Revisiting Fuzzy Password-Authenticated Key Exchange

Unified Leakage and Improved Efficiency

Sihang Pu[1], Li Yao[2], and Yu Yu[1,2]

[1] East China Normal University, School of Cryptology sxpu@sc.ecnu.edu.cn
[2] Shanghai Jiao Tong University {purplecom, yyuu}@sjtu.edu.cn

Abstract. Fuzzy Password-Authenticated Key Exchange (fuzzy PAKE) was introduced at Eurocrypt'18, enabling two parties to securely establish a shared cryptographic key using passwords that are close and potentially *low-entropy*. This is a generalization of standard PAKE and is particularly useful for handling mistyped passwords or inherently "noisy" samples such as biometrics. While this primitive sounds promising, existing definitions and protocols face several challenges: they may suffer from gapped leakage (namely, leakage of a small amount of information from failed authentications), or they may be restricted to Hamming distance.

We review the functionality and describe our contributions below:

1. **Leakage under Reuse.** We show that, when a password may be reused across multiple sessions, *any* leakage from the functionality has the same eventual compromise consequence under repeated use. Concretely, we construct an efficient ideal-world adversary that recovers an n-character password within $O(n)$ active sessions, assuming it starts from a password not too far from the target. Our attack applies to both Hamming distance and Minkowski (L_p) distance under reasonable parameter choices. This implies that, for common distance metrics in practical settings, several notions of leakage are effectively *unified* when passwords are reused, thereby resolving an open question from Eurocrypt'18.
2. **Supporting L_p Distances.** We propose substantially more efficient protocols for generalized Minkowski (L_p) and Hamming distances in the random oracle model. As a key technical step, we show how to achieve malicious security using only a semi-honestly secure oblivious key-value store, while addressing several subtleties in the proof. Concretely, compared with existing approaches for L_2 distance, our protocol reduces runtime by about 86% and bandwidth by about 99%.

1 Introduction

Password-authenticated key exchange (PAKE) has been extensively studied for several decades since the introduction by Bellovin and Merritt [BM92]. It enables two parties to securely establish a shared cryptographic key using a common password, even if the password has *low-entropy*. To ensure security, PAKE must prevent offline dictionary attacks, forcing adversaries to engage in an online session for each password guess attempt.

There are two main security definitions in the literature: the game-based security model formalized by Bellare, Pointcheval, and Rogaway in [BPR00], and the universal composable (UC) security model formalized by Canetti, Halevi, Katz, Lindell, and MacKenzie in [CHK+05]. In general, the UC model is preferred for several reasons. First, *composability*, the ability to integrate with other protocols, is always desirable and comes naturally under the UC model. For example,

the established cryptographic key might be used in subsequent protocols, and PAKE could run concurrently with other protocols. Additionally, the passwords used in PAKE might not be sampled uniformly and independently, but rather from correlated distributions, which are harder to model in game-based definitions. [3] Following standard practice and recent developments in the literature, this work considers only UC security.

A more generalized primitive, known as fuzzy password-authenticated key exchange, was introduced and formalized by Dupont, Hesse, Pointcheval, Reyzin, and Yakoubov in [DHP+18]. This extends the standard PAKE functionality by allowing for approximate matches between passwords based on a predefined metric and a matching threshold δ (i.e., error tolerance). Two parties agree on the same key when the distance between their passwords is not greater than δ. Such generalization is very useful in many scenarios, where passwords may be mistyped due to stress or carelessness [Muf15], or originate from inherently "noisy" sources like biometric data (such as facial patterns, fingerprints, or iris scans) [KMN+21] or geographic coordinates [GRS22].

To the best of our knowledge, fuzzy PAKE is the first authenticated key exchange protocol designed for low-entropy biometric data. Other biometric authentication methods like [CFP+16] are based on fuzzy extractors [DRS04], which assume a certain amount of min-entropy in biometric samples, therefore requiring a tailored heuristic analysis of the distribution of these biometrics [KMN+21].

While fuzzy PAKE is a promising primitive, its existing functionalities and protocols still fall short in several respects, which we outline below.

Recall the Functionality. We denote this functionality by $\mathcal{F}_{\mathsf{fPAKE}}$, following its introduction in [DHP+18]. A formal definition is given in Section 3.4.

Loosely speaking, $\mathcal{F}_{\mathsf{fPAKE}}$ takes as input two passwords $\mathsf{pw}_0, \mathsf{pw}_1$ from parties $\mathcal{P}_0, \mathcal{P}_1$, respectively. If the passwords are close, namely if $\mathsf{dist}(\mathsf{pw}_0, \mathsf{pw}_1) \leq \delta$ for some metric dist and threshold δ, then the functionality outputs the same cryptographic key K to both parties. Otherwise, it outputs independent uniformly random keys.

To capture adversaries' potential behaviors, $\mathcal{F}_{\mathsf{fPAKE}}$ provides a leakage interface for the ideal adversary $\mathcal{S}$:

TestPwd: *During session* sid*, upon receiving* $(\textsc{TestPwd}, \mathsf{sid}, \mathcal{P}_i, \mathsf{pw}')$ *from* $\mathcal{S}$ *for some* $i \in \{0, 1\}$*, it returns* ℓ_{c} *to* $\mathcal{S}$ *if* $\mathsf{dist}(\mathsf{pw}', \mathsf{pw}_i) \leq \delta$*, and returns* $\perp$ *otherwise. Each such query is allowed at most once per session for* $\mathcal{S}$*.*

This apparently generalizes the TestPwd query in the standard PAKE functionality: by setting $\ell_{\mathsf{c}} = \texttt{Correct}$ and $\delta = 0$, we immediately recover the standard PAKE. In contrast, fuzzy PAKE admits various levels of ℓ_{c}, ranging from as strong as the full password ($\ell_{\mathsf{c}} = \mathsf{pw}$) to as weak as a single bit ($\ell_{\mathsf{c}} = \texttt{Correct}$) since the adversary is always able to learn whether authentication succeeds by observing the honest party's output.

The Security Gap. The most practical approach in the literature, without relying on garbled circuits [Yao86,BHR12], was first proposed in [DHP+18] and later deployed and analyzed in [FHA+21]. Bootle, Faller, Hesse, Hostáková, and Ottenhues subsequently identified a flaw in the original construction and proposed a patch in [BFH+23].

[3] We acknowledge that arbitrary password distributions can also be captured through a game-based definition, as shown by Bellare and Shea [BS23] and by Gjøsteen [Gjø24], but composability is not as general as in UC.

Briefly speaking, their key idea is to employ a robust secret sharing (RSS) scheme [CDD+15], based on linear error correction codes (ECC), to mask one party's password with a random codeword. The other party can then decode the "noisy" code after unmasking it with their own password. This approach is quite efficient.

However, there is a gap between functionality and security, which is inevitable for ECC-based approaches. [4] Given a linear (n, k)-code, where n is the block length and k is the rank, the code can correct up to $\delta = \lfloor \frac{n-k}{2} \rfloor$ errors following the Singleton bound (assuming unique decoding). This implies that the fuzzy PAKE scheme

- remains secure as long as the adversary can only guess passwords whose distance from the true password exceeds $\gamma = n - k$ (i.e., security);
- achieves successful key agreement only when the distance between the two passwords is less than $\delta = \lfloor \gamma/2 \rfloor$ (i.e., correctness).

To formally capture this gap, the authors of [DHP+18] introduced an additional leakage term to account for the extra information learned by $\mathcal{S}$. The differences are highlighted in blue:

TestPwd: *During session* sid*, upon receiving* $(\textsc{TestPwd}, \mathsf{sid}, \mathcal{P}_i, \mathsf{pw}')$ *from* $\mathcal{S}$ *for some* $i \in \{0, 1\}$*, it returns* ℓ_{c} *to* $\mathcal{S}$ *if* $\mathsf{dist}(\mathsf{pw}', \mathsf{pw}_i) \leq \delta$*;*

returns ℓ_{m} *to* $\mathcal{S}$ *when* $\delta < \mathsf{dist}(\mathsf{pw}', \mathsf{pw}_i) \leq \gamma$*, and returns* $\perp$ *to* $\mathcal{S}$ *when* $\mathsf{dist}(\mathsf{pw}', \mathsf{pw}_i) > \gamma$*.*

Each such query is allowed at most once per session for $\mathcal{S}$*.*

Intuitively, if setting $\ell_{\mathsf{m}} = \mathsf{pw}$ (i.e., the full password is leaked), security is completely compromised as soon as an adversary can guess a password within fewer than γ errors. Thus, the new leakage ℓ_{m} makes successful guessing easier and weakens the functionality. Beyond this security issue, it also limits the practical usefulness of the primitive.

For example, suppose we want to tolerate 25% errors, i.e., $\delta = 0.25n$. By the Singleton bound, this requires setting $\gamma = 0.5n$. However, a uniformly sampled binary string of length n already agrees with the target in roughly half of the positions. Hence, when $\gamma \approx 0.5n$, an adversary can easily find a password within distance $d \leq \gamma$ and then use the leakage ℓ_{m} to compromise security.

Even if we allow list decoding, as suggested in [BFH+23], the threshold γ can only be improved slightly, to about $0.43n$.[5] This still remains vulnerable to the same trivial attack.

Therefore, for binary alphabets under Hamming distance, setting $\delta \geq 0.25n$ is not very useful in practice.

Leakage under Reuse. Even if we close the functionality gap by setting $\gamma = \delta$ and ruling out ℓ_{m}, the current notion of ℓ_{c} might still compose across executions.

In a single execution, revealing only $\ell_{\mathsf{c}} = \texttt{Correct}$ leaks one bit per accepted session, much weaker than $\ell_{\mathsf{c}} = \mathsf{pw}$. This is why [DHP+18] posed an efficient construction with leakage $\ell_{\mathsf{c}} = \texttt{Correct}$ as an open problem for stronger security in their work. [6]

[4] There is also a recent work that circumvents this issue by relaxing adversarially corrupted shares to random corruptions, as we will discuss in Concurrent Works in Section 1.2.

[5] In fact, list decoding is not desirable in practice, since it is much less efficient than unique decoding algorithms. For instance, list decoding for Reed–Solomon codes has time complexity $O(n^9)$ and is impractical for long passwords [G+07,BFH+23]. Concretely, as reported in [BFH+23], even for parameters $n = 22$ and $k = 16$, correcting only 3 errors is already more than 10 times slower.

[6] They also mention the "no leakage" variant with $\ell_{\mathsf{c}} = \perp$, which is not quite realistic once the adversary can observe the party's output, hence not useful *on its own*. We do not consider that case in this work.

The picture changes, however, when protocols run in multiple sessions or are reused across platforms with the *same* password in practice. Multiple sessions revealing $\ell_{\mathsf{c}} = \texttt{Correct}$ can be combined to drive the adversary toward a closer guess and eventually to the exact password, shrinking the apparent gap between $\ell_{\mathsf{c}} = \texttt{Correct}$ and $\ell_{\mathsf{c}} = \mathsf{pw}$.

One might wonder why an attacker would bother to recover the full pw if a nearby pw' already authenticates (i.e., observing $\ell_{\mathsf{c}} = \texttt{Correct}$). The answer is that users might routinely reuse the *same* fingerprint or memorable passcode across different systems. Importantly, these platforms might not share the same parameters, such as the threshold δ or even the distance function. In these cases, the adversary *is* motivated to fully extract the target password pw. Even if it can authenticate to one protocol with some pw', that pw' might not lie within the threshold — or even under the same metric — of another protocol.

Consider an extreme example. A user registers the same passcode pw with both Facebook [7] and Google servers. Facebook supports fuzzy PAKE (allowing up to δ errors), whereas Google only supports standard PAKE. After compromising a Facebook session, it would be crucial for the adversary (while being devastating for users) to extract the full password pw by exploiting ℓ_{c} through repeated Facebook interactions. By contrast, recovering pw via PAKE sessions directly from Google would require brute-forcing the entire password dictionary through active sessions.

Each query TestPwd, together with its corresponding ℓ_{c} output, provides a structured hint. Thus, even the weakest leakage, $\ell_{\mathsf{c}} = \texttt{Correct}$, may accumulate across repeated queries and reveal additional information to an adversary over multiple sessions.

How should we model leakage to capture the information that accumulates across multiple sessions?

Limited to Hamming Distance. Apart from the security gap, the ECC-based approach is inherently tied to Hamming distance and seems difficult to generalize to other metrics such as Euclidean or angular distance — metrics that are far more relevant for biometric similarity.

A common workaround is to use metric embeddings or approximation techniques, but these inevitably introduce distortions, which further widen the gap.

The authors of the original fuzzy PAKE work [DHP+18] also suggested a garbled-circuit-based approach that supports arbitrary distance functions and avoids this limitation. However, garbled circuits usually incur prohibitive communication costs (as we will compare in Table 4).

Is there an efficient fuzzy PAKE protocol supporting broader distances, without approximation?

1.1 Our Contributions

We try to answer the above questions and highlight our main contributions below.

Leakage Analysis In Section 4, we show that *any* leakage ℓ_{m} or ℓ_{c} under common distances can be *efficiently* amplified into full recovery when the same password may be reused across multiple sessions. Specifically, we construct an efficient ideal adversary $\mathcal{S}$ against the functionality, capable of recovering a password pw of length n within $O(n)$ sessions whenever either $\ell_{\mathsf{m}} \neq \perp$ or $\ell_{\mathsf{c}} \neq \perp$, for both Hamming and Minkowski (L_p) distances under reasonable parameters. Consequently, a functionality using a unified TestPwd interface with $\gamma = \delta$ and $\ell_{\mathsf{c}} = \mathsf{pw}$ already provides the strongest security guarantee one can reasonably expect in practice.

[7] In fact, Facebook does deploy a weak variant of fuzzy authentication by correcting mistyped characters from users [Muf15].

New Constructions In Section 5, we take a different approach and propose efficient protocols realizing the functionality with leakage $\ell_{\mathsf{c}} = \mathsf{pw}$ for both $L_{p\in[1,\infty]}$ and Hamming distances in the random oracle model. These constructions close the gap between security and functionality without relying on garbled circuits. Concretely, compared with existing approaches for L_2 distance, our new protocol saves runtime and bandwidth by around 86% and 99%, respectively.

1.2 Other Related Works

We briefly enumerate some related works that were not yet mentioned in the introduction. One relevant biometric authentication scheme by Agrawal, Badrinarayanan, Mukherjee, and Rindal [ABMR20] also supports low-entropy biometric samples. However, there are key differences between their scheme and fuzzy PAKE. First, their scheme focuses on authentication rather than key exchange, whereas authenticated key exchange implies both. Second, their security model involves three parties and permits only one maliciously corrupted party, which differs from the two-party model typically used in PAKE. Additionally, their protocol relies on tailored zero-knowledge proofs and additive homomorphic encryption, while ours is built on hybrid functionalities in a black-box manner.

Another related line of work is augmented PAKE (aPAKE), which provides stronger security guarantees and is particularly useful in client-server settings [JKX18,HJKW23]. Its fuzzy variant, known as fuzzy aPAKE, was introduced by Erwig, Hesse, Orlt, and Riahi in [EHOR20]. However, the fuzzy aPAKE protocol of [EHOR20] is inefficient and only practical for very short passwords, since its computational cost grows with the size of Hamming balls, which can be exponential. In addition, it also suffers from leakage ℓ_{m}, making it directly vulnerable to our attack. More recently, a stronger fuzzy aPAKE was proposed at Crypto 2025 by McQuoid, Rosulek, and Xu [MRX25], which improves security over [EHOR20] and avoids additional leakage ℓ_{m}. Nevertheless, it remains as inefficient as the earlier construction, with computational costs still scaling with the size of Hamming balls. As a result, it is only suitable for short passwords or for settings that tolerate a small number of errors.

Concurrent Works. Very recently, there are two concurrent works in the literature: one by Richardson, Rosulek, and Xu [RRX25], published at Eurocrypt 2026, and another by Ameri and Blocki [AB26], available on ePrint. The first work saves two rounds of communication compared with the garbled-circuit approach of [DHP+18] by introducing a new technique called conditionally input-revealing 2PC. Beyond round complexity, it also reduces the communication bandwidth by a factor of two compared with the dual execution of garbled circuits. However, given the large communication-performance gap shown in Table 4, our protocols are likely to perform better in practice. Additionally, we note that reducing two garbled circuits to one does not translate into a 2× improvement in running time over dual execution, since the two parties run concurrently.

The second work takes a different approach, introducing random robust secret sharing, which only corrects random corruptions, to bridge the security gap for the ECC-based approach of [EHOR20]. However, their technique is tightly tied to Hamming distance and incurs approximately 8× more communication and 13× more computational time. In contrast, our approach adds only a 1–2× overhead to ECC-based constructions while supporting more general distance measures.

2 Technical Overview

We provide an overview of the high-level intuitions and technical details, covering our leakage analysis and new constructions.

2.1 Leakage Analysis for Standard Functionalities

We show that *any* leakage, whether ℓ_{m} or ℓ_{c}, results in complete password compromise when reusing passwords across multiple sessions for common distance metrics. Without loss of generality, we describe our analysis using ℓ_{m}, noting that the same analysis directly applies to ℓ_{c} by setting $\gamma = \delta$.

Hamming Distances. Our recovery algorithm for Hamming distance is inspired by the black-box tracing algorithms that underlie many traitor-tracing works [BSW06,GGH+13,GKW18,GLW23]. These algorithms trace illegal users by making incremental oracle queries and observing "jumps" in the responses [Zha25].

For simplicity, we focus on the Hamming distance over the binary alphabet. Suppose the honest party $\mathcal{P}_{\mathsf{hon}}$ holds the target password $\tilde{\mathbf{u}} \in \{0,1\}^n$, and the adversary $\mathcal{S}$ observes a password $\mathbf{u}$ at distance $\mathsf{dist}(\tilde{\mathbf{u}}, \mathbf{u}) \le \gamma$. Assume there is leakage $\ell_{\mathsf{m}} \neq \perp$ and $\gamma > \delta$.

The adversary flips the bits of $\mathbf{u}$ incrementally, one at a time, producing a sequence $\mathbf{u}^0, \mathbf{u}^1, \ldots, \mathbf{u}^n$, where $\mathbf{u}^0 = \mathbf{u}$, and for each $i \in [n]$, $\mathbf{u}^i$ is obtained by additionally flipping the i-th bit of $\mathbf{u}^{i-1}$. Equivalently, $\mathbf{u}^i$ is obtained from $\mathbf{u}$ by flipping its first i bits. In particular, $\mathbf{u}^n$ is the bitwise *complement* of $\mathbf{u}$ (i.e., whose every bit is flipped).

