Efficient Fuzzy Matching and Intersection on Private Datasets

Qingsong Ye[1], Ron Steinfeld[1], Josef Pieprzyk[1], and Huaxiong Wang[1,2]

[1] Centre for Advanced Computing – Algorithms and Cryptography
Department of Computing, Macquarie University, NSW 2109, Australia
{qingsong,rons,josef}@ics.mq.edu.au

[2] Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
hxwang@ntu.edu.sg

Abstract. At Eurocrypt'04, Freedman, Nissim and Pinkas introduced a fuzzy private matching problem. The problem is defined as follows. Given two parties, each of them having a set of vectors where each vector has T integer components, the fuzzy private matching is to securely test if each vector of one set matches any vector of another set for at least t components where $t < T$. In the conclusion of their paper, they asked whether it was possible to design a fuzzy private matching protocol without incurring a communication complexity with the factor $\binom{T}{t}$. We answer their question in the affirmative by presenting a protocol based on homomorphic encryption, combined with the novel notion of a *share-hiding error-correcting secret sharing scheme*, which we show how to implement with efficient decoding using interleaved Reed-Solomon codes. This scheme may be of independent interest. Our protocol is provably secure against passive adversaries, and has better efficiency than previous protocols for certain parameter values.

Keywords: private matching, private set intersection, fuzzy private matching, homomorphic encryption, error correction, secret sharing

1 Introduction

In Eurocrypt'04, Freedman, Nissim and Pinkas (FNP) [4] introduced the private fuzzy matching problem. The problem is defined for two parties. Each party holds a set of vectors, where each vector has its length equal to T. The number of vectors in the two sets are m and n, respectively. The fuzzy private matching of the two sets computes the intersection of two sets by considering a match if any pair of vectors from both sets has at least t out of T common components ($t < T$). The computation must preserve the privacy of the sets, i.e. the other party learns no more than the result of the operation.

This error-tolerance property is useful in many applications. For example, database entries are not always accurate or full (e.g. due to errors, omissions, or inconsistent spellings), for example, in the case of biometric pattern matching. Due to the human error and error-prone biometric systems, it would be useful to have an algorithm still reporting a match if two datasets are similar within a threshold.

In [4], Freedman, Nissim and Pinkas gave a simple 2-out-of-3 fuzzy private matching protocol. However, this protocol is not efficient as it requires $O\left(m\binom{T}{t}\right)$ communication complexity and $O\left(mn\binom{T}{t}\right)$ computation complexity. As an open problem, they posed the question of how to construct the private fuzzy matching without incurring a communication complexity with the exponential $\binom{T}{t}$ factor.

Recently, Chmielewski and Hoepman [3] proposed two fuzzy matching protocols with polynomial communication complexity, but at the expense of an exponential $\binom{T}{t}$ factor in the *computation* complexity. We show how to further improve one of these protocols (to be called CH1), making *both* communication and computation polynomial in T. We also show that the second protocol in [3], to be called CH2 (which is claimed to have even better communication complexity) is insecure.

Our solution is based on polynomial encoding and a *share-hiding random error-correcting threshold secret sharing scheme* based on interleaved Reed-Solomon codes.

We first explain the notion of an error-correcting secret sharing scheme. In an ordinary t-of-T secret sharing scheme, the secret can be efficiently recovered from any t shares. However, if one is given a 'noisy' vector of T shares, out of which only t shares are correct and the rest are random values, one may have to try all $\binom{T}{t}$ subsets of t shares until the correct subset is found and the secret is recovered (assuming that the correct secret can be identified). The idea of an error-correcting threshold t-of-T secret sharing scheme is to add additional redundancy to the shares of the secret, such that the correct secret can be efficiently recovered (in time polynomial in T) even in the 'noisy' setting above, where an unknown subset of t of T given shares are correct and the rest are random. At the same time, we also require a *share hiding* privacy property: when there are $< t$ correct shares the above 'noisy' vector of T shares gives no information on the position of the correct shares. This problem naturally leads to consider error correcting codes to perform this decoding. As we explain, although the Shamir t-of-T secret sharing scheme can also be viewed as a Reed Solomon error correcting code, it does *not* quite achieve the goal (since it requires at least $\sqrt{Tt}$ correct shares for efficient decoding, which may be much larger than t). We show how to modify the Shamir scheme into an error-correcting secret sharing scheme by using the concept of interleaved Reed Solomon codes.

