tuyls, D. Denteneer, and J.-P. Linnartz. Reliable Biometric Authentication with Privacy Protection. Proc. 24th Benelux Symp. on Info. Theory, 2003.

# A Constructing Robust Fuzzy Extractors

## A.1 Preliminaries

Before showing the construction of a robust fuzzy extractor, we need to introduce some unfortunate additional notation. First, we define the notion of a *labeled sketch*:

**Definition 8** An $(m, m', t)$-*labeled sketch* over a metric space $(\mathcal{M}, d)$ is a family of algorithms $\{(\mathsf{SS}_L, \mathsf{Rec}_L)\}_{L \in \Lambda}$ such that for any $L \in \Lambda$ the pair $(\mathsf{SS}_L, \mathsf{Rec}_L)$ is an $(m, m', t)$-secure sketch. We say this family is *well-formed* if $(\mathsf{SS}_L, \mathsf{Rec}_L)$ is well-formed for all $L \in \Lambda$. $\diamondsuit$

The definition of a *robust* labeled sketch is largely similar to that of Definition 5, except that we now additionally take the label $L \in \Lambda$ into account. Specifically:

**Definition 9** Given the family $\{(\mathsf{SS}_L, \mathsf{Rec}_L)\}_{L\in\Lambda}$ and random variables $\mathcal{W} = \{W_0, \ldots, W_n\}$ over a metric space $(\mathcal{M}, d)$, consider the following game between an adversary $\mathcal{A}$ and a challenger: Let $w_0$ (resp., $w_i$) be the value assumed by $W_0$ (resp., $W_i$). The challenger chooses a random $L \in \Lambda$, computes $\mathsf{pub} \leftarrow \mathsf{SS}_L(w_0)$, and gives $(\mathsf{pub}, L)$ to $\mathcal{A}$. Next, for $i = 1, \ldots, n$, the challenger and $\mathcal{A}$ proceed as follows: $\mathcal{A}$ outputs $(\mathsf{pub}_i, L_i) \neq (\mathsf{pub}, L)$ and is given $\mathsf{Rec}_{L_i}(w_i, \mathsf{pub}_i)$ in return. If there exists an $i$ such that $\mathsf{Rec}_{L_i}(w_i, \mathsf{pub}_i) \neq \perp$, we say that the adversary succeeds and this event is denoted by $\mathsf{Succ}$.

We say $\{(\mathsf{SS}_L, \mathsf{Rec}_L)\}_{L\in\Lambda}$ is an $(m, m'', n, \varepsilon, t)$-*robust labeled sketch* (over $(\mathcal{M}, d)$) if it is a well-formed $(m, \cdot, t)$-labeled sketch and: (1) for all $t$-bounded distortion ensembles $\mathcal{W}$ with $H_\infty(W_0) \geq m$ and all adversaries $\mathcal{A}$ we have $\Pr[\mathsf{Succ}] \leq \varepsilon$; and (2) the average min-entropy of $W_0$ — conditioned on the entire view of $\mathcal{A}$ throughout the above game — is at least $m''$. ♢

Our construction of a robust labeled sketch from any well-formed secure sketch $(\mathsf{SS}^*, \mathsf{Rec}^*)$ is essentially the same as that given in Section 3, except that we now include the label $L$ as one of the arguments to the random oracle. That is:

| $\underline{\mathsf{SS}_L(w)}$ | $\underline{\mathsf{Rec}_L(w, \mathsf{pub} = \langle \mathsf{pub}^*, h\rangle)}$ |
|---|---|
| $\mathsf{pub}^* \leftarrow \mathsf{SS}^*(w)$ | $w' = \mathsf{Rec}^*(w, \mathsf{pub}^*)$ |
| $h = H(w, \mathsf{pub}^*, L)$ | if $w' = \perp$ output $\perp$ |
| return $\mathsf{pub} = \langle \mathsf{pub}^*, h\rangle$ | if $H(w', \mathsf{pub}^*, L) \neq h$ output $\perp$ |
| | otherwise, output $w'$ |

The appropriate analogue of Theorem 1 holds, following exactly the same proof.