When $\gamma < \lceil n/2 \rceil$,[8] we have $\mathsf{dist}(\tilde{\mathbf{u}}, \mathbf{u}^n) = n - \mathsf{dist}(\tilde{\mathbf{u}}, \mathbf{u}) > \gamma$. Moreover, each flip changes the distance by exactly 1, so

$$\mathsf{dist}(\tilde{\mathbf{u}}, \mathbf{u}^i) \in \{\mathsf{dist}(\tilde{\mathbf{u}}, \mathbf{u}^{i-1}) - 1, \mathsf{dist}(\tilde{\mathbf{u}}, \mathbf{u}^{i-1}) + 1\}.$$

Our crucial observation is that if $\mathcal{S}$ queries $\mathcal{F}_{\mathsf{fPAKE}}$ on inputs

$$\forall i = 0, 1, \ldots, n, \quad (\textsc{TestPwd}, \mathsf{sid}_i, \mathcal{P}_{\mathsf{hon}}, \mathbf{u}^i)$$

obtaining leakages $s_0, s_1, \ldots, s_n$, then there must exist an index $j \in \{0, \ldots, n-1\}$ resulting in a "jump" behavior such that $s_j \neq \perp$ but $s_{j+1} = \perp$. Indeed, $s_0 \neq \perp$ since $\mathsf{dist}(\tilde{\mathbf{u}}, \mathbf{u}^0) \le \gamma$, while $s_n = \perp$ since $\mathsf{dist}(\tilde{\mathbf{u}}, \mathbf{u}^n) > \gamma$. Because the distance changes by 1 at each step, the transition from acceptance to rejection can only occur when $\mathsf{dist}(\tilde{\mathbf{u}}, \mathbf{u}^j) = \gamma$.

Once such a "phase-transition point" $\mathbf{u}^j$ is identified, $\mathcal{S}$ can recover its bits one by one: for each position, it flips the corresponding bit and checks whether the resulting query is rejected, namely whether $\textsc{TestPwd}$ returns $\perp$. This gives a recovery procedure using at most $2n$ queries.

The number of sessions can be reduced further. By using a binary-search-style strategy to locate the boundary pair (s_j, s_{j+1}), the adversary need not test every candidate $\mathbf{u}^i$, reducing the complexity to $n + \log n$ sessions. Further optimizations for non-binary alphabets are discussed in Section 4.1.

Minkowski Distances. For Minkowski distances (L_p), the recovery attack is conceptually simpler than in the Hamming case. Without loss of generality, we focus on L_∞. The adversary queries $\mathcal{F}_{\mathsf{fPAKE}}$ while guessing the password one coordinate at a time, or equivalently, character by character. Starting from a password $\mathbf{u} \in \Sigma^n$ within L_∞-distance δ of the target $\tilde{\mathbf{u}}$, the adversary varies $\mathbf{u}$ along each coordinate axis until the query is rejected.

Repeating this procedure for every coordinate reveals the acceptance interval in each dimension. The midpoint of each such interval gives the corresponding coordinate of the center, which is exactly the target password. As before, a binary-search-style optimization reduces the number of queries needed per coordinate, giving an upper bound of $n \log(2\delta + 1)$ sessions for L_∞ queries. The general L_p case is discussed in Section 4.2.

[8] This is reasonable in most settings; otherwise, the adversary could easily guess a password with fewer than γ errors on average.

Optimality of Our Analysis. In fact, our upper bounds are essentially tight. We prove a lower bound on the number of TestPwd queries needed to fully recover the password through interaction with $\mathcal{F}_{\mathsf{fPAKE}}$ by relating the problem to compressed sensing, or sparse recovery [CRT06,Don06]. In compressed sensing, recovering a k-sparse vector $\mathbf{x} \in \{0,1\}^{\ell}$ from linear measurements $\mathbf{Ax}$ requires the measurement matrix $\mathbf{A}$ to have at least $\Omega\big(k \log \frac{\ell}{k}\big)$ rows.

In our setting, each leakage query induces a binary measurement. Let $\mathbf{x} \in \{0,1\}^{|\Sigma|^n}$ be the unit vector encoding the target password pw, so that $x_{\mathsf{pw}} = 1$. A query on a guessed password pw' defines a measurement vector $\mathbf{a}$ whose entries are

$$a_i := \mathbf{1}\big\{\mathsf{dist}(i, \mathsf{pw}') \le \gamma\big\}.$$

The leakage can then be written as $\ell_{\mathsf{m}}(\mathsf{pw}) = \langle \mathbf{a}, \mathbf{x} \rangle$.

Since the adversary already holds a password at some "medium" distance from the target, the effective search space is not all of Σ^n, but rather the metric ball of radius γ around that password. Let V denote the size of this ball. Even for an unbounded adversary, identifying pw within this search space therefore requires at least $\Omega(\log V)$ leakage queries. In Section B, we bound the volumes of Hamming and L_p balls and show that, for natural parameter regimes, such as $\gamma/n = \Omega(1)$ for Hamming distance, this lower bound simplifies to $\Omega(n)$.

Implications. Our analysis above shows that, when the same password may be reused, the setting $\gamma = \delta$ and $\ell_{\mathsf{c}} = \mathsf{pw}$ in $\mathcal{F}_{\mathsf{fPAKE}}$ is essentially optimal in practice for both Hamming and Minkowski distances. In other words, it already captures the strongest security guarantee one can reasonably hope to achieve. Indeed, the leakage ℓ_{m} is at least as informative as ℓ_{c}; moreover, any nontrivial leakage is enough to recover the full password. We therefore adopt this unified setting for the constructions in the following sections.

2.2 New Constructions for Generalized Distances

We begin with how to realize $\mathcal{F}_{\mathsf{fPAKE}}$ for the infinity distance. The general approach to achieve fuzzy PAKE involves first constructing a protocol that realizes a randomized fuzzy equality (RFE) functionality, and then applying the *split* transformation [BCL+05] to upgrade it to fuzzy PAKE. The RFE functionality introduced in [DHP+18] can be viewed as a fuzzy PAKE functionality under the assumption of **authenticated channels**. Both the garbled circuit-based and ECC-based approaches follow this framework, while the latter didn't employ this transformation in a black-box manner as detailed in [BFH+23].

Thus, realizing fuzzy PAKE reduces to constructing a secure RFE protocol. Motivated by the recent work from van Baarsen and Pu [vBP24], which gives a semi-honest ElGamal-based [ElG84] fuzzy-matching protocol for generalized Minkowski (L_p) distances, we use their approach as a starting point towards malicious RFE.

Fuzzy Matching from [vBP24]. The receiver (party $\mathcal{P}_0$) and sender (party $\mathcal{P}_1$) first encode their passwords as vectors over $\mathbb{Z}_q$: the receiver holds $\mathsf{pw}_0 := \mathbf{u} \in \mathbb{Z}_q^n$ and the sender holds $\mathsf{pw}_1 := \mathbf{v} \in \mathbb{Z}_q^n$, where $q = |\Sigma|$ is the alphabet size. For the L_∞ distance, the protocol works as follows:

1. The receiver creates an oblivious key-value store (OKVS) $\mathbf{e}$ [GPR+21], where for each position $i \in [n]$ and offset $j \in [-\delta, +\delta]$, $\mathbf{e}$ stores at key $u_i + j$ a fresh Diffie-Hellman pair $(R_{i,j}, R_{i,j}^s)$, here $R_{i,j} \leftarrow\!\$\, \mathbb{G}$ and $s \leftarrow\!\$\, \mathbb{Z}_p^*$. The receiver sends $\mathbf{e}$ together with public keys $(g, h = g^s)$ to the sender.

2. For each coordinate i, the sender decodes $(X_i, Y_i) \leftarrow \mathsf{Decode}(\mathbf{e}, v_i)$, and aggregates them as $X = \prod_{i=1}^{n} X_i$ and $Y = \prod_{i=1}^{n} Y_i$. It then re-randomizes (X, Y) into (A, B) using the random self-reduction for DDH tuples [NR97], and sends (A, B) to the receiver.
3. The receiver checks whether $A^s \stackrel{?}{=} B$ and outputs 1 or 0, indicating whether $\mathsf{dist}_\infty(\mathbf{u}, \mathbf{v}) \leq \delta$.

Correctness follows from the OKVS guarantees (see Section 3.1): decoding with a correct key returns a DH pair satisfying $X_i^s = Y_i$, whereas an incorrect key yields a uniform random element of $\mathbb{G} \times \mathbb{G}$. Security against a semi-honest sender relies on the DDH assumption together with OKVS obliviousness, which hides the inserted keys $u_i + j$. Security against a semi-honest receiver comes from the random self-reduction: when $X^s \neq Y$, the tuple (A, B) is statistically random.

To obtain an RFE protocol from this starting point, we must add malicious security and move beyond one-sided authentication (currently only the receiver learns a bit) to mutual key exchange, so that the parties either derive the same random key K or obtain independent keys $K_0, K_1 \leftarrow\!\!\$\, \mathcal{K}$.

Towards RFE. At first glance, extending the protocol to mutual key exchange seems immediate: using dual executions [MF06,HKE12], each party runs the protocol once as a receiver and once as a sender. When the passwords are close, in the second step party $\mathcal{P}_0$ (with secret s) also obtains (A', B') satisfying $B' = (A')^{s'}$ from $\mathcal{P}_1$, while party $\mathcal{P}_1$ (with secret s') obtains (A, B) satisfying $B = A^s$. Exchanging $((A')^s, B')$ and $(A^{s'}, B)$ then lets both sides derive

$$K = (AA')^{s+s'}.$$

Against semi-honest adversaries, this is secure because each tuple (A, B) is either uniformly random or pseudorandom.

Malicious security is more subtle. The main difficulty is extraction: if, say, $\mathcal{P}_1$ is corrupted, then from the simulator's perspective, the adversary's message $(A^{s'}, B)$ look uniformly random, so they reveal nothing about $\mathcal{P}_1$'s password. The simulator is therefore forced to rely on the adversary's encoding $\mathbf{e}'$, or on the adversary's decoding behavior (including queries to a simulated random oracle) against a simulated encoding $\mathbf{e}$.

Neither route is immediate. From $\mathbf{e}'$, extraction can fail because the adversary may insert malformed pairs (X', Y') with $Y' \neq (X')^{s'}$, which the simulator cannot distinguish from well-formed encodings. From the simulated $\mathbf{e}$, extraction during decoding can also fail: the adversary may issue multiple decoding queries per coordinate i, obtaining a list of candidates for each v_i, and the simulator has no way to consistently "mix and match" these lists into a single well-defined password.

Realizing from OPRF. One natural idea is to enforce well-formed encodings using NIZK proofs, or to work in idealized group models [Sho97,Mau05,FKL18]. Unfortunately, these options are either inefficient or require strong idealizing assumptions.

We instead give a simpler and more efficient approach based on malicious-secure oblivious pseudorandom functions (OPRFs) [FIPR05,BJSL11,JKKX16]. The key advantage is that OPRFs enable input extraction with only modest overhead. Concretely, the OPRF functionality $\mathcal{F}_{\mathsf{OPRF}}$ takes an input x from the client, outputs $\mathsf{F}(x)$ to the client, and reveals the corresponding function description F (a random function) to the server.[9]

Using $\mathcal{F}_{\mathsf{OPRF}}$, we consider the following symmetric protocol, where both parties run the same code. Parties $\mathcal{P}_0$ and $\mathcal{P}_1$ execute 2 OPRF instances in parallel with swapped roles. Party $\mathcal{P}_0$ acts as

[9] This is an informal description of $\mathcal{F}_{\mathsf{OPRF}}$. Please check Section 3.3 for formalization.

the server in one instance (holding a random function F), and as the client in the other instance (obtaining $\mathsf{G}(u_i)$ for each coordinate u_i of its password $\mathbf{u}$, where G is the random function held by $\mathcal{P}_1$).

Party $\mathcal{P}_0$ then creates an OKVS $\mathbf{e}$ that maps each key $u_i + j$ [10] to the value $R_i + \mathsf{F}(u_i + j)$, for all $i \in [n]$ and $j \in [-\delta, +\delta]$. Here $(R_1, \ldots, R_n)$ is an n-out-of-n additive sharing of a random value $R \leftarrow\!\!\$\ \mathbb{F}_{2^\lambda}$. Finally, $\mathcal{P}_0$ outputs the session key $K_0 := R + R'$, where R' is obtained from the other party's OKVS $\mathbf{r}$ by "unmasking" the decoded values:

$$R' \ := \ \sum_{i=1}^{n} \mathsf{Decode}(\mathbf{r}, u_i) \ - \ \mathsf{G}(u_i).$$

Semi-honest security follows from standard OPRF guarantees. In particular, the other party $\mathcal{P}_1$ can learn at most one OPRF evaluation per index $i \in [n]$; for all other inputs the values $\mathsf{F}(u_i + j)$ are uniform, and so the OKVS values $R_i + \mathsf{F}(u_i + j)$ are uniform as well, preserving OKVS obliviousness. Moreover, if the passwords are not within the acceptance threshold, then there is some coordinate i with $v_i \notin [u_i - \delta, u_i + \delta]$. For this coordinate, R_i remains perfectly hidden, which makes R, and hence the final key K_0, uniform from $\mathcal{P}_1$'s perspective. A further advantage over prior constructions [vBP24,GQL+24] is that our approach does not require encoding group elements into the OKVS.

The Subtleties about Extraction. This construction can be viewed as a form of programmable OPRF, or OPPRF [KMP+17]. However, to the best of our knowledge, there are no satisfactory candidates for maliciously secure OPPRFs from OKVS, because it is difficult to prevent overfitting attacks entirely.[11] For this reason, many instantiations targeting circuit-based and multi-party private set intersection (PSI) protocols [PRTY20,NTY21] adopt a relaxed security notion that tolerates a linear number of *overfitted* points. This relaxation is often acceptable for PSI, where such leakage may only affect a bounded number of extra set elements, but it is unsuitable for PAKE: each overfitted point could correspond to an additional password guess, thereby allowing an active adversary to test multiple passwords in a single session.

Crucially, malicious OPPRF security is overkill for our purposes: our construction only requires a maliciously secure OPRF together with a *semi-honest* OKVS. The key distinction between a malicious OPPRF and a composition of a malicious OPRF with an OKVS lies in extraction. In the OPPRF setting, the simulator can extract the adversary's effective password directly from the OPPRF functionality by observing the environment's input. In contrast, in the composed approach, the simulator may learn a tentative password from the OPRF interaction, but not from the OKVS, since the OKVS is only semi-honestly secure.

At first glance, this may appear insufficient: a malicious adversary could use one password in the OPRF interaction and a different password when constructing the OKVS. However, as we explain next, this concern does not arise in the RFE setting, where authenticated channels rule out man-in-the-middle attacks. In particular, the party creating the OKVS does *not* need to prove that the OKVS encodes the same password used in the OPRF. Suppose a malicious party $\mathcal{P}_1$, acting as the OPRF client, inputs a password outside the acceptance threshold, while constructing its OKVS encoding $\mathbf{r}$ using an inconsistent password. Then the value R contributed by $\mathcal{P}_0$ remains

[10] In the actual protocol, we domain-separate the coordinates by deriving OKVS keys as $x_{i,j} \leftarrow \mathsf{H}(i \,\|\, u_i + j)$, so that values cannot be matched across different coordinates.

[11] An overfitting attack refers to a malicious encoder embedding more key–value pairs into the OKVS than the scheme is designed to support.

information-theoretically hidden from $\mathcal{P}_1$ and is uniformly random. Indeed, at least one OPRF instance produces a random share, causing the unmasking process to fail. Consequently, the final key derived by the honest party is also uniformly random.

A more subtle concern is that the adversary might evaluate the OPRF on a close password while constructing the OKVS using an inconsistent password (e.g., outside the acceptance threshold). In this case, the real-world execution yields a random key for the honest party, whereas in the ideal world the password extracted from the OPRF interaction appears close to the target password, causing the ideal functionality to accept and output a chosen, or at least non-random, key. The core difficulty is that the simulator cannot distinguish whether the adversary used consistent inputs across the OPRF and the OKVS.

Fortunately, when the leakage function ℓ_c or ℓ_m in TestPwd reveals the full password, the simulator can recover *the honest party's* input from the extracted close password and program the final key accordingly. Specifically, the simulator emulates the honest party's local computation to derive the final key, and then sets this value through the NewKey interface. This is possible because NewKey allows $\mathcal{S}$ to set the honest party's final key whenever $\mathcal{S}$ provides an input within the acceptance threshold.

Generalized Distances. We can extend the construction to L_p distances in a similar way as [vBP24]. The key change is how we encode the per-coordinate shares. Instead of using a single value R_i for all shifts $j \in [-\delta, +\delta]$, we define

$$R_{i,j} := \alpha \cdot |j|^p + \beta_i,$$

where $\alpha, \beta_1, \ldots, \beta_n \leftarrow\!\!\$\ \mathbb{F}_{2^\lambda}$, and let $\beta = \sum_{i=1}^{n} \beta_i$. If both parties are honest, decoding and unmasking yields $\alpha \cdot t + \beta$, where $t = \mathsf{dist}(\mathbf{u}, \mathbf{v})^p = \sum_{i=1}^{n} |u_i - v_i|^p$. In [vBP24], fuzzy matching is enabled by having $\mathcal{P}_0$ transmit a *shuffled* list

$$T_0 := \{\mathsf{H}(\alpha \cdot 0 + \beta), \mathsf{H}(\alpha \cdot 1 + \beta), \ldots, \mathsf{H}(\alpha \cdot \delta^p + \beta)\}$$

of size $\delta^p + 1$, with H modeled as a random oracle. The other party $\mathcal{P}_1$ then checks membership without learning the exact distance. In our setting, however, shuffling prevents us from achieving key exchange, since both parties must agree on the same key index. We therefore keep the list ordered. This reveals $t = \mathsf{dist}(\mathbf{u}, \mathbf{v})^p$ whenever the passwords are within distance δ, but this leakage is already unavoidable and is not an issue, as shown in Section 2.1. Moreover, because both parties learn the same value t in the close case, the index t can drive key agreement: $\mathcal{P}_0$ sets $K_0 = \alpha \cdot t + \beta + R$, where R is recovered from $\mathcal{P}_1$'s OKVS $\mathbf{r}$ such that $T_1[t] = \mathsf{H}(R)$.

The security intuition mirrors the L_∞ case. If the adversary does not guess a δ-close password, then one of two cases occurs. First, all coordinates may satisfy $v_i \in [u_i - \delta, u_i + \delta]$, but the powered distance satisfies $t > \delta^p \geq 0$. In this case, the adversary obtains only masked values of the form $\alpha \cdot |j|^p + \beta_i$ for each dimension, where the fresh masks β_i information-theoretically hide α. Thus the honest party's derived key K_0 remains uniformly random given random α and $t \neq 0$. Second, some coordinate lies outside the allowed range. Then the same masking argument as in the L_∞ protocol applies, and the honest party's derived key is uniformly random.

Finally, we observe that the L_1 protocol also supports Hamming distance over *polynomially* large alphabets. Instead of using OKVS and OPRF, we use a 1-out-of-q OT for $|\Sigma| = q$. As the sender, $\mathcal{P}_0$ provides β_i for the character u_i, and $\alpha + \beta_i$ for all other characters. This mirrors the L_1 construction, since the receiver obtains $\alpha \cdot 1 + \beta_i$ exactly when the characters differ. Full constructions for both L_p and Hamming distances are given in Sections 5.2 and 5.3.

3 Preliminaries

Throughout this work, we denote the security parameter as λ, and the statistical parameter as κ. We use $[n]$ to represent the set $\{1, \ldots, n\}$, and $[a, b]$ to represent the interval $\{a, \ldots, b\}$, for $a \in \mathbb{Z}, b \in \mathbb{Z}, n \in \mathbb{N}$. In this section, we recall several cryptographic primitives and definitions as below. For other notions (including UC framework, PAKE functionality, etc.), please refer to Section A for details. For functionalities, we ignore several standard descriptions for brevity, e.g., for every query from parties, the corresponding sid and query category will be sent to the ideal adversary $\mathcal{S}$.