Given our share hiding error-correcting secret sharing scheme, the idea of our protocol (based on the CH1 protocol) is to let one party, Alice, send to the other party Bob encryptions of her database elements using a homomorphic encryption scheme. Using the homomorphic property, Bob can compute the ciphertext of the *difference vector* between each pair of Alice's and Bob's database words. Bob then homomorphically adds this difference vector to the shares vector of an encryption key, created using the error-correcting secret sharing scheme, and sends the resulting ciphertexts to Alice. As a result, Alice's decryption consists of share vectors having correct shares of Bob's key in the positions where Alice's word matched Bob's word, and, if there are at least t matches, Alice can use the error-correcting property to recover Bob's key (which is then used by Alice to decrypt a ciphertext of Bob's matching word). In order to hide the non-matching elements of Bob, we utilize the randomization technique used in the original FNP private matching protocol (if the element of the Alice dataset is different

from the element of the Bob dataset, then this element will be multiplied by a random number). Moreover the share hiding property also hides the location of the matching elements when there are less than t matches. We remark that the original CH1 protocol in [3] did not make use of an error correcting secret sharing scheme, which forced an exponential search by Alice in decoding.

We prove the security of our protocol against passive attacks and explain its efficiency advantages relative to previous protocols.

1.1 Private Matching and Set Intersection in FNP

We briefly review the (not fuzzy) private matching and set intersection in [4], since it is the basis and the extension of the private fuzzy matching discussed in the same paper.

Polynomial Representation of Datasets and Private Matching. Let $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a semantically-secure public-key cryptosystem with additive homomorphic properties, such as Paillier's [9]. Recall that, given $\mathcal{E}(a)$, $\mathcal{E}(b)$ and a constant c, one can compute $\mathcal{E}(a+b) = \mathcal{E}(a) \odot \mathcal{E}(b)$ and $\mathcal{E}(a \cdot c) = \mathcal{E}(a)^c$.

There are two parties in the protocol, namely, Alice and Bob. Bob owns a value b, while Alice possesses a dataset $A' = \{a_1, \ldots, a_m\}$ and wants to test if $b \in A'$ or not. Alice does not want to reveal A to Bob, and Bob is unwilling to disclose b to Alice.

The protocol runs as follows.

- Alice first presents her dataset A' in the form of a polynomial

$$\mathcal{P}(y) = \prod_{a_i \in A'} (y - a_i) = \sum_{i=0}^{m} \alpha_i y^i, \text{ where } \alpha_m = 1$$

- Applied in the homomorphic encryption. Alice encrypts her polynomial $\mathcal{P}$ with her public-key. Note that the encrypted polynomial $\mathcal{E}(\mathcal{P})$ contains the encryptions of all coefficients α_i except α_m. Next she sends $\mathcal{E}(\mathcal{P})$ to Bob.
- Using the homomorphic properties, Bob evaluates the polynomial for his input b according to the following formula

$$\mathcal{E}(\mathcal{P}(b)) = \mathcal{E}(\alpha_0) \odot \mathcal{E}(\alpha_1)^b \odot \mathcal{E}(\alpha_2)^{b^2} \odot \ldots \odot \mathcal{E}(\alpha_{m-1})^{b^{m-1}} \odot \mathcal{E}(1)^{b^m},$$

  and sends the result $\mathcal{E}(\gamma \mathcal{P}(b) + b)$ to Alice, where γ is a random non-zero integer. Note that $b \in A'$ if and only if $\mathcal{P}(b) = 0$.
- When Alice receives the cryptogram, she decrypts it and checks if the decrypted message belongs to the set A'. If it does she knows the value b, otherwise she knows a random value.