**Theorem 3** *If $(\mathsf{SS}^*, \mathsf{Rec}^*)$ is a well-formed $(m, m', t)$-secure sketch over a metric space $(\mathcal{M}, d)$ and $H$ is a random oracle with $k$ bits of output, then $\{(\mathsf{SS}_L, \mathsf{Rec}_L)\}_{L\in\Lambda}$ is an $(m, m'', n, \varepsilon, t)$-robust labeled sketch over the same metric space for any adversary making at most $q_h \ll 2^{-k/2}$ queries to the random oracle, where*

$$\begin{aligned} \varepsilon &= (q_h^2 + n) \cdot 2^{-k} + (3q_h + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \cdot 2^{-m'} \\ m'' &= m' - \log\left((q_h^2 + n) \cdot 2^{m'-k} + (3q_h + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}})\right). \end{aligned}$$

*If $k \geq m' + \log q_h$, this simplifies to $\varepsilon \leq (4q_h + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \cdot 2^{-m'}$ and $m'' \geq m' - \log(4q_h + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}})$.*

## A.2 Construction

With the above in place, we now construct a robust fuzzy extractor from any robust labeled sketch in a manner exactly following [9, Lemma 3.1]. Let $\{(\mathsf{SS}_L^*, \mathsf{Rec}_L^*)\}_{L\in\Lambda}$ denote an $(m, m', n, \varepsilon, t)$-robust labeled sketch over metric space $(\mathcal{M}, d)$ such that $\Lambda$ indexes a family $\{H_L\}_{L\in\Lambda}$ of pairwise-independent hash functions $H_L : \mathcal{M} \to \{0,1\}^\ell$ with $\ell = m' - 2\log\delta^{-1}$. Construct $(\mathsf{SS}, \mathsf{Rec})$ as follows:

| $\underline{\mathsf{Ext}(w)}$ | $\underline{\mathsf{Rec}(w, \mathsf{pub} = \langle \mathsf{pub}^*, L\rangle)}$ |
|---|---|
| $L \leftarrow \Lambda$ | $w' = \mathsf{Rec}_L^*(w, \mathsf{pub}^*)$ |
| $\mathsf{pub}^* \leftarrow \mathsf{SS}_L^*(w)$ | if $w' = \perp$ output $\perp$ |
| $R = H_L(w)$; $\mathsf{pub} = \langle \mathsf{pub}^*, L\rangle$ | otherwise, output $H_L(w')$ |
| return $(R, \mathsf{pub})$ | |

It is not too difficult to see that:

**Theorem 4** *If* $\{(\mathsf{SS}^*_L, \mathsf{Rec}^*_L)\}_{L\in\Lambda}$ *is an* $(m, m', n, \varepsilon, t)$*-robust labeled sketch over a metric space* $(\mathcal{M}, d)$*, then* $(\mathsf{Ext}, \mathsf{Rec})$ *is an* $(m, \ell, n, \varepsilon, t, \delta)$*-robust fuzzy extractor over the same space.*

We note that in the random oracle model it is clearly trivial to construct a robust fuzzy extractor from a robust sketch. The point of the above conversion is that it preserves robustness without requiring random oracles.

# B Definitions for Authenticated Key-Exchange Protocols

We use the standard notions of security for authenticated key exchange and mutual authentication as proposed by Bellare and Rogaway [3] and later improved by Bellare, Pointcheval, and Rogaway [1]. We provide a brief review of the model and definitions here with a focus on protocols for authenticated key exchange; see [3] for a discussion of mutual authentication.[9]

We assume for simplicity that the relevant honest players include only a single client and a single server, who share some long-term information in advance, which need not necessarily be a uniformly-distributed key. (Security holds also for the more general case when there are multiple parties who may share keys in an arbitrary manner: indeed, we will explicitly consider such a setting in Appendix C.) In the real world, a protocol determines how principals behave in response to signals from their environment. In the model, these signals are sent by the adversary. Each principal can execute the protocol multiple times; this is modeled by allowing each principal an unlimited number of *instances* with which to execute the protocol. We denote instance $i$ of user $U$ as $\Pi^i_U$. A given instance may be used only once. Each instance $\Pi^i_U$ has associated with it the variables $\mathsf{state}^i_U$, $\mathsf{term}^i_U$, $\mathsf{acc}^i_U$, $\mathsf{used}^i_U$, $\mathsf{sid}^i_U$, $\mathsf{pid}^i_U$, and $\mathsf{sk}^i_U$; the important ones for our purposes are the *session ID* $\mathsf{sid}^i_U$ which is simply taken to be the concatenation of all messages sent and received by the instance, the *partner ID* $\mathsf{pid}^i_U$ which reflects the party with whom $\Pi^i_U$ intends to communicate, and the *session key* $\mathsf{sk}^i_U$ whose computation is the goal of the protocol.