3.1 Oblivious Key-Value Stores

We recall the definition of OKVS below [GPR+21]. For OKVS, it has encoding size $m = n/\epsilon$ with some constant rate $\epsilon \leq 1$, encoding time $O(\kappa n)$, and constant decoding time.

Definition 1. *An* OKVS *is a key-value store associated with the following interfaces*

- $\mathsf{Encode}(L; \theta)$ *takes a list of* n *key-value pairs* L*, and returns an encoded vector* $\mathbf{e} \in \mathbb{F}^m$*.*
- $\mathsf{Decode}(\mathbf{e}, k; \theta)$ *takes a key* k*, then returns a decoded value* $v \in \mathbb{F}$*.*

and satisfying the following properties:

1. ***Correctness:*** *For any list of* $L \subset (\mathcal{K} \times \mathbb{F})^n$ *with distinct keys,* $\mathbf{e} \leftarrow \mathsf{Encode}(L; \theta)$ *with* $\theta \leftarrow\!\!\$\ \{0,1\}^\lambda$ *succeeds with* $1 - \mathsf{negl}(\kappa)$ *probability. Moreover, for any* $(k, v) \in L$ *there is* $\mathsf{Decode}(\mathsf{Encode}(L; \theta), k; \theta) = v$ *with probability* 1*.*
2. ***Obliviousness:*** *For any* $\{k_1, \ldots, k_n\}$, $\{k'_1, \ldots, k'_n\} \subseteq \mathcal{K}$ *of* n *distinct keys and any* $\theta \in \{0,1\}^\kappa$*, if* Encode *succeeds, then for* $v_1, \ldots, v_n \leftarrow\!\!\$\ \mathbb{F}$*:*

$$\left\{\mathbf{r} \leftarrow \mathsf{Encode}\big(\{(k_i, v_i)_{i\in[n]}\}; \theta\big)\right\} \quad \approx_c \quad \left\{\mathbf{r}' \leftarrow \mathsf{Encode}\big(\{(k'_i, v_i)_{i\in[n]}\}; \theta\big)\right\}. \tag{1}$$

Concrete Instantiations. In this work, we use polynomial encodings as our concrete instantiation of OKVS given that $n \leq 2^{10}$ in usual contexts of fPAKE. We note that the obliviousness property requires only computational security, even though nearly all known OKVS instantiations in fact provide the stronger guarantee of statistical security. For correctness, the overwhelming success probability of encoding allows standard OKVS constructions to be upgraded to perfect correctness through rejection sampling. Therefore, in our constructions and proofs, we directly treat the OKVS as statistically oblivious and perfectly correct.

3.2 Oblivious Transfer

The 1-out-of-q oblivious transfer (OT) functionality was formalized in [CLOS02,CCL15], and we recall it in Figure 1. In this work, we require an instantiation against malicious adversaries in the random oracle model. Fortunately, there are multiple efficient and maliciously secure two-round [MR19] or three-round protocols [CO15,OOS17,HL17] existing in the literature, where [MR19] and [HL17] are secure in UC model.

$\mathcal{F}_{\mathsf{OPRF}}$

PARAMETERS: Domain $\mathcal{X}$ and codomain $\mathcal{Y}$. For every id and $x \in \mathcal{X}$, value $\mathsf{F}_{\mathsf{sid},\mathsf{id}}(x)$ is initially undefined. Every time reference an undefined value $\mathsf{F}_{\mathsf{sid},\mathsf{id}}(x)$, assigns it $\mathsf{F}_{\mathsf{sid},\mathsf{id}}(x) \leftarrow\!\!\$\ \mathcal{Y}$ before returning.

INITIALIZATION: On message $(\text{INIT}, \mathsf{sid})$ from party SERVER, if this is the first INIT message for sid, set $\mathsf{tx} = 0$. From now on use tag SERVER to denote the unique entity which sent the INIT message for the session identifier sid. Ignore all subsequent INIT messages for sid.

- On $(\text{OFFLINEEVAL}, \mathsf{sid}, \mathsf{id}, x)$ from SERVER, return $\mathsf{F}_{\mathsf{sid},\mathsf{id}}(x)$ if $\mathsf{id} = \text{SERVER}$ or SERVER is corrupted.
- On $(\text{EVAL}, \mathsf{sid}, \mathsf{id}, x)$ from CLIENT, record $\langle \text{CLIENT}, \mathsf{id}, x \rangle$.
- On $(\text{SNDRCOMPLETE}, \mathsf{sid})$ from SERVER, set $\mathsf{tx} = \mathsf{tx} + 1$.
- On $(\text{RCVCOMPLETE}, \mathsf{sid}, \text{SERVER}, \mathsf{id})$ from CLIENT, recover record $\langle \text{CLIENT}, \mathsf{id}, x \rangle$; abort if such record does not exist, or SERVER is honest but $\mathsf{id} \neq \text{SERVER}$, or $\mathsf{tx} = 0$. Return $\mathsf{F}_{\mathsf{sid},id}(x)$ and set $\mathsf{tx} = \mathsf{tx} - 1$.

$\mathcal{F}_{\mathsf{OT}}^{(1,q)}$

PARAMETERS: Index space q and message space $\mathbb{F}_\Lambda$.

- On $(\text{SNDRCOMPLETE}, \mathsf{sid}, \mathbf{e})$ from SENDER, record $\mathbf{e} \in \mathbb{F}_\Lambda^q$.
- On $(\text{RCVCOMPLETE}, \mathsf{sid}, j)$ from RECEIVER, if there is a record $\mathbf{e} \in \mathbb{F}_\Lambda^q$, return e_j and halt.

Fig. 1. Functionalities of OPRF and OT

3.3 Oblivious Pseudorandom Functions

The oblivious pseudorandom function (OPRF) was formally introduced in [FIPR05]. In Figure 1, we recall the standard OPRF definition from [JKKX16, Fig. 1] by adopting some cosmetic changes from [JKX18]. To clarify, our definition aligns with [JKKX16] that assuming authenticated channel and not taking into account the adaptive corruption.

Concrete Instantiation. In this work, we use non-verifiable 2HashDH [JKKX16] as the malicious OPRF in UC. It is known to be secure in multiple variants of OPRF definitions under the One-More Gap CDH assumption [JKKX16] [12]. Specifically, in 2HashDH, CLIENT sends $A := H_1(x)^r$ to SERVER with its input x and randomness coin r, then, SERVER responds with $B := A^s$ where s is its secret key of the PRF. Finally, CLIENT outputs $H_2(x, B^{1/r})$ as the PRF evaluation.

3.4 Fuzzy PAKE

We first depict two associated functions, the distance function dist and the leakage function ℓ, in Figure 2. They are hard-coded as part of the public parameters of the ideal functionality of fuzzy PAKE $\mathcal{F}_{\mathsf{fPAKE}}^{\mathsf{dist},\ell}$. We usually will discard (part of) the superscripts notation when the context is clear or in general cases. Note that our distance function dist for L_p or L_∞ distance is defined over finite integer grid $\{0, \ldots, q\}^d$ (not q-ary lattice) as explained in Remark 1.

Recall the ideal functionality of fuzzy PAKE from [DHP+18,BFH+23] in Figure 3. We made a few modifications to simplify their original functionality as explained below:

[12] The original work [JKKX16] and its followup [JKX18] called it One-More Diffie-Hellman assumption, but it is indeed a Gap assumption as the reduction needs the help of a DDH oracle.

$\mathsf{dist}(\mathsf{pw}_0, \mathsf{pw}_1)$	$\ell_{\{\mathsf{m},\mathsf{c}\}}(\mathsf{pw})$
INPUTS: Two passwords $\mathsf{pw}_0, \mathsf{pw}_1 \in \Sigma^n$.	INPUTS: A password $\mathsf{pw} \in \Sigma^n$.
OUTPUTS:	OUTPUTS:
Parse $\mathsf{pw}_0 := \mathbf{u}$ and $\mathsf{pw}_1 := \mathbf{v}$. – Hamming Distance: $\sum_{i=1}^{n} (u_i \neq v_i)$ – L_∞ Distance: $\max_{i\in[n]} \lvert u_i - v_i \rvert$ – L_p Distance: $\left(\sum_{i=1}^{n} \lvert u_i - v_i \rvert^p\right)^{1/p}$	– Full password: Return pw – One bit: Return `Correct`

Fig. 2. Distance and Leakage Functions

1. We consider only the *symmetric* roles, that is, both parties have the same code and play the same protocol.
2. We use a concise and equivalent description of NEWKEY interface from [CHK+05,AHH21,KR24].
3. We refine the definition to ensure consistency across different levels of leakage. In particular, we do not allow $\ell_{\mathsf{c}} = \perp$ since the environment can always observe the final key of the honest party to deduce at least one bit information. [13]
4. In particular, for the leakage ℓ_{m}, the functionality marks a record as compromised only when $\ell_{\mathsf{m}} \neq \perp$; otherwise, it marks interrupted to capture the situation that $\ell_{\mathsf{m}} = \perp$ is equivalent to $\delta = \gamma$.

Clearly, the definition in Figure 3 recovers the standard PAKE functionality when $\gamma = \delta = 0$ and $\ell_{\mathsf{c}} = \texttt{Correct}$ returns one-bit leakage.

Similar to prior works [BFH+23], we provide an additional correctness property for fuzzy PAKE protocols to rule out the trivial instantiation reported in [RX23].

Definition 2 ([BFH+23]). *We say a fuzzy PAKE protocol $\Pi^{\delta}_{\mathsf{fPAKE}}$ is correct if for all $\mathsf{pw}_0, \mathsf{pw}_1 \in \Sigma^n$ with $\mathsf{dist}(\mathsf{pw}_0, \mathsf{pw}_1) \leq \delta$, where Σ is a finite alphabet, and for all passive (that is, honest-but-curious) adversaries $\mathcal{A}$, the probability that in the execution of Π with honest parties $\mathcal{P}_0, \mathcal{P}_1$ the first party $\mathcal{P}_0$ on input pw_0 outputs a different key than the second party $\mathcal{P}_1$ on input pw_1 in the presence of $\mathcal{A}$ is bounded by:*

$$\Pr\Big[K_0 \neq K_1 \;\Big|\; (K_0, K_1) \leftarrow out_{\mathcal{A}, \Pi^{\delta}_{\mathsf{fPAKE}}} \langle \mathcal{P}_0(\mathsf{pw}_0), \mathcal{P}_1(\mathsf{pw}_1)\rangle\Big] \;\leq\; \mathsf{negl}(\lambda) \tag{2}$$

where the probability is taken over the random coins of $\mathcal{P}_0, \mathcal{P}_1$, and $\mathcal{A}$.

3.5 Randomized Fuzzy Equality

We recall a useful functionality in Figure 4, denoted as randomized fuzzy equality $\mathcal{F}_{\mathsf{RFE}}$, introduced by [DHP+18]. It can be considered a fuzzy PAKE functionality with **authenticated channels**.

[13] In contrast, prior work [DHP+18] also considered the no-leakage case by explicitly *disallowing* the adversary from observing the honest party's output. While this model may be reasonable in certain contexts or useful as a building block, we do not adopt it here, as it is not very useful *on its own* in practice.

$\mathcal{F}_{\mathsf{fPAKE}}$

PARAMETERS: a password domain Σ^n, a key agreement threshold δ, a relaxed threshold γ, a key domain $\mathcal{K}$, a distance function dist and two leakage functions $\ell_{\mathsf{c}}, \ell_{\mathsf{m}}$ defined in Figure 2. It interacts with an adversary $\mathcal{S}$ and two parties $\mathcal{P}_0$ and $\mathcal{P}_1$ through the following queries:

- **Upon receiving a query** (NEWSESSION, $\mathsf{sid}, \mathsf{pw}_i$) **from the party** $\mathcal{P}_i$**:**
  - Send (NEWSESSION, $\mathsf{sid}, \mathcal{P}_i$) to $\mathcal{S}$;
  - If there is no record of the form $(\mathsf{sid}, \mathcal{P}_i, \cdot)$, record $(\mathsf{sid}, \mathcal{P}_i, \mathsf{pw}_i)$ and mark it fresh.
- **Upon receiving a query** (TESTPWD, $\mathsf{sid}, \mathcal{P}_i, \mathsf{pw}$) **from the adversary** $\mathcal{S}$**:** If there is a fresh record $(\mathsf{sid}, \mathcal{P}_i, \mathsf{pw}_i)$, then set $d \leftarrow \mathsf{dist}(\mathsf{pw}_i, \mathsf{pw})$ and do:
  - If $d \leq \delta$, mark the record $\mathsf{compromised}$, and send $\ell_{\mathsf{c}}(\mathsf{pw}_i)$ to $\mathcal{S}$;
  - If $\delta < d \leq \gamma$ and $\ell_{\mathsf{m}} \neq \perp$, mark the record $\mathsf{compromised}$, and send $\ell_{\mathsf{m}}(\mathsf{pw}_i)$ to $\mathcal{S}$;
  - Otherwise, mark the record $\mathsf{interrupted}$, and send $\perp$ to $\mathcal{S}$.
- **Upon receiving a query** (NEWKEY, $\mathsf{sid}, \mathcal{P}_i, K$) **from the adversary** $\mathcal{S}$**:** If there is a record $(\mathsf{sid}, \mathcal{P}_i, \mathsf{pw}_i)$ *not* marked $\mathsf{completed}$, then do:
  - If the record is $\mathsf{compromised}$, set $K_i = K$;
  - If the record is fresh, $\mathcal{P}_{1-i}$ is corrupt, and there is a record $(\mathsf{sid}, \mathcal{P}_{1-i}, \mathsf{pw}_{1-i})$ with $\mathsf{dist}(\mathsf{pw}_i, \mathsf{pw}_{1-i}) \leq \delta$, set $K_i = K$;
  - If the record is fresh, both parties are honest, and there is a $\mathsf{completed}$ record $(\mathsf{sid}, \mathcal{P}_{1-i}, \mathsf{pw}_{1-i}, K_{1-i})$ with $\mathsf{dist}(\mathsf{pw}_i, \mathsf{pw}_{1-i}) \leq \delta$, set $K_i = K_{1-i}$;
  - Otherwise, sample $K_i \leftarrow\!\!\$\ \mathcal{K}$.

  If $(\mathsf{sid}, \mathcal{P}_i, \mathsf{pw}_i)$ is still fresh, update it to $(\mathsf{sid}, \mathcal{P}_i, \mathsf{pw}_i, K_i)$. Finally, mark the record as $\mathsf{completed}$, and send K_i to $\mathcal{P}_i$.

Fig. 3. Fuzzy Password-Authenticated Key Exchange

The primary difference from $\mathcal{F}_{\mathsf{fPAKE}}$ is that TESTPWD interface does not take pw as a parameter, which means the adversary $\mathcal{S}$ can only test passwords using its protocol inputs. Note that we ignore the sid parameter for brevity, and set $\delta = \gamma$.

We recall the theorem to compile RFE into fPAKE in Theorem 1. This transformation is concretely efficient since it basically initializes an authenticated channel at the beginning of the protocol from an authenticated Diffie-Hellman key exchange, and makes further communication authenticated by using message-authenticated codes (MAC). The transformation adds additional 2 rounds of communication.

Theorem 1 (Recalled from [DHP+18]). *The randomized fuzzy equality functionality* $\mathcal{F}_{\mathsf{RFE}}^{\mathsf{dist},\ell_{\mathsf{c}}}$ *presented in Figure 4, can be transformed to UC-realize the functionality* $\mathcal{F}_{\mathsf{fPAKE}}^{\mathsf{dist},\ell_{\mathsf{c}}}$ *depicted in Figure 3, with the same* dist *and* ℓ_{c} *function; and it also satisfies the correctness property defined in Definition 2.*

Proof (sketch). We first transform the randomized fuzzy equality $\mathcal{F}_{\mathsf{RFE}}$ into a *split* functionality, then we show this split functionality is sufficient to realize the standard functionality $\mathcal{F}_{\mathsf{fPAKE}}$. The former is followed from [CCGS10, Theorem 3] and the latter is followed from [DHP+18, Theorem 11]. The correctness follows since the correctness of $\mathcal{F}_{\mathsf{RFE}}$ would be preserved after the transformation when assuming a passive adversary; moreover, given $\mathcal{F}_{\mathsf{RFE}}$ is defined over authenticated channels, the idealized adversary cannot issue TESTPWD for an empty password, their correctness are automatically implied by the functionality behaviors. □

$\mathcal{F}_{\mathsf{RFE}}$

PARAMETERS: a password domain Σ^n, a threshold δ, a key domain $\mathcal{K}$, a selected distance dist and leakage function ℓ_{c}. It interacts with an adversary $\mathcal{S}$ and two parties $\mathcal{P}_0$ and $\mathcal{P}_1$ via the following queries.

- **Upon receiving a query** (NEWSESSION, pw_i) **from the party** $\mathcal{P}_i$**:**
  - Send (NEWSESSION, $\mathcal{P}_i$) to $\mathcal{S}$;
  - If there is no record of the form $(\mathcal{P}_i, \cdot)$, record $(\mathcal{P}_i, \mathsf{pw}_i)$ and mark it fresh.
- **Upon receiving a query** (TESTPWD, $\mathcal{P}_i$) **from the adversary** $\mathcal{S}$**:** If there are two fresh records $(\mathcal{P}_0, \mathsf{pw}_0)$ and $(\mathcal{P}_1, \mathsf{pw}_1)$, and $\mathcal{P}_{1-i}$ is corrupt, then do:
  - If $\mathsf{dist}(\mathsf{pw}_i, \mathsf{pw}_{1-i}) \leq \delta$, mark $(\mathcal{P}_i, \mathsf{pw}_i)$ $\mathsf{compromised}$, and send $\ell_{\mathsf{c}}(\mathsf{pw}_i)$ to the adversary $\mathcal{S}$;
  - Otherwise, mark $(\mathcal{P}_i, \mathsf{pw}_i)$ $\mathsf{interrupted}$, and send 0 to the adversary $\mathcal{S}$.
- **Upon receiving a query** (NEWKEY, $\mathcal{P}_i, K$) **from the adversary** $\mathcal{S}$**:** If there is a record $(\mathcal{P}_i, \mathsf{pw}_i)$ *not* marked $\mathsf{completed}$, and a record $(\mathcal{P}_{1-i}, \mathsf{pw}_{1-i})$ marked fresh or $\mathsf{completed}$, then do:
  - If $(\mathcal{P}_i, \mathsf{pw}_i)$ is $\mathsf{compromised}$, set $K_i = K$;
  - If $(\mathcal{P}_i, \mathsf{pw}_i)$ is fresh, $\mathcal{P}_{1-i}$ is corrupt, and $\mathsf{dist}(\mathsf{pw}_0, \mathsf{pw}_1) \leq \delta$, set $K_i = K$;
  - If $(\mathcal{P}_i, \mathsf{pw}_i)$ is fresh, both parties are honest, and there is a $\mathsf{completed}$ record $(\mathcal{P}_{1-i}, \mathsf{pw}_{1-i}, K_{1-i})$ with $\mathsf{dist}(\mathsf{pw}_i, \mathsf{pw}_{1-i}) \leq \delta$, set $K_i = K_{1-i}$;
  - Otherwise, sample $K_i \leftarrow\!\!\$\, \mathcal{K}$.