Private Computation of Set Intersection. Suppose Alice and Bob, each has a dataset $A' = \{a_1, \ldots, a_m\}$ and $B' = \{b_1, \ldots, b_n\}$ respectively, where the set cardinalities m and n are publicly known. Alice wishes to learn the intersection of two sets $A \cap B$. To compute the set intersection, we simply run the above private matching protocol m times in parallel for each of $b_j \in B$. In the end, Alice decrypts all the cryptograms and checks if each one is in A, and then establishes $A \cap B$.

1.2 Related Work on Fuzzy Private Matching

A simple 2-out-of-3 fuzzy matching protocol is given in [4]. We are going to call it the FNP protocol. Although it is flawed (for a detailed analysis refer to [3]), the approach seems to be sound. Alice has m 3-tuples $A_1, \ldots, A_m$, where $A_i = (a_{i1}, a_{i2}, a_{i3})$ for $i = 1, \ldots, m$. Let P_1, P_2, P_3 be polynomials, such that P_ℓ is used to encode the ℓ-th elements $(a_{1\ell}, \ldots, a_{m\ell})$ of the 3-tuples. For each $i = 1, \ldots, m$, Alice chooses a random value $\gamma_i = P_1(a_{i1}) = P_2(a_{i2}) = P_3(a_{i3})$. Note that for each polynomial P_ℓ; $\ell = 1, 2, 3$, there are m equations so the degree of the polynomials P_ℓ is at most $m - 1$.

Next Alice sends $(\mathcal{E}(P_1), \mathcal{E}(P_2), \mathcal{E}(P_3))$ to Bob in the form of encrypted coefficients as in Section 1.1.

For every 3-tuple B_j in his dataset of size n, Bob responds to Alice in a manner similar to the protocol in Section 1.1. He computes the encrypted values $\mathcal{E}(r \cdot (P_1(b_{j1}) - P_2(b_{j2})) + B_j)$, $\mathcal{E}(r' \cdot (P_2(b_{j2}) - P_3(b_{j3})) + B_j)$ and $\mathcal{E}(r'' \cdot (P_1(b_{j1}) - P_3(b_{j3})) + B_j)$ by encoding B_j as $b_{j1}||b_{j2}||b_{j3}$, where r, r' and r'' are random values. If two elements in A_i are the same as those in B_j, Alice obtains B_j in one of the entries after decrypting received ciphertexts.

The generalization of this approach for matching t our of T positions is possible but the resulting protocol is not going to be efficient. Clearly, for each B_j; $j = 1, \ldots, n$, Alice has to check all the combinations $\binom{T}{t}$ so both communication and computation complexities of the protocol have the factor $\binom{T}{t}$.

Chmielewski and Hoepman [3] extend the FNP protocol and propose two modified protocols that we call CH1 and CH2, both avoiding the $\binom{T}{t}$ factor in the communication complexity, but at the expense of a $\binom{T}{t}$ factor in computation. The protocol CH1 has quadratic complexity, while CH2 has linear complexity. Unfortunately, the protocol CH2 is insecure, as we explain below. Our work shows how to improve CH1 by further removing the $\binom{T}{t}$ factor from the computation. Both protocols CH1 and CH2, are achieved by combining secret sharing [12] and homomorphic encryption. The idea of the CH1 protocol (which forms the basis for our protocol) was already explained in the Introduction. Here we explain the CH2 protocol and why it is insecure.

CH2 Protocol. For each secret vector $B_j \in B$, Bob constructs t-out-of-T secret sharing that defines a collection of shares $(s_{j1}, \ldots, s_{jT})$. Note that, B_j is encoded as $b_{j1}||b_{j2}|| \ldots ||b_{jT}$ for a convenience. If $b_{j\ell} = b_{j'\ell}$, then $s_{j\ell} = s_{j'\ell}$ where $j \neq j'$. Bob also constructs T polynomials of degree n, $P_1, \ldots, P_T$ such that

$$((P_\ell(b_{1\ell}) = s_{1\ell}) \text{ and } (P_\ell(b_{2\ell}) = s_{2\ell}) \text{ and }, \ldots, \text{ and } (P_\ell(b_{n\ell}) = s_{n\ell})).$$

Bob sends all $\mathcal{E}(P_\ell)$ to Alice for $\ell = 1, \ldots, T$.