The adversary is assumed to have complete control over all communication in the network. An adversary's interaction with the principals in the network (more specifically, with the various instances) is modeled by the following *oracles*:

- $\mathsf{Send}(U, i, M)$ — This sends message $M$ to instance $\Pi^i_U$, and outputs the reply generated by this instance. We allow the adversary to prompt the unused instance $\Pi^i_U$ to initiate the protocol with partner $U'$ via the oracle call $\mathsf{Send}(U, i, \langle \mathsf{initiate}, U' \rangle)$.
- $\mathsf{Execute}(U, i, U', j)$ — This executes the protocol between the (unused) instances $\Pi^i_U$ and $\Pi^j_{U'}$, and outputs the transcript of the execution.
- $\mathsf{Reveal}(U, i)$ — This outputs session key $\mathsf{sk}^i_U$ for a terminated instance $\Pi^i_U$.
- $\mathsf{Test}(U, i)$ — This query does not correspond to any real-world action of the adversary, but instead enables a definition of security. A random bit $b$ is generated; if $b = 1$ the adversary is given $\mathsf{sk}^i_U$, and if $b = 0$ the adversary is given a random session key. This query is allowed only once, at any time during the adversary's execution.

[9] In any case, it is well known how to convert any authenticated key exchange protocol (with, say, implicit authentication) to a protocol which achieves (explicit) mutual authentication.

Send queries correspond to *active* attacks, while Execute queries correspond to *passive* (eavesdropping) attacks. Although an Execute query may be simulated by multiple Send queries, including the Execute query in the model potentially enables a tighter concrete security analysis.

Before defining a notion of security, we must first define the notion of *partnering*. We say terminated instances $\Pi_U^i$ and $\Pi_{U'}^j$ (with $U \neq U'$) are *partnered* iff (1) $\mathsf{pid}_U^i = U'$; (2) $\mathsf{pid}_{U'}^j = U$; and (3) $\mathsf{sid}_U^i = \mathsf{sid}_{U'}^j$. For correctness, we require that any two partnered instances which accept should output the same session key. The notion of partnering allows us to define the notion of *freshness*. We say a terminated instance $\Pi_U^i$ is *fresh* unless the adversary has queried $\mathsf{Reveal}(U, i)$ or $\mathsf{Reveal}(U', j)$ where $\Pi_U^i$ is partnered with $\Pi_{U'}^j$. Note that once an instance $\Pi_U^i$ is no longer fresh, the adversary already "knows" the value of $\mathsf{sk}_U^i$ and therefore should not "succeed" by correctly identifying it.

Finally, we say that event Succ occurs if the adversary queries the Test oracle on a fresh instance and correctly guesses the value of the bit $b$ used in answering this query. The advantage of adversary $\mathcal{A}$ in attacking protocol $\Pi$ is defined as $\mathsf{Adv}_{\mathcal{A},\Pi}(k) \stackrel{\text{def}}{=} |\Pr[\mathsf{Succ}] - \frac{1}{2}|$. We informally refer to the "security" of a protocol as the maximum advantage of any adversary running in some (implicit) time bound $T$. Formally, a protocol is secure if the advantage of any PPT adversary $\mathcal{A}$ is negligible in some security parameter $\kappa$ (where both the information shared between the parties as well as the protocol itself may depend on $\kappa$).

# C Definitions of Security for PAK Protocols

To define the security of a password-only authenticated key exchange (PAK) protocol, we use the same basic framework as in Appendix B but modify the definition of security. The definition given here is based on the definitions used in [1, 16, 11] except that, as discussed in Section 4.1, we define a stronger notion of security in which the adversary may choose (in an adaptive manner) non-uniform and dependent password distributions for the various clients. We model this in the following way: at the beginning of the experiment a random value $r \leftarrow \{0,1\}^{\mathsf{poly}(\kappa)}$ is chosen (and not revealed to the adversary), where $\kappa$ again represents the security parameter. Some number $i$ of clients/servers may be initialized with the password of the $i^{\text{th}}$ such party set to $W_i^{\mathsf{init}}(r)$ for some (poly-size) circuit(s) $\{\hat{W}_i\}$ known to the adversary. The adversary is furthermore allowed to make queries (at any time) to an oracle Initialize with the following functionality: on query $\mathsf{Initialize}(\mathsf{client}, C, W)$ a new client with name $C$ is initialized with password $W(r)$, where $W$ here is taken to be a circuit mapping $\{0,1\}^{\mathsf{poly}(\kappa)}$ to the space of possible passwords. (The query $\mathsf{Initialize}(\mathsf{server}, S, W)$ is defined similarly.) In case $C$ (resp., $S$) corresponds to the name of an existing client or server, the password is updated as requested (as discussed in [17], however, an instance which is already running must use the same password throughout even if the password changes). Note that this allows for arbitrary password distributions as well as arbitrary dependencies among the passwords used by various clients/servers.