  If $(\mathcal{P}_i, \mathsf{pw}_i)$ is fresh, update it to $(\mathcal{P}_i, \mathsf{pw}_i, K_i)$. Finally, mark $(\mathcal{P}_i, \mathsf{pw}_i)$ as $\mathsf{completed}$, and send K_i to $\mathcal{P}_i$.

Fig. 4. Randomized Fuzzy Equality

4 Analysis of the Functionality Leakage

In this section, we analyze the leakage of the functionality in the ideal world. Specifically, we show that there exists an efficient ideal-world adversary $\mathcal{S}$ that can fully recover the honest party's password using $O(n)$ sessions, where n denotes the password length, under both Hamming and Minkowski distances. Our attack is generic: it works for any amount of leakage, whether the leakage is ℓ_{m} or ℓ_{c}, even when the leakage consists of only a single bit.

We begin by describing, in Figure 5, a variant of binary search. Given an interval $[\alpha, \beta]$ such that $s_\alpha \neq \perp$ and $s_\beta = \perp$, the procedure finds an index $L \in [\alpha, \beta]$ for which $s_L \neq \perp$ while $s_{L+1} = \perp$.

$\mathsf{BinSearch}(\mathbf{u}^{[\alpha,\beta]})$

PARAMETERS: A range $[\alpha, \beta]$ and $\beta - \alpha + 1$ password candidates $\{\mathbf{u}^\alpha, \ldots, \mathbf{u}^\beta\}$ to be fed into TESTPWD.

1. Set $L = \alpha, R = \beta$.
2. If $L + 1 = R$, output (L, R); otherwise, set $M = \lfloor \frac{L+R}{2} \rfloor$.
3. Obtain the leakage s by sending (TESTPWD, $\mathsf{sid}, \mathcal{P}, \mathbf{u}^M$) to $\mathcal{F}_{\mathsf{fPAKE}}$ through a session sid.
4. If $s \neq \perp$, set $L = M$; otherwise, set $R = M$. Go to Step 2.

Fig. 5. Binary Search for Fuzzy PAKE Passwords

4.1 Hamming Distance

We consider standard parameter ranges: $\delta \leq \gamma < \lceil n/2 \rceil$, where γ denotes the security threshold and δ the key-agreement threshold; an alphabet of size $q = \mathsf{poly}(\lambda)$; passwords of length n; and security parameter λ.

Let $\tilde{\mathbf{u}}$ be the target password. Suppose the adversary holds a password $\mathbf{u} := u_1, \ldots, u_n$ that differs from $\tilde{\mathbf{u}}$ in at most γ positions. Equivalently, the functionality provides nontrivial leakage. We show that the following algorithm recovers $\tilde{\mathbf{u}}$.

Without loss of generality, we view the alphabet Σ as the additive group $\mathbb{Z}_q$, where $|\Sigma| = q$.

RecPwdHamming

1. Denote $\mathbf{u}^0 := \mathbf{u}$. Randomize each character of $\mathbf{u}$ one by one to get n passwords $\{\mathbf{u}^1, \ldots, \mathbf{u}^n\}$ such that
$$\forall\, i \in [n],\ \mathbf{u}^i = \mathbf{u}^{i-1} + \mathbf{e}^i \mod q,$$
where $\mathbf{e}^i$ is a unit vector such that all positions are 0 except that its i-th position $e^i_i \leftarrow\!\!\$\ \{1, \ldots, q-1\}$. That is, $\mathbf{u}^i$ is obtained by randomizing u^{i-1}_i from $\mathbf{u}^{i-1}$.
2. Find index j such that $\mathbf{u}^j$ has exact γ different characters from the targeted $\tilde{\mathbf{u}}$ through a binary search provided in Figure 5:
   - Obtain $(L, R) \leftarrow \mathsf{BinSearch}(\mathbf{u}^{[0,n]})$;
   - Set $j = L$.
3. Denote $\mathbf{r} := \mathbf{u}^j$. Randomize each character of $\mathbf{r}$ to get n passwords $\{\mathbf{r}^1, \ldots, \mathbf{r}^n\}$ such that
$$\forall\, i \in [n],\ \mathbf{r}^i = \mathbf{r} + \mathbf{e}^i \mod q,$$
where $\mathbf{e}^i$ is the same as in Step 1.
4. Feed each password from $\{\mathbf{r}^1, \ldots, \mathbf{r}^n\}$ into TestPwd of n fuzzy PAKE sessions to obtain n returned leakages $\mathbf{s} = \{s_1, \ldots, s_n\}$. If $s_i = \perp$, then it implies r_i is the correct character; otherwise, r_i is the wrong character. Denote these wrong positions as set $E := \{i \in [n] \mid s_i \neq \perp\}$.
5. [a] Set $\mathbf{u}^* := \mathbf{u}^{j+1}$ (which has exact $\gamma + 1$ errors). Enumerate the entire alphabet for each wrong position (i.e., $i \in E$) of $\mathbf{u}^*$ and feed the resulting password through TestPwd to obtain leakages. Update $\mathbf{u}^*$ with correct character if it returns any non-$\perp$ leakages.
6. Output the updated password.

[a] This step can be optimized as shown in Corollary 1.

Fig. 6. Recovery algorithm for Hamming distance through $\mathcal{F}_{\mathsf{fPAKE}}^{\mathrm{Hamming}}$

Theorem 2. *The algorithm depicted in Figure 6 can recover the n-character targeted password by issuing at most $n + \log n + (q-2)\gamma$ queries through* TestPwd *of $\mathcal{F}_{\mathsf{fPAKE}}^{Hamming}$ over an alphabet Σ of size $q = |\Sigma|$ if the security threshold $\gamma < \lceil n/2 \rceil$.*

Proof. In Step 1, the number of error positions in each $\mathbf{u}^i$ changes by at most one compared to $\mathbf{u}^{i-1}$ for every $i \in [n]$. Specifically:

- If i is a correct position (i.e., $u^{i-1}_i = \tilde{u}_i$), then $\mathbf{u}^i$ introduces one additional error compared to $\mathbf{u}^{i-1}$, since the i-th character of $\mathbf{u}^{i-1}$ is flipped into an error.
- If i is an error position, then $\mathbf{u}^i$ reduces the error count by one only if the randomized character satisfies $u^{i-1}_i + e^i_i = \tilde{u}_i \pmod q$; otherwise, the error count remains unchanged.

By the end, the final password $\mathbf{u}^n$ must have at least $n-\gamma$ error positions, since there are at least $n-\gamma$ correct characters in $\mathbf{u}$. With the threshold condition $\gamma < n/2$, it follows that $n-\gamma > \frac{n}{2} > \gamma$, so $\mathbf{u}^n$ necessarily exceeds the threshold, and hence TestPwd outputs $\perp$.

Now consider the BinSearch algorithm. It only returns an interval (L, R) when $L+1 = R$ and the algorithm never aborts. Moreover, it guarantees that $\mathbf{u}^L$ always produces some non-$\perp$ leakage, while $\mathbf{u}^R$ always yields $\perp$. Since the error count can vary by at most one between successive candidates, this implies that in Step 2 we must have $\mathbf{u}^j$ with exactly γ errors and $\mathbf{u}^{j+1}$ with exactly $\gamma+1$ errors.

Finally, Steps 3 and 4 recover the correct positions of $\mathbf{r} := \mathbf{u}^j$. Step 4 identifies correct characters by checking whether replacing a position with a randomized character causes the leakage to become $\perp$ (indicating the password has exceeded the threshold γ). In Step 5, the algorithm enumerates the possible values for each incorrect position of $\mathbf{u}^{j+1}$ to test whether it leads to non-$\perp$ leakage, exploiting the fact that $\mathbf{u}^{j+1}$ contains exactly $\gamma+1$ errors. □

Optimizations The algorithm provided in Figure 6 can be improved in Step 5 as it is possible to test *multiple* error positions in a single session. Specifically, this step can be modified as follows (sketch):

1. Set $\mathbf{u}^* = \mathbf{u}^{j+1}$, which has exact $\gamma+1$ errors. Denote as E these error positions. Generate $q-2$ [14] $\{\mathbf{r}^1, \ldots, \mathbf{r}^{q-2}\}$ such that
$$\forall\, i \in [q-2], \mathbf{r}^i = \mathbf{u}^* + \mathbf{e}^i_E,$$
where $\mathbf{e}^i_E$ is a multi-point vector that all positions are 0 except that for each $k \in E$ there is $e^i_k = i$.
2. Feed $\mathbf{r}^i$ into TestPwd for each $i \in [q-2]$ as a loop:
   (a) If TestPwd returns $\perp$, continue the loop;
   (b) Otherwise, use divide-and-conquer to find which position in E contains the correct character:
      i. Set the left half of positions of E from $\mathbf{r}^i$ and the right half from $\mathbf{u}^*$;
      ii. If TestPwd returns anything $\neq\perp$, then repeat the procedure on the left half, otherwise on the right half, until finding the correct position.
   (c) Update $\mathbf{u}^*$ on this position.

The above algorithm is easy to verify: there are at most γ errors, thus the algorithm will invoke the divide-and-conquer procedure at most γ times. Combined with Theorem 2, we have the following corollary.

Corollary 1. *The number of* TestPwd *queries from Theorem 2, that is, the number of active fuzzy PAKE sessions, can be optimized to*

$$n + \log n + \min\left\{(q-2)\gamma, \quad \gamma\log\gamma + q - 2\right\}.$$

4.2 L_p Distance

Given an initial password $\mathbf{u} := u_1, \ldots, u_n$ that is within a ball of radius γ centered at the targeted password $\tilde{\mathbf{u}}$, the following protocol would recover $\tilde{\mathbf{u}}$. We consider each component from a finite integer universe $\Sigma := \{0, \ldots, q-1\}$. The ideal functionality will compare the p-powered distance between two passwords.

[14] Note that we use $q-2$ here because at most $q-2$ queries are needed in worst case. Indeed, after $q-1$ failed attempts, the remaining value is guaranteed to be correct.

Remark 1. Note that our L_p and L_∞ distances are defined over finite integer grid $\{0,\ldots,q-1\}^d$ (not q-ary lattice $\mathbb{Z}_q$) and the distance is not mod-q wrapped distance. For instance, if $|\Sigma| = q$ and $u_i = 0, v_i = q-1$, then we use $|u_i - v_i| = q-1$ instead of $|u_i - v_i| \equiv 1 \mod q$.

However, we point out that our leakage analysis in this section and concrete protocols in Section 5 can be adapted to support q-wrapped distance for free.

Theorem 3. *The algorithm depicted in Figure 7 can recover the n-dimensional password by issuing at most $2n\log(2\gamma+1)$ queries through* TestPwd *of $\mathcal{F}_{\mathsf{fPAKE}}^{L_p}$ over the universe $\Sigma := \{0,\ldots,q-1\}$. Particularly, for L_∞ distance, it is at most $n\log(2\gamma+1)$.*

Proof. (Sketch.) The algorithm recovers the password dimension by dimension. For each dimension, we only need to consider the projection of the γ-radius ball centered at the targeted password $\tilde{\mathbf{u}}$. Given the initial password $\mathbf{u}$ leading to leakages $\neq\perp$ (i.e., $\mathsf{dist}(\tilde{\mathbf{u}},\mathbf{u}) \le \gamma$), it must be inside or on the ball surface. If we move the password $\mathbf{u}$ *through the coordinate axis*, it will encounter the boundary at some point. Since the projection of an L_p ball on each dimension is symmetric, we can always calculate the center of this projection; moreover, since any $L_{p<\infty}$ ball is confined in the L_∞ ball with the same radius, the length of the projection on each dimension is at most $2\gamma+1$. We can use binary search to efficiently find both upper and lower boundaries. Specifically, for L_∞ distance, the length of the projection is fixed implying one boundary is sufficient to calculate the center. □

RecPwdLp

1. Prepare an empty password $\mathbf{u}^*$.
2. For each dimension $i \in [n]$ of the initial password $\mathbf{u}^0 := \mathbf{u}$, run the following steps:
   (a) Prepare 2γ passwords $\{\mathbf{u}^1,\ldots,\mathbf{u}^{2\gamma}\}$ such that
   $$\forall\, j \in [2\gamma],\ \mathbf{u}^j = \mathbf{u} + \mathbf{e}_i^j,$$
   where $\mathbf{e}_i^j$ is a unit vector such that all positions are 0 except its i-th position is j. If $u_i + j > q-1$, set $\mathbf{u}^j = \perp$.
   (b) Locate the upper boundary B_{up} of the projection on dimension i from the ball through a binary search provided in Figure 5:
   – Obtain $(L,R) \leftarrow \mathsf{BinSearch}(\mathbf{u}^{[0,2\gamma]})$;
   – Set $B_{\mathsf{up}} = u_i + L$.
   (c) If $p = \infty$, set $B_{\mathsf{low}} = B_{\mathsf{up}} - 2\gamma$. Otherwise, locate the lower boundary B_{low} similarly as above by invoking $(L,R) \leftarrow \mathsf{BinSearch}(\mathbf{r}^{[-2\gamma,0]})$ and setting $B_{\mathsf{low}} = u_i + L$. Here, $\mathbf{r}^j = \mathbf{u} + \mathbf{e}_i^{-j}$ if $u_i - j \ge 0$ and $\perp$ otherwise.
   (d) Set the correct character as $w_i^* = \frac{B_{\mathsf{up}} + B_{\mathsf{low}}}{2}$.
3. Output the recovered password $\mathbf{u}^*$.

Fig. 7. Recovery algorithm for L_p distance by interacting $\mathcal{F}_{\mathsf{fPAKE}}^{L_p}$

5 New Constructions

In this section, we present practical and efficient fuzzy PAKE protocols that UC-realize $\mathcal{F}_{\mathsf{fPAKE}}$ for L_p and Hamming distances, with parameters set to $\delta = \gamma$ and $\ell_{\mathsf{c}} = \mathsf{pw}$.

We first describe the basic protocol for L_∞ in Section 5.1, and then extend it to L_p in Section 5.2. We then show that a protocol for Hamming distance over a polynomial-size alphabet can be obtained through a simple modification of the L_1 variant.

Our protocols follow the framework introduced in [BCL+05,CCGS10]: ideal functionalities that assume authenticated or secure channels can be upgraded to ones that make no such channel assumptions via the *split functionality transformation* (Theorem 1). This framework has also been adopted in prior works [DHP+18,BFH+23].[15]

For completeness, we recall the UC framework in Section A.1 and present the additional ideal functionalities such as PAKE in Section A.3.

5.1 The Basic Setting: Infinity Distance L_∞

By Theorem 1, it suffices in the following sections to focus on efficient realizations of $\mathcal{F}_{\mathsf{RFE}}$ with $\ell_{\mathsf{c}} = \mathsf{pw}$, since such realizations directly imply realizations of $\mathcal{F}_{\mathsf{fPAKE}}$. The full functionality $\mathcal{F}_{\mathsf{RFE}}$ is defined in Section 3.5.

We begin with the protocol $\Pi^{L_\infty}_{\mathsf{RFE}}$, shown in Figure 8, which realizes $\mathcal{F}_{\mathsf{RFE}}$. Let the password be $\mathbf{u} \in \Sigma^n$, where Σ denotes the alphabet. We assume an OKVS with algorithms Encode and Decode as in Section 3.1, and an OPRF functionality $\mathcal{F}_{\mathsf{OPRF}}$ as defined in Section 3.3. Each function output by $\mathcal{F}_{\mathsf{OPRF}}$ is a uniformly random function from $\{0,1\}^\lambda$ to $\mathbb{F}_\Lambda$. For simplicity, we write F for the function $\mathsf{F}_{\mathsf{sid},\mathcal{P}_0}$ obtained by $\mathcal{P}_0$, which $\mathcal{P}_0$ can evaluate offline through the $(\textsc{OfflineEval}, \mathsf{sid}, \mathcal{P}_0)$ interface of $\mathcal{F}_{\mathsf{OPRF}}$ in Figure 1. Evaluations of the hash function H are modeled through the random-oracle functionality $\mathcal{F}_{\mathsf{RO}}$.

Since the protocol is symmetric, it suffices to describe the view of one party, say $\mathcal{P}_0$. In one instance of $\mathcal{F}_{\mathsf{OPRF}}$, $\mathcal{P}_0$ acts as the sender, or Server, and obtains a function F, while $\mathcal{P}_1$ acts as the receiver. In a second instance, the roles are reversed: $\mathcal{P}_0$ acts as the receiver, or Client, queries on inputs $x_i \leftarrow \mathsf{H}(i \,\|\, u_i)$, and obtains outputs $g_i \leftarrow \mathsf{G}(x_i)$, where G is the random function obtained by $\mathcal{P}_1$ through $\mathcal{F}_{\mathsf{OPRF}}$.

In the Prep procedure, $\mathcal{P}_0$ first secret-shares a random value R into additive shares $R_1, \ldots, R_n \in \mathbb{F}_\Lambda$ such that $R = \sum_{i=1}^n R_i$. Then, for each coordinate $i \in [n]$ and each shift $j \in [-\delta, \delta]$ such that $u_i + j \in [0, |\Sigma|)$, it sets

$$x_{i,j} \leftarrow \mathsf{H}(i \,\|\, u_i + j)$$

and inserts the pair

$$(x_{i,j},\ R_i + \mathsf{F}(x_{i,j}))$$

into the OKVS. It then pads the OKVS input to size $n(2\delta+1)$ using dummy random pairs. The resulting encoding is denoted by $\mathbf{e}$.

Finally, $\mathcal{P}_0$ decodes the other party's encoding $\mathbf{r}$ at its own inputs x_i, subtracts the corresponding masks g_i, and sums the results:

$$R' := \sum_{i=1}^{n} \mathsf{Decode}(\mathbf{r}, x_i) - g_i.$$

It outputs the key $K_0 := R + R'$.

Lemma 1. *The protocol $\Pi^{L_\infty}_{\mathsf{RFE}}$ presented in Figure 8 UC-realizes the functionality $\mathcal{F}^{L_\infty}_{\mathsf{RFE}}$ defined in Figure 4 with $\ell_{\mathsf{c}} = \mathsf{pw}$ against malicious adversaries in the $\{\mathcal{F}_{\mathsf{OPRF}}, \mathcal{F}_{\mathsf{RO}}\}$-hybrid model, where the OKVS satisfies the definition in Section 3.1, and the finite field $\mathbb{F}_\Lambda$ has order $\Lambda = 2^\lambda$.*

[15] Although [BFH+23] does not explicitly apply this transformation, it relies on the same underlying idea.