For $i = 1, \ldots, m$ and $\ell = 1, \ldots, T$, Alice computes $\mathcal{E}(P_\ell(a_{i\ell}))$ using homomorphic properties. Note that $P_\ell(a_{i\ell}) = s_{j\ell}$ if $a_{i\ell} = b_{j\ell}$. To hide the information about $a_{i\ell}$, Alice random selects a integer $r_{i\ell}$ and sends $\mathcal{E}(P_\ell(a_{i\ell}) + r_{i\ell})$ to Bob for $i \in \{1, \ldots, m\}$.

Assume that Bob does not want to reveal any information about $s_{j\ell}$. Bob decrypts $\mathcal{E}(P_\ell(a_{i\ell}) + r_{i\ell})$, and prepares t-out-of-T shares $(\hat{s}_{i1}, \ldots, \hat{s}_{iT})$ of a value 0 for $i = 1, \ldots, m$. Bob sends $P_\ell(a_{i\ell}) + r_{i\ell} + \hat{s}_{i\ell}$ to Alice. Note that $P_\ell(a_{i\ell}) + r_{i\ell} + \hat{s}_{i\ell} = s_{j\ell} + r_{i\ell} + \hat{s}_{i\ell}$ if $a_{i\ell} = b_{j\ell}$ for some j.

After receiving all the values from Bob, Alice computes $v_{i\ell} = (P_\ell(a_{i\ell}) + r_{i\ell} + \hat{s}_{i\ell}) - r_{i\ell}$ for all i and ℓ. For each $i = 1, \ldots, m$, Alice tries to computes A'_i from all $\binom{T}{t}$

combinations of $(v_{i1}, \ldots, v_{iT})$ by using t-out-of-T secret sharing scheme. If $A'_i \in_f A$, then Alice adds A'_i to her output set.

Attack on CH2 Protocol. We show that CH2 is insecure. Suppose that A_1 and A_2 are two words in Alice's dataset and B_1, B_2, B_3 are three words in Bob's dataset. Suppose that A_1 matches B_1 on $t-1$ letters in positions $1, ..., (t-1)$, matches B_2 on $t-1$ letters in positions $t, ..., 2t-2$, and matches B_3 on $t-1$ letters in positions $2t-1, ..., 3t-3$. Suppose further that A_2 matches B_2 on $t-1$ letters in positions $1, ..., (t-1)$, matches B_3 on $t-1$ letters in positions $t, ..., 2t-2$, and matches B_1 on $t-1$ letters in positions $2t-1, ..., 3t-3$.

The above condition implies that the shares $v_{i\ell}$ obtained by Alice in CH2, are related to Bob's shares $s_{j\ell}$ and $\hat{s}_{j\ell}$ as follows: $v_{1\ell} = s_{1\ell} + \hat{s}_{1\ell}$ for $\ell = 1, \ldots, t-1$, $v_{1\ell} = s_{2\ell} + \hat{s}_{1\ell}$ for $\ell = t, \ldots, 2t-2$, $v_{1\ell} = s_{3\ell} + \hat{s}_{1\ell}$ for $\ell = 2t-1, \ldots, 3t-3$, $v_{2\ell} = s_{2\ell} + \hat{s}_{2\ell}$ for $\ell = 1, \ldots, t-1$, $v_{2\ell} = s_{3\ell} + \hat{s}_{2\ell}$ for $\ell = t, \ldots, 2t-2$, $v_{2\ell} = s_{1\ell} + \hat{s}_{2\ell}$ for $\ell = 2t-1, \ldots, 3t-3$. Assume we are using t-of-T Shamir sharing for the 5 secret sharing vectors $\{s_{j\ell}\}_\ell, \{\hat{s}_{i\ell}\}_\ell$ for $j \in \{1,2,3\}$ and $i \in \{1,2\}$. Each sharing has a polynomial of degree $\leq t-1$ associated with it, so we have $5t$ random variables (coefficients) involved. On the other hand, the above relations give us overall $6(t-1)$ known linear equations in these random variables. For sufficiently large t, we have $6(t-1) > 5t$, which means we can find a linear dependency among the equations. The corresponding non-trivial linear combination of the $v_{i\ell}$'s will be zero, and this can be detected by Alice. On the other hand, for example, if the A_1 and A_2 don't match the $B_1, ...B_3$ in any position, the $v_{i\ell}$'s will be independent and uniformly random, so the tested non-trivial linear combination of them will be zero with negligible probability $1/p$. Hence the attack allows Alice to tell when the prescribed condition holds, which is a privacy leak (since the condition involves only $t-1 < t$ matches between any A_i and B_j).