Partnering, freshness, and the event Succ are defined exactly as in Appendix B. As for security, for a given adversary we say that *passwords have entropy* $m$ if $\bar{H}_\infty(W_i^{\mathsf{init}}) \geq m$ for all $i$ (where this is the average min-entropy conditioned on the initial view of the adversary before it makes any oracle queries) and also for all queries of the form $\mathsf{Initialize}(*, *, W)$ made by the adversary it is the case that $\bar{H}_\infty(W) \geq m$ (where the entropy of a distribution sampled by a circuit is defined in the natural way). In other words, the min-entropy of any password in the system is at least $m$. We stress that it is not essential to be able to efficiently determine the entropy of any given circuit —

all that is required is that the stated entropy bound hold. We say a PAK protocol $\Pi$ is secure if for all $m$ and all PPT adversaries $\mathcal{A}$ for which passwords have entropy $m$ we have[10]:

$$|2 \cdot \Pr_{\mathcal{A},\Pi}[\mathsf{Succ}] - 1| \leq Q/2^m + \mathsf{negl}(\kappa),$$

where $Q$ denotes a bound on the number of "on-line" attacks made by the adversary (which is at most the number of Send queries). Note that when the passwords for each client/server are chosen uniformly and independently from a dictionary of size $|D|$ we recover the definitions of [1, 16, 11].

We remark that if the adversary is required to make all his Initialize queries at the beginning of the game, then this model was considered implicitly (without any formal definitions or proofs of security) in [16, 11]. For our application, we need to allow the adversary to adaptively query the Initialize oracle at any point during the experiment.

# D Proof of Theorem 2

Given adversary $\mathcal{A}'$ attacking $\Pi'$, we construct an adversary $\mathcal{A}$ attacking the PAK protocol $\Pi$. Relating the success probabilities of these adversaries gives the desired result.

We assume for simplicity that the first message in $\Pi$ is sent by the client. Given explicitly computable $t$-bounded distribution ensemble $\mathcal{W} = \{W_0, \ldots\}$ with $H_\infty(W_0) \geq m$, and $(m, m', t)$-secure sketch $(\mathsf{SS}, \mathsf{Rec})$, and adversary $\mathcal{A}'$, we construct adversary $\mathcal{A}$ attacking $\Pi$ as follows:

1. The system is initialized by choosing $r \leftarrow \{0,1\}^{\mathsf{poly}(\kappa)}$, sampling $w_0$ according to $W_0(r)$, computing $\mathsf{pub} \leftarrow \mathsf{SS}(w_0)$, creating a client named $U|\mathsf{pub}$ with password $w_0$, and creating a server named $S|\mathsf{pub}$ with the same password. Adversary $\mathcal{A}$ is given $\mathsf{pub}$. (Note that $\bar{H}_\infty(W_0|\mathsf{pub}) \geq m'$.)

2. The adversary $\mathcal{A}$ then runs $\mathcal{A}'$, answering its oracle queries as follows (recall that for $\mathcal{A}'$ attacking $\Pi'$, there are only the two users with names $U$ and $S$):

   - When $\mathcal{A}'$ makes a query of the form $\mathsf{Execute}(U, i, S, j)$, adversary $\mathcal{A}$ simply makes the query $\mathsf{Execute}(U|\mathsf{pub}, i, S|\mathsf{pub}, j)$, obtains transcript $T$, and returns to $\mathcal{A}'$ the transcript $\mathsf{pub}|T$.
   - When $\mathcal{A}'$ makes a query of the form $\mathsf{Send}(S, j, M)$, adversary $\mathcal{A}$ simply makes the query $\mathsf{Send}(S|\mathsf{pub}, j, M)$. (The only exception is when $\mathcal{A}'$ requests $S$ to initiate execution of the protocol, in which case $\mathcal{A}$ simply returns $\mathsf{pub}$.)
   - When $\mathcal{A}'$ makes a query of the form $\mathsf{Send}(U, i, M)$ which is the first Send query to instance $\Pi^i_U$, then $M$ will be of the form $\mathsf{pub}_i$. There are two sub-cases to consider: (1) if $\mathsf{pub}_i = \mathsf{pub}$ then $\mathcal{A}$ requests $U|\mathsf{pub}$ to initiate execution of the protocol and returns the response to $\mathcal{A}'$. On the other hand, if (2) $\mathsf{pub}_i \neq \mathsf{pub}$ then $\mathcal{A}$ first calls $\mathsf{Initialize}(\mathsf{client}, U|\mathsf{pub}_i, W'_\ell)$, where $W'_\ell = \mathsf{Rec}(W_\ell, \mathsf{pub}_i)$ and this is the $\ell^{\text{th}}$ Send query directed by $\mathcal{A}'$ to the user. Next, $\mathcal{A}$ request $U|\mathsf{pub}_i$ to initiate execution of the protocol, and returns the response to $\mathcal{A}'$.
   - For any other queries of the form $\mathsf{Send}(U, i, M)$ made by $\mathcal{A}'$, the appropriate Send query (i.e., to the appropriate instance) is made by $\mathcal{A}$ and the response is given to $\mathcal{A}'$.
   - Similarly, for any Reveal or Test query made by $\mathcal{A}'$, the appropriate Reveal or Test query (i.e., to the appropriate instance) is made by $\mathcal{A}$ and the response is given to $\mathcal{A}'$.