$\mathsf{Prep}(\mathbf{u} \in \Sigma^n, R \in \mathbb{F}_\Lambda, \mathsf{F} : \{0,1\}^\lambda \mapsto \mathbb{F}_\Lambda)$

Sample $R_i \leftarrow\!\!\$\ \mathbb{F}_\Lambda$ such that $R = \sum_{i=1}^{n} R_i$
For each $i \in [n], j \in [-\delta, +\delta]$:
  If $u_i + j \in [0, |\Sigma|)$, set $x_{i,j} \leftarrow \mathsf{H}(i \,\|\, u_i + j)$; Otherwise, continue
  Set $\mathsf{list} := \mathsf{list} \cup \{(x_{i,j}, R_i + \mathsf{F}(x_{i,j}))\}$
Padding list to size $n(2\delta + 1)$ with random pairs
Return $\mathsf{Encode}(\mathsf{list})$

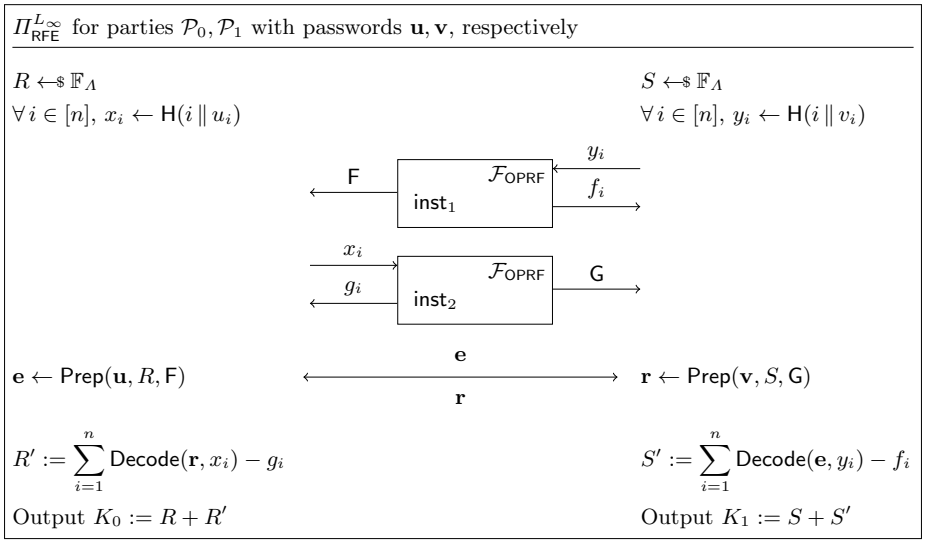


Fig. 8. Randomized Fuzzy Equality for L_∞ Distance

Proof. We present the intuitions in Section 2.2, and please refer to Section C.1 for the formal proof. □

Corollary 2 (of Theorem 1). *The protocol described in Figure 8 can be efficiently adapted to realize the functionality $\mathcal{F}^{L_\infty}_{\mathsf{fPAKE}}$, as defined in Figure 3. These adaptations also satisfy the correctness properties outlined in Definition 2.*

5.2 Extending to L_p

We extend the protocol for L_∞ distance to support general L_p distances. The resulting protocol is shown in Figure 9, where the changes relative to Figure 8 are highlighted in blue. We also show in Section 5.3 how the same approach can be adapted to handle Hamming distance over polynomial-size alphabets.

As shown in Figure 9, the parties $\mathcal{P}_0$ and $\mathcal{P}_1$ proceed symmetrically: they sample (α, β) and (ξ, ζ), respectively, obtain F and G from the OPRF instances, and exchange $(T_0, \mathbf{e})$ and $(T_1, \mathbf{r})$. Each

$\mathsf{Prep}(\mathbf{u} \in \Sigma^n, \alpha \in \mathbb{F}_\Lambda, \beta \in \mathbb{F}_\Lambda, \mathsf{F} : \{0,1\}^\lambda \mapsto \mathbb{F}_\Lambda)$

Sample $\{\beta_i \leftarrow\!\$\, \mathbb{F}_\Lambda\}$ such that $\beta = \sum_{i=1}^n \beta_i$

For each $i \in [n], j \in [-\delta, +\delta]$:

If $u_i + j \in [0, |\Sigma|)$, set $x_{i,j} \leftarrow \mathsf{H}(i \,\|\, u_i + j)$; Otherwise, continue

Set $\mathsf{list} := \mathsf{list} \cup \{(x_{i,j}, R_{i,j} + \mathsf{F}(x_{i,j}))\}$ where $R_{i,j} = \alpha \cdot |j|^p + \beta_i$

Padding list to size $n(2\delta+1)$ with random pairs

Return $\mathsf{Encode}(\mathsf{list})$ and $\{\mathsf{H}_T(\alpha \cdot t + \beta \,\|\, t)\}_{t \in [0, \delta^p]}$

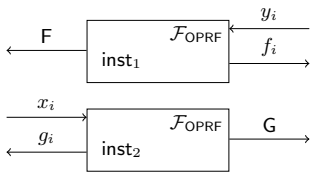
$\Pi_{\mathsf{RFE}}^{L_p}$ for parties $\mathcal{P}_0, \mathcal{P}_1$ with passwords $\mathbf{u}, \mathbf{v}$, respectively

$\alpha, \beta \leftarrow\!\$\, \mathbb{F}_\Lambda$

$\forall\, i \in [n],\ x_i \leftarrow \mathsf{H}(i \,\|\, u_i)$

$\xi, \zeta \leftarrow\!\$\, \mathbb{F}_\Lambda$

$\forall\, i \in [n],\ y_i \leftarrow \mathsf{H}(i \,\|\, v_i)$



$\mathbf{e}, T_0 \leftarrow \mathsf{Prep}(\mathbf{u}, \alpha, \beta, \mathsf{F})$

$T_0, \mathbf{e}$

$T_1, \mathbf{r}$

$\mathbf{r}, T_1 \leftarrow \mathsf{Prep}(\mathbf{v}, \xi, \zeta, \mathsf{G})$

$R := \sum_{i=1}^n \mathsf{Decode}(\mathbf{r}, x_i) - g_i$

$S := \sum_{i=1}^n \mathsf{Decode}(\mathbf{e}, y_i) - f_i$

If $\exists\, t$ such that $\mathsf{H}_T(R \,\|\, t) \in T_1$:

Output $K_0 := R + \alpha \cdot t + \beta$;

Otherwise, output $K_0 \leftarrow\!\$\, \mathbb{F}_\Lambda$

If $\exists\, t$ such that $\mathsf{H}_T(S \,\|\, t) \in T_0$:

Output $K_1 := S + \xi \cdot t + \zeta$;

Otherwise, output $K_1 \leftarrow\!\$\, \mathbb{F}_\Lambda$

Fig. 9. Randomized Fuzzy Equality for L_p Distance

party then computes

$$R = \sum_{i=1}^{n} \mathsf{Decode}(\mathbf{r}, x_i) - g_i$$

or, respectively,

$$S = \sum_{i=1}^{n} \mathsf{Decode}(\mathbf{e}, y_i) - f_i,$$

and outputs a key only if the corresponding hash value $\mathsf{H}_T(R \,\|\, t)$, respectively $\mathsf{H}_T(S \,\|\, t)$, appears in the received list.

For correctness, the field order Λ must satisfy $\Lambda > n\delta^p$. Since for security we set $\Lambda = 2^\lambda$, the protocol applies to L_p distances for any p satisfying

$$p < \frac{\lambda - \log_2 n}{\log_2 \delta}.$$

Lemma 2. *The protocol $\Pi^{L_p}_{\mathsf{RFE}}$ presented in Figure 9 UC-realizes the functionalities $\mathcal{F}^{L_p}_{\mathsf{RFE}}$ defined in Figure 4 against malicious adversaries in the $\{\mathcal{F}_{\mathsf{OPRF}}, \mathcal{F}_{\mathsf{RO}}\}$-hybrid model, where $p < \frac{\lambda - \log_2 n}{\log_2 \delta}$, the* OKVS *satisfies the definition in Section 3.1, and the finite field $\mathbb{F}_\Lambda$ has order $\Lambda = 2^\lambda$.*

Proof. Please refer to Section C.2 for the proof. □

Corollary 3 (of Theorem 1). *The protocol described in Figure 9 can be efficiently adapted to realize the functionality $\mathcal{F}^{L_p}_{\mathsf{fPAKE}}$, as defined in Figure 3. These adaptations also satisfy the correctness properties outlined in Definition 2.*

5.3 Simplifying for Hamming Distances

We show that the protocol for L_1 distance shown in Figure 9 can be simplified to support Hamming distance over a *polynomially* large alphabet. To modify a protocol designed for L_1 distance, two key changes are required:

- Modify the Prep function such that it directly encodes $\mathbf{e}_1, \ldots, \mathbf{e}_n$ without resorting to OKVS.
- Replace 2 instances of $\mathcal{F}_{\mathsf{OPRF}}$ with $2n$ instances of 1-out-of-q $\mathcal{F}^{1,q}_{\mathsf{OT}}$, where $q = |\Sigma|$ is the size of the alphabet.

Note that for each $i \in [n]$, $\mathbf{e}_i \in \mathbb{F}^q_\Lambda$ is encoded using all $q = |\Sigma|$ characters.

The protocol is illustrated in Figure 10, where the modifications from Figure 9 are highlighted in blue. In Figure 10, $\mathcal{P}_0, \mathcal{P}_1$ run Prep to obtain $(\mathbf{e}_{[n]}, T_0)$ and $(\mathbf{r}_{[n]}, T_1)$, exchange T_0 and T_1, and use 1-out-of-q OT to learn g_i (resp. f_i) based on their characters u_i (resp. v_i). They sum to R and S, then check membership in the received list and output K_0 and K_1 accordingly.

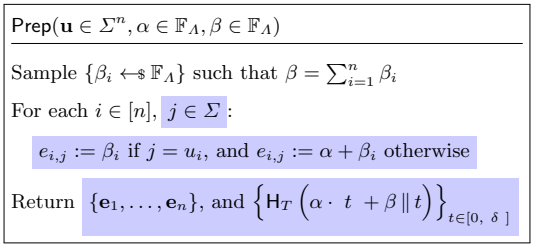
Lemma 3. *The protocol $\Pi^{Hamming}_{\mathsf{RFE}}$ described in Figure 10 can UC-realize the randomized fuzzy equality functionality $\mathcal{F}^{Hamming}_{\mathsf{RFE}}$.*

Proof. (sketch). The security guarantee is analogous to the L_1 protocol. Correctness follows since, for each position i, the OT returns β_i if the correct character is chosen, and $\alpha + \beta_i$ otherwise. Thus, the sum over all positions yields $R = \xi \cdot t + \zeta$ for $\mathcal{P}_0$ and $S = \alpha \cdot t + \beta$ for $\mathcal{P}_1$, where t is the Hamming distance between the two passwords. Each party outputs a key only if $t \leq \delta$, as verified by the received hash list T_1 (resp. T_0). Security against malicious adversaries follows from the security of the underlying OT and the random values α, β (resp. ξ, ζ). □

5.4 Efficiency

We thoroughly evaluate the efficiency of our protocols in this section and provide the costs of the protocols in terms of rounds, communication, and computation, by using 2HashDH [JKKX16] to instantiate the $\mathcal{F}_{\mathsf{OPRF}}$ functionality, using SipHash [AB12,DMS14] as the message authentication code to instantiate the split functionality, using AES to instantiate the ideal cipher, and SHA256 the random oracle, and using [HL17] (a variant of [CO15] taking three rounds) to instantiate $\mathcal{F}^{1,q}_{\mathsf{OT}}$.

Round Complexity. We report the round complexity of our protocols in Table 1. We measure round complexity by the message from each party that can be sent in parallel, often referred to as "flows" in some works. In other words, a round consists of a message from each party, which can be sent either sequentially or simultaneously. For example, an unauthenticated non-interactive



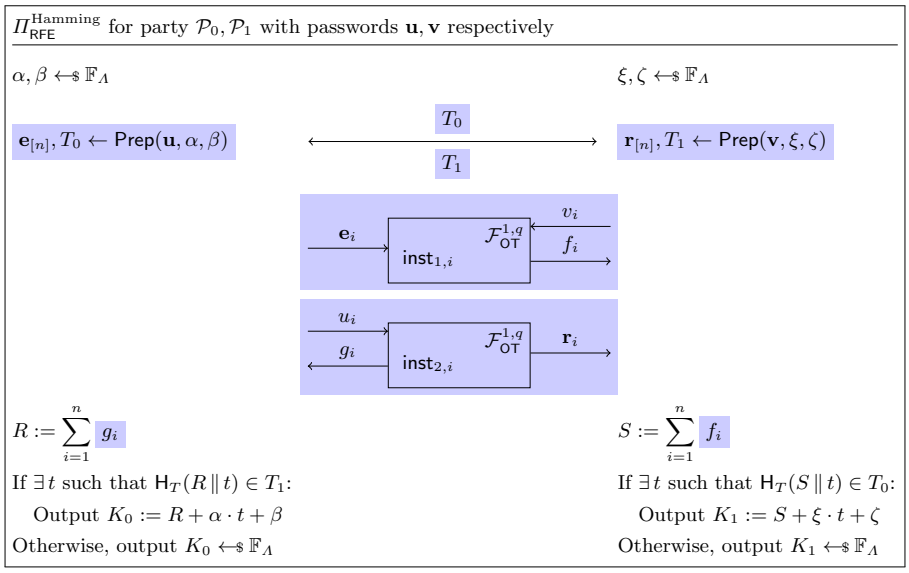


Fig. 10. Randomized Fuzzy Equality for Hamming Distances

Table 1. Number of rounds. For Hamming distance, if we use a two-round OT protocol to instantiate $\mathcal{F}_{\mathsf{OT}}^{1,q}$, the total number of rounds are presented in parentheses.

Functionality	L_∞	L_p	Hamming
$\mathcal{F}_{\mathsf{fPAKE}}^{\mathsf{pw}}$	4	4	5 (4)

Diffie-Hellman key exchange requires only one round, while an OT protocol needs at least two rounds.

It is important to note that the round numbers reported in [BFH+23, Table 1] use a different convention and do not align with our definition of "rounds". As shown in [BFH+23, Figure 8], the hybrid protocol actually requires at least 2 rounds in $\mathcal{F}_{\mathsf{SA}}$-hybrid model. Additionally, setting up the split authenticated channels $\mathcal{F}_{\mathsf{SA}}$ involves another 2 rounds. In total, their protocols also require 4 rounds as depicted in their full protocol [BFH+23, Figure 14]. The round complexity of [DHP+18, GC] is the same as our protocol for $\mathcal{F}_{\mathsf{fPAKE}}^{\mathrm{Hamming}}$, whether 4 or 5 rounds depending on the choice of OT protocol.

Communication Costs. We present the communication complexity of our protocols in Table 2. Each OPRF takes $2 \cdot \mathbb{G}$ elements, each batch of OT takes $qn \cdot \mathbb{F}_\Lambda + 2n \cdot \mathbb{G}$ elements. Split transformation takes $2 \cdot \mathbb{G}$ elements at the beginning, and one additional $\mathbb{F}_\Lambda$ element for each message sent by each party. Polynomial OKVS has encoding rate 1, so it takes $m \cdot \mathbb{F}_\Lambda$ elements to encode m values.

Table 2. Total communication costs (bits), where n is the password length, δ is the maximal allowed distance (note that $\delta \leq n$ in Hamming distance), and $q = |\Sigma|$ is the alphabet size. Each group element $\mathbb{G}$ (used in OT/OPRF) and each $\mathbb{F}_\Lambda$ element take 2λ and λ bits, respectively. Here $\Leftarrow$ should be replaced with the cell it points to.

Functionality	L_∞	L_p	Hamming
$\mathcal{F}_{\mathsf{fPAKE}}^{\mathsf{pw}}$	$(4n\delta + 10n + 12)\lambda$	$\Leftarrow + (2\delta^p + 2)\lambda$	$(2\delta + 2qn + 8n + 2)\lambda$

Computational Costs. We present the computational complexity of our protocols in Table 3. We denote the cost of Encode by $\mathtt{E}$ operations per key-value pair, and the cost of Decode by $\mathtt{D}$ operations. We use $\mathtt{exp}$ and $\mathtt{mul}$ to represent each group exponentiation and multiplication, respectively. Remaining operations (hashing, field addition/multiplication) are marginal costs compared to group operations and OKVS costs.

Table 3. Computational costs (measured by $\mathtt{exp}, \mathtt{mul}, \mathtt{E}, \mathtt{D}$) for each party (since our protocols are symmetric). Here $\Leftarrow$ should be replaced with the cell it points to.

Functionality	L_∞	L_p	Hamming
$\mathcal{F}_{\mathsf{fPAKE}}^{\mathsf{pw}}$	$(2\delta n + n)\mathtt{E} + n\mathtt{D} + (2\delta n + 4n)\mathtt{exp}$	$\Leftarrow$	$(3nq - 4n)\mathtt{mul} + 5n\mathtt{exp}$

Estimated Costs with Concrete Parameters. We provide the estimated costs of our protocols (realizing $\mathcal{F}_{\mathsf{fPAKE}}$) with concrete parameters in Table 4, where we set $\lambda = 128$, $\delta = 0.3n$, $q = |\Sigma| = 2$ for Hamming distance, and set $\delta = 32$, $q = |\Sigma| = 2^{64}$ for L_2 and L_∞ distances (which are common settings in fuzzy protocols). Note that the construction from [BFH+23] has **a gap** between the security and correctness (i.e., the security threshold and correctness threshold are not perfectly matched) as we have shown in the introduction. We estimate the running time of our (and previous) protocols on a 3.0 GHz CPU. We use $\mathtt{Curve25519}$ to instantiate the OT protocol [HL17], where $\mathtt{exp}$ and $\mathtt{mul}$ take $15.5\mu s$ and $0.3\mu s$, respectively. We estimate [DHP+18, GC] by assuming $0.05\mu s$ for each AES [16] encryption/decryption and $0.1\mu s$ to hash each 128-bit message. The garbled circuit is estimated based on Half Gates [ZRE15] with FreeXOR applied, where each AND gate takes 2 ciphertexts.

[16] Our estimates do not rely on AES-NI instructions, since such hardware acceleration may be unavailable on resource-constrained IoT devices.

Table 4. Estimated costs with concrete parameters. Note that the runtime of [BFH+23] is estimated by considering OT cost only, without taking account of the ECC decoding time.

Protocols	Bandwidth (KB)			Est. Run time (ms)		
	$n = 50$	$n = 100$	$n = 500$	$n = 50$	$n = 100$	$n = 500$
[BFH+23, ECC, Hamming, with security gap]	3	5	23	3.10	6.20	31.00
[DHP+18, GC, L_∞]	2009	4018	20093	269.12	538.24	2691.20
[DHP+18, GC, L_2]	14009	28018	140093	384.32	767.84	3843.20
[DHP+18, GC, Hamming]	13	25	125	3.92	7.84	39.20
Ours L_∞	108	215	1078	53.46	107.32	534.58
Ours L_2	140	247	1110	53.56	107.42	534.68
Ours Hamming	11	19	98	3.91	7.81	39.05

Relation to Fuzzy PSI. For completeness, we briefly discuss the relationship between RFE and fuzzy PSI.[17] Labeled fuzzy PSI is a conceptually closely related primitive: a *maliciously secure* labeled fuzzy-PSI protocol instantiated on singleton sets can realize a one-sided RFE functionality. Indeed, our semi-honest starting point is adapted from [vBP24]. The main challenge, however, lies in achieving malicious UC security. In particular, simply applying dual execution to a semi-honest fuzzy-PSI protocol does not provide the required consistency, and an additional mechanism for binding the parties to their inputs would be necessary. More recent semi-honest fuzzy-PSI techniques based on VOLE, prefix tries, or graph-based OKVS constructions are also conceptually relevant, but they neither provide malicious UC security nor yield a black-box construction of fuzzy PAKE.