2 Preliminaries

2.1 Additively Homomorphic Encryption

We will utilize an additive homomorphic public key cryptosystem, such as Paillier [9]. Following Adida and Wikstrom [1], we use the following definition.

Definition 1 ([1]). *A cryptosystem $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ defined by the key generator, encryption and decryption algorithms, respectively, is said to be homomorphic if for every key pair $(pk, sk) \in \mathcal{K}(1^l)$, the following conditions hold.*

1. *The message space $\mathcal{M}$ is a subset of an additive abelian group $\mathcal{G}(\mathcal{M})$.*
2. *The randomizer space $\mathcal{R}$ is an additive abelian group.*
3. *The ciphertext space is an multiplicative abelian group.*
4. *Given a public key pk, the group operations can be computed in polynomial time. For every $m, m' \in \mathcal{M}$ and $r, r' \in \mathcal{R}$, the following relation holds*

$$\mathcal{E}(m, r) \odot \mathcal{E}(m', r') = \mathcal{E}(m + m', r + r').$$

5. *The cryptosystem is said to be additive if the message space $\mathcal{M}$ is the additive modular group $\mathbb{Z}_n$ for some integer $n > 1$.*

When such operations are performed, we require that the resulting ciphertexts be re-randomized for security reasons. During such a process, the ciphertext e of the plaintext m is transformed into e' such that e' is still a valid cryptogram for the same message m but created with a different random string.

For simplicity, we use $\mathcal{E}(m)$ to represent $\mathcal{E}(r, m)$ in the rest of the presentation as we assume that there is always a corresponding random string r.

2.2 Definitions

We use the usual asymptotic notation O, o, Ω, ω. We say that a function $f(s)$ is *negligible*, denoted $f(s) = \mathrm{neg}(s)$, if $f(s) = 1/s^{\omega(1)}$. For two probability distributions D_1, D_2 parameterized by a security parameter s, we say that D_1 and D_2 are *computationally indistinguishable*, denoted $D_1 =_c D_2$, if any distinguisher A with run-time $O(\mathrm{poly}(s))$ has negligible distinguishing advantage $|\Pr_{x \leftarrow D_1}[A(x) = 1] - \Pr_{x \leftarrow D_2}[A(x) = 1]| = \mathrm{neg}(s)$. We say that D_1 and D_2 are *statistically indistinguishable*, denoted $D_1 =_s D_2$, if any distinguisher A with unbounded run-time has negligible distinguishing advantage.

Throughout this chapter, the computations are carried out over an arbitrary finite field $\mathcal{F}$. There are two parties Alice $\mathcal{I}_\mathcal{C}$ and Bob $\mathcal{I}_\mathcal{S}$. Let $A = \{A_1, \ldots, A_m\}$ and $B = \{B_1, \ldots, B_n\}$ be Alice's and Bob's datasets respectively. We call the dataset elements *words*, and assume that each word consists of an ordered list of T *letters* from $\mathcal{F}$, i.e. $A_i = (a_{i1}, \ldots, a_{iT}) \in \mathcal{F}^T$, $B_j = (b_{j1}, \ldots, b_{jT}) \in \mathcal{F}^T$.