[10] In fact, a tighter definition is possible but will not be required for our application.

3. Finally, $\mathcal{A}$ outputs whatever $\mathcal{A}'$ outputs.

It is not hard to see that $\mathcal{A}$ provides a perfect simulation to $\mathcal{A}'$. The only thing that needs to be verified is that any instance which is *fresh* in $\Pi'$ corresponds to a *fresh* instance in $\Pi$, but this follows easily from the following observations: if an instance $\Pi^i_{U|\mathsf{pub}'}$ is *not* fresh in $\Pi$ this means that a $\mathsf{Reveal}$ query was made to either this instance or to a partnered instance. In the former case, it is immediate that the corresponding instance in $\Pi'$ is not fresh. In the latter case, this means that a $\mathsf{Reveal}$ query was made to an instance of the form $\Pi^j_{S|\mathsf{pub}'}$ with $\mathsf{sid}^j_{S|\mathsf{pub}'} = \mathsf{sid}^i_{U|\mathsf{pub}'}$. But then the sessions IDs of the corresponding instances in $\Pi'$ also match, and hence the corresponding instances are partnered in $\Pi'$ as well. Thus, $\mathrm{Pr}_{\mathcal{A},\Pi}[\mathsf{Succ}] = \mathrm{Pr}_{\mathcal{A}',\Pi'}[\mathsf{Succ}]$.

To complete the proof, we need only observe that, in the above execution of $\Pi$, passwords have entropy $m''$. (For the case of $\bar{H}_\infty(W_0|\mathsf{pub})$ this follows from the facts that $(\mathsf{SS}, \mathsf{Rec})$ is an $(m, m', t)$-secure sketch and $m' > m''$; for the case of $W'_\ell$ the fact that $\bar{H}_\infty(W'_\ell|\mathsf{pub}) \geq m''$ for all $\ell$ follows from the same arguments as in the proof of Theorem 1.) This completes the proof of Theorem 2.

# E Instantiating Our Solution Using the KOY Protocol

We assume here that the reader is familiar with the KOY protocol [16]. In that protocol, the length of the password is limited to be at most the length of the order of the subgroup in which the DDH assumption is considered. I.e., if the subgroup under consideration is the set of quadratic residues in $\mathbb{Z}^*_p$ (where $p = 2q+1$ and $p, q$ are prime), then passwords must fall in the range $\{1, \ldots, q\}$. While this does not present a problem when "short" passwords are used, it may be problematic when "longer" passwords are used as in this work. Of course, one can always solve the problem by using larger values of $p, q$ (whose lengths will still be polynomial in the security parameter) but this may be inefficient in practice.

Luckily, for the case of the KOY protocol (as well as the protocols of [11]), better solutions are possible. Without going in to the full details, the idea is to modify the "Cramer-Shoup" encryption [7] used within the KOY protocol so as to allow for encryption of more than a single group element (with the rest of the KOY protocol modified accordingly). Thus, whereas the original KOY protocol uses a Cramer-Shoup "encryption" of the value $pw'$ of the form

$$g_1^r,\ g_2^r,\ h_1^r g_1^{pw'},\ (cd^\alpha)^r,$$

where $\alpha$ is computed as a hash of the first three components and $g_1, g_2, h_1, c, d$ make up the public key, we may, for example, encrypt the (longer) value $pw = pw_1|pw_2$ as follows:

$$g_1^r,\ g_2^r,\ h_1^r g_1^{pw_1},\ h_2^r g_1^{pw_2},\ (cd^\alpha)^r,$$

where $\alpha$ is computed now as a hash of the first four components and $h_2$ is additionally included in the public key. Whereas the former limits $pw'$ to be of length $|q|$, the latter allows $pw$ to be twice as long with only minimal added computation. Extensions to longer values of $pw'$ are possible, following the same approach.