We therefore use Garimella–Rosulek–Singh [GRS22] as the closest malicious fuzzy-PSI baseline; to the best of our knowledge, there are no other maliciously secure fuzzy-PSI protocols. Their construction, based on FSS and cut-and-choose, incurs a cost that grows exponentially with the dimension. For singleton L_∞ matching with $u = \delta = 32$, their communication complexity is $O(\lambda(\sigma + d + \lambda + 1))$, where $\sigma = O(\lambda(ud + (\log \delta)^d))$. We therefore estimate communication costs of approximately 180 KB for $d = 2$, 5.9 MB for $d = 5$, and 15.5 GB for $d = 10$. In comparison, our protocol requires only 108 KB even at $d = 50$, whereas the GC-based construction of [DHP+18] requires 1.96 MB. Thus, existing black-box approaches based on malicious fuzzy PSI do not offer an improvement over the GC-based construction of [DHP+18] for realizing fuzzy PAKE.

Acknowledgments.

Yu Yu is supported by the National Natural Science Foundation of China (Grant Nos. 62125204 and 92270201).

Use of Generative AI. The authors used ChatGPT (OpenAI) solely to assist with language refinement, including improving grammar, clarity, and phrasing of text originally written by the authors. The tool was not used to generate scientific results, technical contributions, analyses, or conclusions. All technical content and final wording were reviewed and approved by the authors.

[17] We thank the anonymous reviewers for bringing this to our attention.

References

AB12. Jean-Philippe Aumasson and Daniel J. Bernstein. SipHash: A fast short-input PRF. In Steven D. Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012: 13th International Conference in Cryptology in India*, volume 7668 of *Lecture Notes in Computer Science*, pages 489–508, Kolkata, India, December 9–12, 2012. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-34931-7_28.

AB26. Mohammad Hassan Ameri and Jeremiah Blocki. Random robust secret sharing with perfect privacy and its applications. Cryptology ePrint Archive, Paper 2026/653, 2026. URL: https://eprint.iacr.org/2026/653.

ABMR20. Shashank Agrawal, Saikrishna Badrinarayanan, Pratyay Mukherjee, and Peter Rindal. Game-set-MATCH: Using mobile devices for seamless external-facing biometric matching. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1351–1370, Virtual Event, USA, November 9–13, 2020. ACM Press. doi:10.1145/3372297.3417287.

AHH21. Michel Abdalla, Björn Haase, and Julia Hesse. Security analysis of CPace. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 711–741, Singapore, December 6–10, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-92068-5_24.

BCL+05. Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 361–377, Santa Barbara, CA, USA, August 14–18, 2005. Springer Berlin Heidelberg, Germany. doi:10.1007/11535218_22.

BFH+23. Jonathan Bootle, Sebastian H. Faller, Julia Hesse, Kristina Hostáková, and Johannes Ottenhues. Generalized fuzzy password-authenticated key exchange from error correcting codes. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VIII*, volume 14445 of *Lecture Notes in Computer Science*, pages 110–142, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore. doi:10.1007/978-981-99-8742-9_4.

BHR12. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012: 19th Conference on Computer and Communications Security*, pages 784–796, Raleigh, NC, USA, October 16–18, 2012. ACM Press. doi:10.1145/2382196.2382279.

BIPW10. Khanh Do Ba, Piotr Indyk, Eric Price, and David P. Woodruff. Lower bounds for sparse recovery. In Moses Charika, editor, *21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1190–1197, Austin, TX, USA, January 17–19, 2010. ACM-SIAM. doi:10.1137/1.9781611973075.95.

BJSL11. Ali Bagherzandi, Stanislaw Jarecki, Nitesh Saxena, and Yanbin Lu. Password-protected secret sharing. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 2011: 18th Conference on Computer and Communications Security*, pages 433–444, Chicago, Illinois, USA, October 17–21, 2011. ACM Press. doi:10.1145/2046707.2046758.

BM92. Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992. doi:10.1109/RISP.1992.213269.

BPR00. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, Bruges, Belgium, May 14–18, 2000. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-45539-6_11.

BS23. Mihir Bellare and Laura Shea. Flexible password-based encryption: Securing cloud storage and provably resisting partitioning-oracle attacks. In Mike Rosulek, editor, *Topics in Cryptology – CT-RSA 2023*, volume 13871 of *Lecture Notes in Computer Science*, pages 594–621, San Francisco, CA, USA, April 24–27, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-30872-7_23.

BSW06. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 573–592, St. Petersburg, Russia, May 28 – June 1, 2006. Springer Berlin Heidelberg, Germany. doi:10.1007/11761679_34.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. doi:10.1109/SFCS.2001.959888.

CCGS10. Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup. Credential authenticated identification and key exchange. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 255–276, Santa Barbara, CA, USA, August 15–19, 2010. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-14623-7_14.

CCL15. Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 3–22, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-48000-7_1.

CDD+15. Ronald Cramer, Ivan Bjerre Damgård, Nico Döttling, Serge Fehr, and Gabriele Spini. Linear secret sharing schemes from error correcting codes and universal hash functions. In Oswald and Fischlin [OF15], pages 313–336. doi:10.1007/978-3-662-46803-6_11.

CFP+16. Ran Canetti, Benjamin Fuller, Omer Paneth, Leonid Reyzin, and Adam D. Smith. Reusable fuzzy extractors for low-entropy distributions. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 117–146, Vienna, Austria, May 8–12, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-49890-3_5.

CHK+05. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421, Aarhus, Denmark, May 22–26, 2005. Springer Berlin Heidelberg, Germany. doi:10.1007/11426639_24.

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press. doi:10.1145/509907.509980.

CO15. Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *Progress in Cryptology - LATINCRYPT 2015: 4th International Conference on Cryptology and Information Security in Latin America*, volume 9230 of *Lecture Notes in Computer Science*, pages 40–58, Guadalajara, Mexico, August 23–26, 2015. Springer, Cham, Switzerland. doi:10.1007/978-3-319-22174-8_3.

CRT06. Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

DHP+18. Pierre-Alain Dupont, Julia Hesse, David Pointcheval, Leonid Reyzin, and Sophia Yakoubov. Fuzzy password-authenticated key exchange. In Nielsen and Rijmen [NR18], pages 393–424. doi:10.1007/978-3-319-78372-7_13.

DMS14. Christoph Dobraunig, Florian Mendel, and Martin Schläffer. Differential cryptanalysis of SipHash. In Antoine Joux and Amr M. Youssef, editors, *SAC 2014: 21st Annual International Workshop on Selected Areas in Cryptography*, volume 8781 of *Lecture Notes in Computer Science*, pages 165–182, Montreal, QC, Canada, August 14–15, 2014. Springer, Cham, Switzerland. doi:10.1007/978-3-319-13051-4_10.

Don06. D.L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006. doi:10.1109/TIT.2006.871582.

DRS04. Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors,

Advances in Cryptology – EUROCRYPT 2004, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-24676-3_31.

EHOR20. Andreas Erwig, Julia Hesse, Maximilian Orlt, and Siavash Riahi. Fuzzy asymmetric password-authenticated key exchange. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 761–784, Daejeon, South Korea, December 7–11, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-64834-3_26.

ElG84. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, USA, August 19–23, 1984. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-39568-7_2.

FHA+21. Mikhail Fomichev, Julia Hesse, Lars Almon, Timm Lippert, Jun Han, and Matthias Hollick. Fastzip: faster and more secure zero-interaction pairing. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '21, page 440–452, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3458864.3467883.

FIPR05. Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324, Cambridge, MA, USA, February 10–12, 2005. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-30576-7_17.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-96881-0_2.

G+07. Venkatesan Guruswami et al. Algorithmic results in list decoding. *Foundations and Trends® in Theoretical Computer Science*, 2(2):107–195, 2007.

GGH+13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press. doi:10.1109/FOCS.2013.13.

Gjø24. Kristian Gjøsteen. Password-authenticated key exchange and applications. Cryptology ePrint Archive, Report 2024/1057, 2024. URL: https://eprint.iacr.org/2024/1057.

GKW18. Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th Annual ACM Symposium on Theory of Computing*, pages 660–670, Los Angeles, CA, USA, June 25–29, 2018. ACM Press. doi:10.1145/3188745.3188844.

GLW23. Junqing Gong, Ji Luo, and Hoeteck Wee. Traitor tracing with $N^{1/3}$-size ciphertexts and $O(1)$-size keys from k-Lin. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 637–668, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-30620-4_21.

GPR+21. Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 395–425, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-84245-1_14.

GQL+24. Ying Gao, Lin Qi, Xiang Liu, Yuanchao Luo, and Longxin Wang. Efficient fuzzy private set intersection from fuzzy mapping. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part VI*, volume 15489 of *Lecture Notes in Computer Science*, pages 36–68, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore. doi:10.1007/978-981-96-0938-3_2.

GRS22. Gayathri Garimella, Mike Rosulek, and Jaspal Singh. Structure-aware private set intersection, with applications to fuzzy matching. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 323–352, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-15802-5_12.

Gur10. Venkatesan Guruswami. Introduction to coding theory: Gilbert varshamov bound, 2010. https://www.cs.cmu.edu/~venkatg/teaching/codingtheory/notes/notes2.pdf.

HJKW23. Julia Hesse, Stanislaw Jarecki, Hugo Krawczyk, and Christopher Wood. Password-authenticated TLS via OPAQUE and post-handshake authentication. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 98–127, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-30589-4_4.

HKE12. Yan Huang, Jonathan Katz, and David Evans. Quid-Pro-Quo-tocols: Strengthening semi-honest protocols with dual execution. In *2012 IEEE Symposium on Security and Privacy*, pages 272–284, San Francisco, CA, USA, May 21–23, 2012. IEEE Computer Society Press. doi:10.1109/SP.2012.43.

HL17. Eduard Hauck and Julian Loss. Efficient and universally composable protocols for oblivious transfer from the CDH assumption. Cryptology ePrint Archive, Report 2017/1011, 2017. URL: https://eprint.iacr.org/2017/1011.

HM04. Dennis Hofheinz and Jörn Müller-Quade. Universally composable commitments using random oracles. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 58–76, Cambridge, MA, USA, February 19–21, 2004. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-24638-1_4.

JKKX16. Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. Highly-efficient and composable password-protected secret sharing (or: How to protect your Bitcoin wallet online). In *2016 IEEE European Symposium on Security and Privacy*, pages 276–291, Saarbrücken, Germany, March 21–24, 2016. IEEE Computer Society Press. doi:10.1109/EuroSP.2016.30.

JKX18. Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In Nielsen and Rijmen [NR18], pages 456–486. doi:10.1007/978-3-319-78372-7_15.

KMN+21. Shuichi Katsumata, Takahiro Matsuda, Wataru Nakamura, Kazuma Ohara, and Kenta Takahashi. Revisiting fuzzy signatures: Towards a more risk-free cryptographic authentication system based on biometrics. In Vigna and Shi [VS21], pages 2046–2065. doi:10.1145/3460120.3484586.

KMP+17. Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1257–1272, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. doi:10.1145/3133956.3134065.

KR24. Jonathan Katz and Michael Rosenberg. LATKE: A framework for constructing identity-binding PAKEs. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part II*, volume 14921 of *Lecture Notes in Computer Science*, pages 218–250, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-68379-4_7.

Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12, Cirencester, UK, December 19–21, 2005. Springer Berlin Heidelberg, Germany. doi:10.1007/11586821_1.

MF06. Payman Mohassel and Matthew Franklin. Efficiency tradeoffs for malicious two-party computation. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 458–473, New York, NY, USA, April 24–26, 2006. Springer Berlin Heidelberg, Germany. doi:10.1007/11745853_30.

MR19. Daniel Masny and Peter Rindal. Endemic oblivious transfer. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on*

Computer and Communications Security, pages 309–326, London, UK, November 11–15, 2019. ACM Press. doi:10.1145/3319535.3354210.

MRX25. Ian McQuoid, Mike Rosulek, and Jiayu Xu. How to tolerate typos in strong asymmetric PAKE. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 610–641, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland. doi:10.1007/978-3-032-01881-6_19.

Muf15. Alec Muffett. Facebook: Password hashing & authentication, 2015. https://rwc.iacr.org/2015/program.html.

NR97. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, pages 458–467, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press. doi:10.1109/SFCS.1997.646134.

NR18. Jesper Buus Nielsen and Vincent Rijmen, editors. *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

NTY21. Ofri Nevo, Ni Trieu, and Avishay Yanai. Simple, fast malicious multiparty private set intersection. In Vigna and Shi [VS21], pages 1151–1165. doi:10.1145/3460120.3484772.

OF15. Elisabeth Oswald and Marc Fischlin, editors. *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany.

OOS17. Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-N OT extension with application to private set intersection. In Helena Handschuh, editor, *Topics in Cryptology – CT-RSA 2017*, volume 10159 of *Lecture Notes in Computer Science*, pages 381–396, San Francisco, CA, USA, February 14–17, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-52153-4_22.

PRTY20. Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 739–767, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-45724-2_25.

RRX25. David Richardson, Mike Rosulek, and Jiayu Xu. Conditionally input-revealing 2PC and fuzzy password-authenticated key exchange. Eurocrypt 2026, 2025. URL: https://eprint.iacr.org/2025/2315.

RX23. Lawrence Roy and Jiayu Xu. A universally composable PAKE with zero communication cost - (and why it shouldn't be considered UC-secure). In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 13940 of *Lecture Notes in Computer Science*, pages 714–743, Atlanta, GA, USA, May 7–10, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-31368-4_25.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-69053-0_18.

Top01. Flemming Topsøe. Bounds for entropy and divergence for distributions over a two-element set. *JIPAM. Journal of Inequalities in Pure and Applied Mathematics*, 2(2):Paper No. 25, 13 p.–Paper No. 25, 13 p., 2001. URL: http://eudml.org/doc/122035.

vBP24. Aron van Baarsen and Sihang Pu. Fuzzy private set intersection with large hyperballs. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part V*, volume 14655 of *Lecture Notes in Computer Science*, pages 340–369, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-58740-5_12.

VS21. Giovanni Vigna and Elaine Shi, editors. *ACM CCS 2021: 28th Conference on Computer and Communications Security*, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press. doi:10.1109/SFCS.1986.25.

Zha25. Mark Zhandry. Optimal traitor tracing from pairings. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part III*, volume 15603 of *Lecture Notes in Computer Science*, pages 305–335, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland. doi:10.1007/978-3-031-91131-6_11.

ZRE15. Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Oswald and Fischlin [OF15], pages 220–250. doi:10.1007/978-3-662-46803-6_8.

A Additional Definitions

A.1 Universal Composability

We use the UC framework of [Can01] throughout this work, which is the de facto standard in the PAKE literature. We introduce the basic terminology and high level intuitions of UC in this section. Please refer to [Can01] for a more detailed and formal explanation of the UC framework.

We only consider protocols between two parties (typically denoted as $\mathcal{P}_0$ and $\mathcal{P}_1$) that are executed in the presence of a PPT adversary $\mathcal{A}$, who can corrupt either party at the start of the protocol execution (i.e., we consider static corruption). This is because universal composability takes care of ensuring that a two-party functionality remains secure in a multi-party world. By corruption, we mean that the adversary gains full control over the corrupted party and learns its internal state. Both parties and the adversary $\mathcal{A}$ receive inputs from a special entity called the environment $\mathcal{Z}$, which represents everything external to the protocol execution. $\mathcal{Z}$ also receives outputs from both the parties and the adversary. The execution of a protocol in the presence of an adversary $\mathcal{A}$ is typically referred to as the "real world".

In the UC framework, security requirements for a protocol are defined via ideal functionalities. At a high level, an ideal functionality $\mathcal{F}$ specifies the intended input/output behavior of the parties, the permitted leakage, and the influence of an adversary on the protocol. $\mathcal{F}$ communicates with the environment through dummy parties $\mathcal{P}_0$ and $\mathcal{P}_1$, which simply forward messages between $\mathcal{Z}$ and $\mathcal{F}$. Additionally, the functionality interacts with an adversary, usually referred to as the simulator and denoted as $\mathcal{S}$. The execution of $\mathcal{F}$ in the presence of a simulator $\mathcal{S}$ is referred to as the "ideal world". A protocol Π is said to UC-realize (or UC-emulate) an ideal functionality $\mathcal{F}$ if, for every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for every PPT environment $\mathcal{Z}$, the ideal and real worlds are indistinguishable, except with negligible probability. In other words, any attack that can be carried out on the protocol Π can be simulated as an attack on the ideal functionality $\mathcal{F}$.

One of the key advantages of the UC framework is its native support for protocol composition, allowing for the natural modularization of protocol designs. Specifically, parties in a protocol can communicate with hybrid ideal functionalities $\mathcal{F}'_1, \mathcal{F}'_2, \ldots$. In such cases, the protocol is said to be defined in the $(\mathcal{F}'_1, \mathcal{F}'_2, \ldots)$-hybrid world. The UC composition theorem then allows these hybrid ideal functionalities to be securely replaced by concrete protocols that UC-realize those functionalities.

A.2 Random Oracle

We recall the Random Oracle (RO) functionality [HM04] as follows. It is clear that the random oracle model UC-emulates this functionality.

A.3 Password-Authenticated Key Exchange

We recall the standard PAKE functionality in Figure 12 from [AHH21] which is a modified variant of [CHK+05].

B The Lower Bounds and Near-Optimality

We recall the following lemma and present a theorem regarding the optimal number of queries needed for $\mathcal{F}_{\mathsf{fPAKE}}$.

$\mathcal{F}_{\mathsf{RO}}$

The functionality $\mathcal{F}_{\mathsf{RO}}$ proceeds as follows, running on security parameter λ, with a set of (dummy) parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ and an adversary $\mathcal{S}$:

- $\mathcal{F}_{\mathsf{RO}}$ keeps a list L (which is initially empty) of pairs of bit strings.
- **Upon receiving a value** $(\mathsf{sid}, m \in \{0,1\}^*)$ **from** $\mathcal{P}_i$ **or** $\mathcal{S}$:
  - If there is a pair (m, h') for some $h' \in \{0,1\}^\lambda$ in the list L, set $h := h'$.
  - If there is no such pair, choose uniformly $h \in \{0,1\}^\lambda$ and store the pair $(m, h) \in L$.

  Once h is set, reply to the activating machine with (sid, h).

Fig. 11. Random Oracle Functionality $\mathcal{F}_{\mathsf{RO}}$

Lemma 4 ([BIPW10]). *For the sparse recovery problem with linear measurement* $\mathbf{Ax}$*, the matrix* $\mathbf{A}$ *requires* $\Omega(k \log \frac{n}{k})$ *rows to recover a* n*-length and* k*-sparse* $\mathbf{x}$*.*

We show below the lower bound of the volume for the Hamming and L_p balls.

Volume of Hamming Balls. For any Hamming ball of radius γ over an alphabet of size $q > 1$, its volume V_{Hamming} is bounded by the following lemma.