Definition 2. *Given two words A_i and B_j defined as above and integer $t \leq T$, we say that A_i and B_j are* t-fuzzy equal*, written as $A_i \approx_t B_j$, if the words A_i and B_j agree on at least t letters, i.e. if*

$$|\{\ell : a_{i\ell} = b_{j\ell}\}| \geq t.$$

Definition 3. *Given two datasets A and B as defined above and integer $t \leq T$, the* t-fuzzy set intersection *of datasets A and B, denoted $A \cap_t B$ is defined as*

$$A \cap_t B = \{(A_i, B_j) | A_i \in A, B_j \in B, A_i \approx_t B_j\}.$$

Now we formally define the private fuzzy matching protocol. Let client Alice $\mathcal{I}_\mathcal{C}$ and server Bob $\mathcal{I}_\mathcal{S}$ be two probabilistic polynomial time interactive Turing machines. The interaction of $\mathcal{I}_\mathcal{C}$ and $\mathcal{I}_\mathcal{S}$ yields a result to Alice $\mathcal{I}_\mathcal{C}$ only (server Bob outputs nothing).

We use the standard definitions for passive security of two-party computation, adapted to the private fuzzy matching setting.

Definition 4. (Private Fuzzy Matching) *A protocol Π for two probabilistic polynomial time interactive Turing machines,* Client *$\mathcal{I}_\mathcal{C}$ and* Server *$\mathcal{I}_\mathcal{S}$, is said to be a* passive-secure *t-fuzzy matching protocol if it satisfies the following properties. The common input to both $\mathcal{I}_\mathcal{C}$ and $\mathcal{I}_\mathcal{S}$ is a security parameter s (implicit below). The private input of $\mathcal{I}_\mathcal{C}$ is dataset A and the private input of $\mathcal{I}_\mathcal{S}$ is dataset B.*

Completeness. *If both parties follow the protocol, then at the end of the protocol, with probability $\geq 1 - \mathrm{neg}(s)$ (over the random coins of $\mathcal{I}_\mathcal{C}$ and $\mathcal{I}_\mathcal{S}$), $\mathcal{I}_\mathcal{C}$ learns the t-fuzzy set intersection $A \cap_t B$.*

Security Against Passive Attacks. *Let* $\mathrm{View}_{C,\Pi}(A,B)$ *and* $\mathrm{View}_{S,\Pi}(A,B)$ *denote the protocol view of Client and Server, respectively, after a run of protocol* Π *with private inputs* (A,B) *in which both parties follow the protocol. Then there exist* simulator *algorithms* $\mathcal{S}_C$ *and* $\mathcal{S}_S$ *respectively, with run-time* $O(\mathrm{poly}(s))$*, that, for all* A, B*, can simulate the view of Client and Server, respectively, given only their own private input and output, i.e.:*

$$S_C(A, A \cap_t B) =_c \mathrm{View}_{C,\Pi}(A,B) \text{ and } S_S(B) =_c \mathrm{View}_{S,\Pi}(A,B).$$

3 Share-Hiding Error-Correcting Secret Sharing from Interleaved Reed-Solomon Codes

For our protocol we introduce a primitive that we call a *Share-Hiding Error-Correcting Threshold Secret Sharing Scheme* (SHEC-TSS). In this section, we give the relevant coding background and our SHEC-TSS construction from Interleaved Reed-Solomon codes. We start by formulating abstractly the properties we require from a SHEC-TSS scheme.

Our first requirement is *random error correction*: the secret can be recovered with high probability from a 'noisy' n share vector, in which a subset I of $|I| \geq t$ shares are correct (the rest being uniformly random), even without knowing the positions of the correct shares. It can be viewed as a strengthening of the usual *correctness* requirement on a t-of-n threshold scheme, i.e. that any t shares can be used to recover the secret. The formal definition follows.