Lemma 5 (Hamming Ball Volume Bound [Gur10]).

$$q^{\left(h_q(\frac{\gamma}{n}) - o(1)\right) \cdot n} \leq V_{Hamming} \leq q^{h_q(\frac{\gamma}{n}) \cdot n},$$

where $\frac{\gamma}{n} \in [0, 1 - 1/q]$ *and* $h_q : \{0,1\} \mapsto \mathbb{R}$ *is the* q*-ary entropy function that*

$$h_q(x) = x \log_q(q-1) - x \log_q x - (1-x) \log_q (1-x).$$

Lemma 6 (Lower Bounds of the Entropy Function [Top01]). *For* $x \in [0,1]$ *and* $q \geq 2$*, there is*

$$\frac{4x(1-x) + x \log(q-1)}{\log q} \leq h_q(x).$$

Combined with the above two lemmas, it is natural to obtain the following lower bound on the number of queries to recover the targeted password.

Theorem B.1. *It is necessary to issue*

$$\Omega\left(\frac{4\gamma(n-\gamma)}{n} + \gamma \log(q-1)\right)$$

queries (active sessions) through TestPwd *of* $\mathcal{F}_{\mathsf{fPAKE}}$ *to recover the targeted password for Hamming distance. Particularly, if the allowed error rate* $\frac{\gamma}{n} = \Omega(1)$*, we have the bound*

$$\Omega\left(n + \gamma \log(q-1)\right).$$

Note that our upper bound from Theorem 2 asymptotically matches this lower bound if the alphabet size $q = O(n)$ and $\frac{\gamma}{n} = \Omega(1)$: The upper bound from Theorem 2 and Corollary 1 is

$$O(n + \min\{(q-2)\gamma, \quad \gamma \log \gamma\}),$$

ignoring the additive $\log n$ terms.

Concretely, consider an example where the alphabet is binary ($q = 2$) and the tolerance rate is $0.3 = \frac{\gamma}{n}$. Given a close password, the adversary can recover the targeted password in at most $n + \log n$ queries against $\mathcal{F}_{\mathsf{fPAKE}}$, whereas it requires at least $0.84n$ queries.

$\mathcal{F}_{\mathsf{PAKE}}$

PARAMETERS: a password domain Σ^n and a key domain $\mathcal{K}$. It interacts with an adversary $\mathcal{S}$ and two parties $\mathcal{P}_0$ and $\mathcal{P}_1$ through the following queries:

- **Upon receiving a query** (NEWSESSION, $\mathsf{sid}, \mathsf{pw}_i$) **from the party** $\mathcal{P}_i$**:**
  - Send (NEWSESSION, $\mathsf{sid}, \mathcal{P}_i$) to $\mathcal{S}$;
  - If there is no record of the form $(\mathsf{sid}, \mathcal{P}_i, \cdot)$, record $(\mathsf{sid}, \mathcal{P}_i, \mathsf{pw}_i)$ and mark it fresh.
- **Upon receiving a query** (TESTPWD, $\mathsf{sid}, \mathcal{P}_i, \mathsf{pw}$) **from the adversary** $\mathcal{S}$**:** If there is a fresh record $(\mathsf{sid}, \mathcal{P}_i, \mathsf{pw}_i)$, then do:
  - If $\mathsf{pw}_i = \mathsf{pw}$, mark the record $\mathsf{compromised}$, and send 1 to $\mathcal{S}$;
  - Otherwise, mark the record $\mathsf{interrupted}$, and send 0 to $\mathcal{S}$.
- **Upon receiving a query** (NEWKEY, $\mathsf{sid}, \mathcal{P}_i, K$) **from the adversary** $\mathcal{S}$**:** If there is a record $(\mathsf{sid}, \mathcal{P}_i, \mathsf{pw}_i)$ *not* marked $\mathsf{completed}$, then do:
  - If the record is $\mathsf{compromised}$, set $K_i = K$;
  - If the record is fresh, $\mathcal{P}_{1-i}$ is corrupt, and there is a record $(\mathsf{sid}, \mathcal{P}_{1-i}, \mathsf{pw}_{1-i})$ with $\mathsf{pw}_i = \mathsf{pw}_{1-i}$, set $K_i = K$;
  - If the record is fresh, both parties are honest, and there is a $\mathsf{completed}$ record $(\mathsf{sid}, \mathcal{P}_{1-i}, \mathsf{pw}_{1-i}, K_{1-i})$ with $\mathsf{pw}_i = \mathsf{pw}_{1-i}$, set $K_i = K_{1-i}$;
  - Otherwise, sample $K_i \leftarrow\!\!\$\, \mathcal{K}$.

  If $(\mathsf{sid}, \mathcal{P}_i, \mathsf{pw}_i)$ is fresh, update it to $(\mathsf{sid}, \mathcal{P}_i, \mathsf{pw}_i, K_i)$. Finally, mark the record as $\mathsf{completed}$, and send K_i to $\mathcal{P}_i$.

Fig. 12. Password-Authenticated Key Exchange

Volume of L_p Balls. In contrast to L_p balls in continuous space, we focus on $L_{p\in[1,\infty]}$ balls defined over finite integer grid $\{0, \ldots, q\}^d$ in this section. For completeness, we provide the following lower bound on the volume.

Lemma 7. *The volume of an n-dimensional closed ball of radius γ is*

$$V_{L_p} \geq \left(\frac{2\gamma + 1}{n^{1/p}}\right)^n,$$

where a zero-centered closed ball is defined as $\mathsf{ball}_0^{\gamma, L_p} := \{x \in \mathbb{Z}^n \mid \|x\|_p \leq \gamma\}$.

Proof. It follows from the fact that an n-dimensional hypercube with radius $\frac{\gamma}{n^{1/p}}$ is inside $\mathsf{ball}_0^{\gamma, L_p}$. □

We now obtain the following lower bound on the number of queries.

Theorem B.2. *It is necessary to issue*

$$\Omega\left(n\left(\log(2\gamma + 1) - \frac{1}{p}\log n\right)\right)$$

queries (active sessions) through TESTPWD *of* $\mathcal{F}_{\mathsf{fPAKE}}$ *to recover the targeted password for* $L_{p\in[1,\infty]}$. *Moreover, if the radius* $\gamma = \Omega(n^{1/p})$, *we have the bound*

$$\Omega\left(n \cdot \log \gamma\right).$$

Similarly as above, our upper bound in Section 4.2 matches this lower bound tightly when $\gamma = \Omega(n^{1/p})$. Particularly, for L_∞ distance, the upper bound $n\log(2\gamma + 1)$ is *concretely* tight.

C Missing Proofs

C.1 The Proof of Lemma 1

Proof. We prove the theorem through a sequence of indistinguishable hybrids starting with the real execution of the protocol and ending with the simulated one in the ideal world. Without loss of generality, we assume $\mathcal{P}_0$ is corrupted and sometimes use $\mathcal{P}_{\mathsf{cor}}$ to refer to it.

Hybrid 0: Real Game. This is the real execution of $\Pi^{L_\infty}_{\mathsf{RFE}}$ where the environment $\mathcal{Z}$ runs the protocol with parties $\mathcal{P}_0$ and $\mathcal{P}_1$, both having access to ideal functionalities $\mathcal{F}_{\mathsf{OPRF}}, \mathcal{F}_{\mathsf{RO}}$. There is also a dummy adversary $\mathcal{A}$ completely instructed by $\mathcal{Z}$.

Hybrid 1: Simulator and the Dummy Functionality. This is the same as the real game, but the entire protocol is run inside a single machine, called the simulator $\mathcal{S}$. There is also a dummy (i.e., empty) functionality $\mathcal{F}$. The simulation among $\mathcal{F}, \mathcal{S}, \mathcal{P}_{\{0,1\}}$ is as follows:

- $\mathcal{S}$ simulates $\mathcal{F}_{\mathsf{RO}}$ and 2 instances of $\mathcal{F}_{\mathsf{OPRF}}$ by running their code directly;
- Honest party $\mathcal{P}_{\mathsf{hon}}$ merely sends input and receives output from the dummy functionality $\mathcal{F}$;
- $\mathcal{F}$ simply forwards input from $\mathcal{P}_{\mathsf{hon}}$ to $\mathcal{S}$ and forwards output from $\mathcal{S}$ to $\mathcal{P}_{\mathsf{hon}}$;
- $\mathcal{S}$ runs the code of each honest party, then outputs the result to $\mathcal{F}$.

This hybrid is perfectly indistinguishable from the $\mathbf{Hybrid_0}$ as the view of $\mathcal{Z}$ is unchanged.

Hybrid 2: Adding NewSession, NewKey Interfaces. Now the functionality $\mathcal{F}$ implements NewSession and NewKey interfaces:

- $(\textsc{NewSession}, \mathsf{sid}, \mathsf{pw}_i)$ from $\mathcal{P}_i$: The same as in Figure 4, but also forwards pw_i to $\mathcal{S}$.
- $(\textsc{NewKey}, \mathsf{sid}, \mathcal{P}_i, K)$ from $\mathcal{S}$: Directly forwards (sid, K) to $\mathcal{P}_i$.

Moreover, $\mathcal{S}$ would issue NewKey query to $\mathcal{F}$ once there is an output from the simulated parties. This hybrid is perfectly indistinguishable from $\mathbf{Hybrid_1}$ since the behaviour of $\mathcal{F}$ does not affect the view of $\mathcal{A}$, nor the output of $\mathcal{P}_{\{0,1\}}$.

Hybrid 3: Choosing Keys for Honest Parties with Close Passwords. The behaviour of $\mathcal{S}$ remains the same, whereas the functionality $\mathcal{F}$ modifies the NewKey interface as follows:

- $(\textsc{NewKey}, \mathsf{sid}, \mathcal{P}_i, K)$ from $\mathcal{S}$:
  - If both parties are honest and $\mathsf{dist}(\mathsf{pw}_0, \mathsf{pw}_1) \leq \delta$, do: Set $K_i = K_{1-i}$ if a key K_{1-i} was sent to $\mathcal{P}_{1-i}$ already, and set $K_i \leftarrow\!\!\$\, \mathcal{K}$ otherwise; Output (sid, K_i) to $\mathcal{P}_i$.
  - Otherwise, forwards (sid, K) to $\mathcal{P}_i$.

This hybrid is perfectly indistinguishable from $\mathbf{Hybrid_2}$ according to the functionality $\mathcal{F}_{\mathsf{OPRF}}$ and the correctness of OKVS. Since the input (resp. output) to (resp. from) ideal functionalities are hidden for the adversary $\mathcal{A}$ when both parties are honest, $\mathcal{A}$ can only observe the transcripts between parties. The only transcripts $\mathbf{e}, \mathbf{r}$ (in $\Pi^{L_\infty}_{\mathsf{RFE}}$ as well as in this hybrid) leak no information about R, S since they are perfectly hidden by the random function F, G. The only effect of the modification of $\mathcal{F}$ from $\mathbf{Hybrid_2}$ is the output of $\mathcal{P}_{\{0,1\}}$, from the view of $\mathcal{Z}$. Given a perfectly correct OKVS, each party would recover the other party's randomness share R_i and S_i correctly. Therefore, both parties output K in the end, and it is uniformly random for $\mathcal{Z}$ with the knowledge of $\mathbf{e}, \mathbf{r}$.

Hybrid 4: Choosing Keys for Honest Parties with Far Passwords. The behaviour of $\mathcal{S}$ remains the same, whereas the functionality $\mathcal{F}$ modifies the NewKey interface from above to:

- $(\textsc{NewKey}, \mathsf{sid}, \mathcal{P}_i, K)$ from $\mathcal{S}$:

- If both parties are honest: Set $K' = K_{1-i}$ if $\mathsf{dist}(\mathsf{pw}_0, \mathsf{pw}_1) \leq \delta$ *and* a key K_{1-i} was sent to $\mathcal{P}_{1-i}$ already, otherwise, set $K' \leftarrow\!\!\$\, \mathcal{K}$; Output (sid, K') to $\mathcal{P}_i$.
- Otherwise, forwards (sid, K) to $\mathcal{P}_i$.

This hybrid is perfectly indistinguishable from **Hybrid$_3$**. Note that when $\mathsf{dist}(\mathsf{pw}_0, \mathsf{pw}_1) > \delta$, there must be some $i \in [n]$ such that $v_i \notin [u_i - \delta, u_i + \delta]$ where u_i and v_i are the i-th character of pw_0 and pw_1. Therefore, g_i would be uniformly random (and independent of S) according to $\mathcal{F}_{\mathsf{OPRF}}$, making two parties' final keys uniformly random and independent of each other.

Hybrid 5: Stop Forwarding Passwords and Keys When Both Parties are Honest. The simulator $\mathcal{S}$ would issue NewKey queries with an empty key $K = \perp$ when both parties are honest. Also, the functionality $\mathcal{F}$ stops forwarding passwords to $\mathcal{S}$ by modifying NewSession to be the same as in Figure 4, that is, only forwarding $\big(\textsc{NewSession}, \mathsf{sid}, \mathcal{P}_{\{0,1\}}\big)$ to $\mathcal{S}$ when both parties are honest. Since the output keys of honest parties are already handled by $\mathcal{F}$ from above hybrids, this change is not observable by the environment $\mathcal{Z}$. This hybrid is therefore perfectly indistinguishable from **Hybrid$_4$**.

Hybrid 6: Extraction of the Malicious Input When One of the Parties is corrupt. We consider the static corruption, where the $\mathcal{S}$ knows at the beginning which party is corrupt. Remember $\mathcal{P}_{\mathsf{cor}}$ denotes the corrupt party. $\mathcal{S}$ sets $\mathsf{pw}_{\mathsf{cor}}$ as the input of the corrupt party $\mathcal{P}_{\mathsf{cor}}$ through NewSession:

- Once $\mathcal{S}$ observes the inputs and outputs of $\mathcal{F}_{\mathsf{OPRF}}$ where $\mathcal{P}_{\mathsf{cor}}$ is the receiver, it can extract the password $\mathsf{pw}_{\mathsf{cor}}$ as below;
- $\mathcal{S}$ sets the input for $\mathcal{P}_{\mathsf{cor}}$ by issuing $(\textsc{NewSession}, \mathsf{sid}, \mathsf{pw}_{\mathsf{cor}})$ to $\mathcal{F}$ on behalf of $\mathcal{P}_{\mathsf{cor}}$.

Note that the functionality has not yet implemented the NewKey interface when one of the parties is corrupt, thus the view of $\mathcal{Z}$ is exactly the same as before.

Now we show how the extraction works. After observing x_i sent to $\mathcal{F}_{\mathsf{OPRF}}$ where the corrupt party is the i-th receiver, $\mathcal{S}$ would search for histories of $\mathcal{F}_{\mathsf{RO}}$ until there is a record $(i \,\|\, u_i, x_i) \in \mathcal{L}_{\mathsf{RO}}$ for each $i \in [n]$. If such pair (u_i, x_i) exists for every $i \in [n]$, $\mathcal{S}$ sets $\mathsf{pw}_{\mathsf{cor}} = \mathbf{u}$; otherwise sets $\mathsf{pw}_{\mathsf{cor}} = \perp$. Notably, it does not matter whether $\mathcal{P}_{\mathsf{cor}}$ inputs the same password to $\mathcal{F}_{\mathsf{OPRF}}$ and the OKVS, as we show below.

Hybrid 7: Choosing the Key When One Party is Corrupt. Now $\mathcal{F}$ fully implement the NewKey interface, that is, it generates the output keys precisely following Figure 4. There are only two cases we need to consider:

- The extraction is succeed, $\mathsf{pw}_{\mathsf{cor}} \neq \perp$, and $\mathsf{dist}(\mathsf{pw}_{\mathsf{hon}}, \mathsf{pw}_{\mathsf{cor}}) > \delta$;
- The extraction is failed, $\mathsf{pw}_{\mathsf{cor}} = \perp$.

If the extraction succeeds, we don't need to consider the case where passwords are close to each other, since $\mathcal{F}$ would forward the key from $\mathcal{S}$. Specifically, since the record of $\mathcal{P}_{\mathsf{hon}}$ remains fresh, when given close passwords $\mathcal{F}$ simply forwards the key from $\mathcal{S}$, which is currently running $\mathcal{P}_{\mathsf{hon}}$'s code inside, and the resulting key is exactly the same as in the real world.

In the first case, there is at least one $i \in [n]$ such that $v_i \notin [u_i - \delta, u_i + \delta]$ resulting in a uniformly random g_i according to $\mathcal{F}_{\mathsf{OPRF}}$ (assuming the $\mathcal{P}_{\mathsf{cor}}$ holds $\mathbf{u}$ and R). Thus, the real randomness S and the final key K_1 of $\mathcal{P}_{\mathsf{hon}}$ are still perfectly hidden and uniformly random from the view of $\mathcal{P}_{\mathsf{cor}}$ and $\mathcal{Z}$.

In the second case, the probability of finding a correct x_i such that $x_i \in \{y_{i,-\delta}, \ldots, y_{i,+\delta}\}$ (for the specific $i \in [n]$ which makes the extraction failed) without querying $\mathcal{F}_{\mathsf{RO}}$ is negligible. Then according to $\mathcal{F}_{\mathsf{OPRF}}$, without knowing the correct x_i, the real evaluated values $\{\mathsf{G}_i(x_{i,j})\}_{j \in [-\delta, +\delta]}$

and real randomness S_i are still uniformly random. This implies the final key is statistically close to uniformly random and hidden from the view of $\mathcal{P}_{\mathsf{cor}}$ and $\mathcal{Z}$.

Hybrid 8: Simulating the Transcripts of the Honest Party. $\mathcal{S}$ only needs to simulate $\mathbf{r}$ from $\mathcal{P}_{\mathsf{hon}}$ as which is the only transcripts. Specifically, $\mathcal{S}$ is modified in this hybrid as follows.

- $\mathcal{S}$ sets the input for $\mathcal{P}_{\mathsf{cor}}$ by invoking NEWSESSION with the extracted $\mathsf{pw}_{\mathsf{cor}}$ as in $\mathbf{Hybrid_6}$;
- $\mathcal{S}$ sets $\mathbf{r}$ with $\mathsf{Encode}(\mathsf{list}')$ where list' is a list of size $n(2\delta+1)$ with random key-value pairs;
- Observing $x_1, \ldots, x_n$ and $g_1, \ldots, g_n$ as the queries sent to (and the responses from) $\mathcal{F}_{\mathsf{OPRF}}$ from $\mathcal{P}_{\mathsf{cor}}$, $\mathcal{S}$ sets
$$S := \sum_{i=1}^{n} \mathsf{Decode}(\mathbf{r}, x_i) - g_i$$
for $\mathcal{P}_{\mathsf{hon}}$;
- $\mathcal{S}$ follows the real protocol of $\mathcal{P}_{\mathsf{hon}}$ (by using $\mathsf{pw}_{\mathsf{hon}}$) to get S' from $\mathcal{P}_{\mathsf{cor}}$;
- $\mathcal{S}$ sets $K_1 := S+S'$, and outputs the final key for the honest party by issuing $(\text{NEWKEY}, \mathsf{sid}, \mathcal{P}_{\mathsf{hon}}, K_1)$ to $\mathcal{F}$.

This modification is statistically indistinguishable from the previous hybrid and thus unobservable to $\mathcal{Z}$. According to the definition of $\mathcal{F}_{\mathsf{OPRF}}$, it does not allow offline evaluation for CLIENT, which means $\mathcal{P}_{\mathsf{cor}}$ (and hence $\mathcal{A}$) can only get at most one $\mathsf{G}_i(x_i)$ for each $i \in [n]$. Then we argue the simulated $\mathbf{r}$ is statistically indistinguishable from the real $\mathbf{r}$ in the real protocol (and in $\mathbf{Hybrid_7}$): For each $i \in [n]$, if $x_i \in \mathsf{list}'$ in the real protocol, then $\mathcal{A}$ decodes the OKVS correctly to get S_i which is uniformly random; if $x_i \notin \mathsf{list}'$, then $\mathcal{A}$ gets a uniformly random g_i whereas $\mathsf{G}_i(y_{i,j})$ are still perfectly hidden. Therefore, list' in the real protocol is composed of uniformly random values, even given $g_1, \ldots, g_n$. This implies $\mathbf{r}$ is statistically indistinguishable between $\mathbf{Hybrid_7}$ and this hybrid due to the obliviousness property of OKVS.

Moreover, if two parties' passwords are close, S' and thus K_1 set by $\mathcal{S}$ is consistent with the previous hybrid; if they are far from each other, the final key K_1 of $\mathcal{P}_{\mathsf{hon}}$ is handled by the $\mathcal{F}$ as in $\mathbf{Hybrid_7}$. Thus the output of $\mathcal{P}_{\mathsf{hon}}$ is identical in two hybrids.

Hybrid 9: Removing Password Forwarding for the Honest Party. Now $\mathcal{F}$ stops forwarding $\mathsf{pw}_{\mathsf{hon}}$ from the honest party $\mathcal{P}_{\mathsf{hon}}$ to $\mathcal{S}$, that is, it fully implements NEWSESSION interfaces, and $\mathcal{F}$ fully implements the TESTPWD interface as in Figure 4. $\mathcal{S}$ learns $\mathsf{pw}_{\mathsf{hon}}$ by issuing a $(\text{TESTPWD}, \mathsf{sid}, \mathcal{P}_{\mathsf{hon}})$ query if they are $\mathsf{dist}(\mathsf{pw}_{\mathsf{hon}}, \mathsf{pw}_{\mathsf{cor}}) \leq \delta$. If $\mathcal{S}$ does not obtain a password from TESTPWD query, it directly sends $(\text{NEWKEY}, \mathsf{sid}, \mathcal{P}_{\mathsf{hon}}, K)$ to $\mathcal{F}$ with $K \leftarrow\!\!\$\ \mathcal{K}$. Note that $\mathcal{F}$ in this hybrid is identical to the ideal functionality $\mathcal{F}_{\mathsf{RFE}}^{L_\infty}$.

This change is not observable by $\mathcal{Z}$: when $\mathsf{dist}(\mathsf{pw}_{\mathsf{hon}}, \mathsf{pw}_{\mathsf{cor}}) \leq \delta$, this hybrid is identical to $\mathbf{Hybrid_8}$; otherwise, $\mathcal{F}$ outputs a random key as in $\mathbf{Hybrid_7}$.

□

C.2 The Proof of Lemma 2

Proof. The proof strategy closely follows that of Lemma 1, and the first three hybrids ($\mathbf{Hybrid_{0,1,2}}$) are exactly the same. Throughout, we use $R := \sum_{i=1}^{n} \mathsf{Decode}(\mathbf{r}, x_i) - g_i$ and $S := \sum_{i=1}^{n} \mathsf{Decode}(\mathbf{e}, y_i) - f_i$ as in Figure 9.

Hybrid 0: Real Game. This is the real execution of $\Pi_{\mathsf{RFE}}^{L_p}$, where the environment $\mathcal{Z}$ interacts with parties $\mathcal{P}_0$ and $\mathcal{P}_1$, both of which have access to the ideal functionalities $\mathcal{F}_{\mathsf{OPRF}}$ and $\mathcal{F}_{\mathsf{RO}}$. There is also a dummy adversary $\mathcal{A}$ that is fully controlled by $\mathcal{Z}$. We denote the corrupt party $\mathcal{P}_0$ as $\mathcal{P}_{\mathsf{cor}}$ and the honest party as $\mathcal{P}_{\mathsf{hon}}$.

Hybrid 1: Simulator and Dummy Functionality. This hybrid is identical to the real game, except the entire protocol is executed within a single machine, called the simulator $\mathcal{S}$. There is also a dummy (i.e., empty) functionality $\mathcal{F}$. The simulation among $\mathcal{F}, \mathcal{S}, \mathcal{P}_{\{0,1\}}$ proceeds as follows:

- $\mathcal{S}$ simulates $\mathcal{F}_{\mathsf{RO}}$ and $\mathcal{F}_{\mathsf{OPRF}}$ by running their code directly;
- The honest party $\mathcal{P}_{\mathsf{hon}}$ simply sends its input to and receives its output from the dummy functionality $\mathcal{F}$;
- $\mathcal{F}$ just forwards inputs from $\mathcal{P}_{\mathsf{hon}}$ to $\mathcal{S}$, and outputs from $\mathcal{S}$ to $\mathcal{P}_{\mathsf{hon}}$;
- $\mathcal{S}$ runs the code for each honest party and outputs the result to $\mathcal{F}$.

This hybrid is perfectly indistinguishable from $\mathbf{Hybrid_0}$ since the view of $\mathcal{Z}$ is unchanged.

Hybrid 2: Adding NewSession, NewKey Interfaces. Now the functionality $\mathcal{F}$ implements NewSession and NewKey interfaces:

- $(\textsc{NewSession}, \mathsf{sid}, \mathsf{pw}_i)$ from $\mathcal{P}_i$: As in Figure 4, but also forwards pw_i to $\mathcal{S}$.
- $(\textsc{NewKey}, \mathsf{sid}, \mathcal{P}_i, K)$ from $\mathcal{S}$: Forwards (sid, K) directly to $\mathcal{P}_i$.

Moreover, $\mathcal{S}$ issues a NewKey query to $\mathcal{F}$ whenever there is an output from the simulated parties. This hybrid is perfectly indistinguishable from $\mathbf{Hybrid_1}$, as the behavior of $\mathcal{F}$ does not affect the view of $\mathcal{A}$ or the outputs of $\mathcal{P}_{\{0,1\}}$.

Hybrid 3: Choosing Keys for Honest Parties with Close Passwords. The behavior of $\mathcal{S}$ remains unchanged, but the functionality $\mathcal{F}$ modifies the NewKey interface as follows:

- $(\textsc{NewKey}, \mathsf{sid}, \mathcal{P}_i, K)$ from $\mathcal{S}$:
  - If both parties are honest and $\mathsf{dist}(\mathsf{pw}_0, \mathsf{pw}_1) \leq \delta$: Set $K' = K_{1-i}$ if a key K_{1-i} was already sent to $\mathcal{P}_{1-i}$, and set $K' \leftarrow\!\!\$\ \mathcal{K}$ otherwise; Output (sid, K') to $\mathcal{P}_i$.
  - Otherwise, forward (sid, K) to $\mathcal{P}_i$.

This hybrid is statistically indistinguishable from $\mathbf{Hybrid_2}$ if the OKVS is perfectly correct as defined in Section 3.1 and H_T is modeled as a random oracle. It is natural that both parties output the same key K when their passwords are close, since the distance function dist_{L_p} is symmetric and they will obtain the same index $t \in [0, \delta^p]$.

We now argue that such a key is uniformly random from the perspective of $\mathcal{Z}$, making the view indistinguishable from that in $\mathbf{Hybrid_2}$. Since the inputs and outputs to the ideal functionalities are hidden from the adversary $\mathcal{A}$ when both parties are honest, $\mathcal{A}$ can only observe the transcripts between parties. However, the only transcripts in $\Pi^{L_p}_{\mathsf{RFE}}$ and in this hybrid are the sets and encodings $T_0, \mathbf{e}$ and $T_1, \mathbf{r}$. T_0 can be seen as a point function obfuscation of β and α, while $\mathbf{e}$ reveals nothing about α, β since the values are masked by $\mathsf{F}(x_{i,j})$. Therefore, the distribution of β given $T_0, \mathbf{e}$ is statistically close to uniform. Regarding the outputs of $\mathcal{P}_{\{0,1\}}$, from the view of $\mathcal{Z}$, they are uniformly random up to a negligible statistical distance.

Hybrid 4: Choosing Keys for Honest Parties with Far Passwords. The behavior of $\mathcal{S}$ remains unchanged, but the functionality $\mathcal{F}$ further modifies the NewKey interface:

- $(\textsc{NewKey}, \mathsf{sid}, \mathcal{P}_i, K)$ from $\mathcal{S}$:
  - If both parties are honest: Set $K' = K_{1-i}$ if $\mathsf{dist}(\mathsf{pw}_0, \mathsf{pw}_1) \leq \delta$ and a key K_{1-i} was already sent to $\mathcal{P}_{1-i}$; otherwise, set $K' \leftarrow\!\!\$\ \mathcal{K}$; Output (sid, K') to $\mathcal{P}_i$.
  - Otherwise, forward (sid, K) to $\mathcal{P}_i$.

This hybrid is statistically indistinguishable from $\mathbf{Hybrid_3}$. Consider the case when $\mathsf{dist}(\mathsf{pw}_0, \mathsf{pw}_1) > \delta$. Let u_i and v_i denote the i-th characters of pw_0 and pw_1, respectively. There are two possibilities:

– There exists some $i \in [n]$ such that $v_i \notin [u_i - \delta, u_i + \delta]$;
– For every $i \in [n]$, $v_i \in [u_i - \delta, u_i + \delta]$.

In the first case, by the properties of $\mathcal{F}_{\mathsf{OPRF}}$, f_i is uniformly random, so the decoded value $S = \sum_{i=1}^{n} \mathsf{Decode}(\mathbf{e}, y_i) - f_i$ is uniformly random and $\mathsf{H}_T(S \,\|\, \cdot)$ will be in T_1 only with negligible probability.

In the second case, $S = \alpha \cdot t + \beta$ for some $t \notin [0, \delta^p]$, and again, $\mathsf{H}_T(S \,\|\, t)$ lands in T_1 only with negligible probability.

Therefore, in both cases, each party outputs a uniformly random key $K \leftarrow\!\!\$\, \mathcal{K}$, which matches the behavior specified by the functionality in this hybrid.

Hybrid 5: Stop Forwarding Passwords and Keys When Both Parties are Honest. In this hybrid, the simulator $\mathcal{S}$ issues NewKey queries with an empty key $K = \perp$ whenever both parties are honest. Additionally, the functionality $\mathcal{F}$ stops forwarding passwords to $\mathcal{S}$ by updating the NewSession interface to match Figure 4, meaning it only forwards $(\text{NewSession}, \mathsf{sid}, \mathcal{P}_{\{0,1\}})$ to $\mathcal{S}$ when both parties are honest. Since the final keys of honest parties are already handled by $\mathcal{F}$ from the above hybrids, this change is not observable by the environment $\mathcal{Z}$. This hybrid is perfectly indistinguishable from $\mathbf{Hybrid_4}$.

Hybrid 6: Extraction of the Malicious Input When One of the Parties is Corrupt. We consider static corruption, where $\mathcal{S}$ knows at the outset which party is corrupt. Recall that $\mathcal{P}_{\mathsf{cor}}$ denotes the corrupt party. $\mathcal{S}$ sets $\mathsf{pw}_{\mathsf{cor}}$ as the input of the corrupt party $\mathcal{P}_{\mathsf{cor}}$ via NewSession:

– $\mathcal{S}$ extracts $\mathsf{pw}_{\mathsf{cor}}$ from $\mathcal{F}_{\mathsf{OPRF}}$ as in the proof of Lemma 1.
– $\mathcal{S}$ sets the input for $\mathcal{P}_{\mathsf{cor}}$ by issuing $(\text{NewSession}, \mathsf{sid}, \mathsf{pw}_{\mathsf{cor}})$ to $\mathcal{F}$ on behalf of $\mathcal{P}_{\mathsf{cor}}$.

Note that the functionality has not yet implemented the NewKey interface when one of the parties is corrupt, so the view of $\mathcal{Z}$ is exactly the same as before.

Hybrid 7: Choosing the Key When One Party is Corrupt. Now $\mathcal{F}$ fully implements the NewKey interface, generating output keys exactly as in Figure 4. There are only two cases to consider:

– The extraction succeeds, $\mathsf{pw}_{\mathsf{cor}} \neq \perp$, and $\mathsf{dist}(\mathsf{pw}_{\mathsf{hon}}, \mathsf{pw}_{\mathsf{cor}}) > \delta$;
– The extraction fails, $\mathsf{pw}_{\mathsf{cor}} = \perp$.

If the extraction succeeds, we do not need to consider cases where passwords are close within the threshold, or both parties are corrupt, since $\mathcal{F}$ would forward the key from $\mathcal{S}$. In the setting that distance is greater than δ, we consider two cases depending on whether the infinity distance $d_\infty \leftarrow \mathsf{dist}_{L_\infty}(\mathsf{pw}_{\mathsf{hon}}, \mathsf{pw}_{\mathsf{cor}})$ is smaller than δ.

– When $d_\infty > \delta$, the values ξ and ζ are perfectly hidden, and the final key K_1 is uniformly random from the perspective of $\mathcal{P}_{\mathsf{cor}}$ and $\mathcal{Z}$, for the same reason as in the L_∞ setting; see the proof of Lemma 1.
– Otherwise, even if $\mathcal{P}_{\mathsf{cor}}$ obtains the correct decoded values $\xi \cdot j_i^p + \zeta_i$ for each coordinate i and some shift j_i, the value α remains perfectly hidden due to the fresh mask ζ_i in each coordinate. Consequently, for any $t > \delta^p \geq 0$, the value $\xi \cdot t$ is uniformly random as well. Hence, the final key K_1 is uniformly random and hidden from the view of $\mathcal{P}_{\mathsf{cor}}$ and $\mathcal{Z}$.

In the second case, the probability of finding a correct $\mathbf{x}$ without querying $\mathcal{F}_{\mathsf{RO}}$ is negligible. Then, according to $\mathcal{F}_{\mathsf{OPRF}}$, the final key is statistically close to uniformly random and hidden from the view of $\mathcal{P}_{\mathsf{cor}}$ and $\mathcal{Z}$, for the same reason as in the infinity distance setting.

Hybrid 8: Simulating the Transcripts of the Honest Party. In this hybrid, $\mathcal{S}$ only needs to simulate $\mathbf{r}$ and T_1 for the honest party $\mathcal{P}_{\mathsf{hon}}$. Specifically, $\mathcal{S}$ is modified as follows:

– $\mathcal{S}$ sets the input for $\mathcal{P}_{\mathsf{cor}}$ by invoking NEWSESSION with the extracted $\mathsf{pw}_{\mathsf{cor}}$ as in **Hybrid$_6$**;
– $\mathcal{S}$ sets T_1 as a list of random elements;
– $\mathcal{S}$ samples $\mathbf{r} \leftarrow \mathsf{Encode}(\mathsf{list}')$, where list' is a list of $n(2\delta+1)$ random key-value pairs;
– Observing $x_1, \ldots, x_n$ and $g_1, \ldots, g_n$ as the queries sent to (and the responses from) $\mathcal{F}_{\mathsf{OPRF}}$ from $\mathcal{P}_{\mathsf{cor}}$, $\mathcal{S}$ sets

$$R := \sum_{i=1}^{n} \mathsf{Decode}(\mathbf{r}, x_i) - g_i$$

for $\mathcal{P}_{\mathsf{hon}}$;
– $\mathcal{S}$ programs $\mathcal{F}_{\mathsf{RO}}$ so that T_1 is consistent with $\mathsf{H}_T(R \,\|\, d^p)$ if $d \le \delta$, where $d \leftarrow \mathsf{dist}(\mathsf{pw}_{\mathsf{hon}}, \mathsf{pw}_{\mathsf{cor}})$;
– $\mathcal{S}$ follows the real protocol of $\mathcal{P}_{\mathsf{hon}}$ (using $\mathsf{pw}_{\mathsf{hon}}$) to compute

$$S := \sum_{i=1}^{n} \mathsf{Decode}(\mathbf{e}, y_i) - \mathsf{F}(y_i)$$

from $\mathcal{P}_{\mathsf{cor}}$'s message $(\mathbf{e}, T_0)$;
– $\mathcal{S}$ sets $K_1 := R + S$ if there is t such that$\mathsf{H}_T(S \,\|\, t) \in T_0$ and $t = d^p$, otherwise sets $K_1 \leftarrow\!\!\$\ \mathcal{K}$;
– $\mathcal{S}$ outputs the final key for the honest party by issuing $(\text{NEWKEY}, \mathsf{sid}, \mathcal{P}_{\mathsf{hon}}, K_1)$ to $\mathcal{F}$.

Note that $\mathbf{r}$ in this hybrid is indistinguishable from **Hybrid$_7$** for the same reason as in the infinity setting (see the proof of Lemma 1), and T_1 is indistinguishable since H_T is modeled as a random oracle.

We now argue that the modification in this hybrid is unobservable by $\mathcal{Z}$: If the passwords of the two parties are far (beyond the threshold), then $\mathcal{F}$ handles the final key and the value K_1 in the NEWKEY interface is ignored; if the passwords are close, for $d \le \delta$, we consider:

– If $t = d^p$, it means T_0 is consistently generated by $\mathcal{P}_{\mathsf{cor}}$. The simulated $R = \xi \cdot d^p + \zeta = \xi \cdot t + \zeta$, so the final key in **Hybrid$_7$** is identical to that in this hybrid;
– If $t \neq d^p$, then $\xi \cdot t + \zeta$ in **Hybrid$_7$** is still uniformly random, and has the same distribution as K_1 in this hybrid.

Hybrid 9: Removing Password Forwarding for the Honest Party. Now $\mathcal{F}$ stops forwarding $\mathsf{pw}_{\mathsf{hon}}$ from the honest party $\mathcal{P}_{\mathsf{hon}}$ to $\mathcal{S}$, i.e., it fully implements the NEWSESSION interfaces, and $\mathcal{F}$ fully implements the TESTPWD interface as in Figure 4. $\mathcal{S}$ learns $\mathsf{pw}_{\mathsf{hon}}$ by issuing a $(\text{TESTPWD}, \mathsf{sid}, \mathcal{P}_{\mathsf{hon}})$ query if $\mathsf{dist}(\mathsf{pw}_{\mathsf{hon}}, \mathsf{pw}_{\mathsf{cor}}) \le \delta$. If $\mathcal{S}$ does not obtain a password from a TESTPWD query, it directly sends $(\text{NEWKEY}, \mathsf{sid}, \mathcal{P}_{\mathsf{hon}}, K_1)$ to $\mathcal{F}$ with $K_1 \leftarrow\!\!\$\ \mathcal{K}$. Note that $\mathcal{F}$ in this hybrid is identical to the ideal functionality $\mathcal{F}_{\mathsf{RFE}}^{L_p, \mathsf{pw}}$.

This change is not observable by $\mathcal{Z}$: when $\mathsf{dist}(\mathsf{pw}_{\mathsf{hon}}, \mathsf{pw}_{\mathsf{cor}}) \le \delta$, this hybrid is identical to **Hybrid$_8$**; otherwise, $\mathcal{F}$ outputs a random key as in **Hybrid$_7$**. □