Definition 5. *Let* $\delta > 0$*. A* t*-of-*n *secret sharing scheme* SS *with share space* S *is called* δ-random error correcting *if it has the following property. Let* $C = (c_1, \ldots, c_n) \in S^n$ *be the share vector for some secret* s *using scheme* SS*, and let* $I \subseteq [n]$ *be a subset of size* $|I| \geq t$*. Let* $D_{C,I}$ *denote the the probability distribution of 'noisy share vectors'* $\bar{C} = (\bar{c}_1, \ldots, \bar{c}_n)$ *generated as follows: For* $i \in I$*, we set* $\bar{c}_i = c_i$ *(i.e.* $\bar{C}$ *agrees with* C *on shares with indices in* I*), and for* $i \in [n] - I$*, we choose* $\bar{c}_i$ *independently and uniformly at random from share space* S*. Then there exists an efficient (probabilistic poly-time) decoding algorithm* D *that, given* n, t *and* $\bar{C}$ *sampled from distribution* $D_{C,I}$*, returns* C *with probability at least* $1-\delta$ *over the random choice of* $\bar{c}_i$ *for* $i \in [n]-I$ *and the random coins of* D *(note that* D *must succeed with probability* $\geq 1-\delta$ *for every valid share vector* C *and every* $I \subseteq [n]$ *with* $|I| \geq t$*).*

Our second requirement is *share hiding*: for any fixed secret, any collection I of $|I| < t$ shares is a uniformly random $(t-1)$-tuple of elements from the share space. It can be viewed as a strengthening of the usual *security* requirement for a t-of-n threshold scheme, i.e. that any subset of $< t$ shares gives no information on the secret. The formal definition follows.

Definition 6. *A* t*-of-*n *secret sharing scheme* SS *with share space* S *is called* share hiding *if it has the following property. Fix a secret* s *and let* $C = (c_1, \ldots, c_n) \in S^n$ *be the share vector for* s *generated with randomness* ω*. Then, for each* s *and subset* $I \subseteq [n]$ *of size* $|I| < t$*, the* $|I|$*-tuple of shares* $(s_i)_{i \in I}$ *is uniformly random in* $S^{|I|}$ *(over the choice of randomness* ω*).*

Remark 1. The name 'share hiding' comes from the following useful implication that is used in our protocol: let $I \subseteq [n]$ be a share subset and let $D_{C,I}$ denote the distribution of noisy n-share vectors generated as in Def. 5, in which the $|I|$ shares indexed by I are correct, and the rest chosen uniformly at random. Then the share hiding property implies that when $|I| < t$, $D_{C,I}$ is the uniform distribution of n-tuples on the share space, independent of the subset I – hence the correct share subset I is 'hidden'.

Remark 2. The share hiding requirement implies the standard perfect security for t-of-n secret sharing, but the converse is not true in general (see next remark).

Remark 3. It is easy to satisfy the random error correcting property while violating the share hiding property, e.g. share s with a standard t-of-n secret sharing scheme and define the ith share for the new scheme to be the ith share s_i for the old scheme concatanated with some redundancy information on s_i.

Finally, our third technical requirement is *sparsity*.

Definition 7. *Let $\delta > 0$. A t-of-n secret sharing scheme SS with share space S is called* δ-sparse *if a uniformly random n-tuple from S^n has probability at most δ to agree with a valid share vector of some secret s on $\geq t$ positions.*

We can now formally define the notion of a SHEC-TSS.

Definition 8. *A t-of-n secret sharing scheme SS is called* Share-Hiding Error-Correcting *(SHEC-TSS) with error δ, if it is δ-random error correcting, share hiding, and δ-sparse.*

Reed and Solomon [10] discovered the Reed-Solomon code, an important class of error-correcting code. The key idea behind a Reed-Solomon code is that the original data are encoded as a polynomial. The polynomial is then encoded by its evaluation at various points, and these values are what is actually sent. During transmission, some of these values may become corrupted. The Reed-Solomon decoding algorithm can reconstruct the original polynomial as long as sufficient values are received correctly, and hence decode the original data.

Definition 9. *Let p be a prime number and let $t \leq n \leq p$ and let $z = (z_1, \ldots, z_n) \in \mathbb{Z}_p^n$ be a vector of n distinct elements in $\mathbb{Z}_p$. The* Reed-Solomon code $RS_{n,t,p,z}$ *over the field $\mathbb{Z}_p$ with t message symbols and n code symbols is defined as follows. Given a message vector $m = [m_0, m_2, \ldots, m_{t-1}] \in \mathbb{Z}_p^t$, let $P(x)$ be the polynomial*

$$P(x) = m_{t-1}x^{t-1} + \ldots + m_1 x + m_0.$$

Then the codeword $C(m) \in \mathbb{Z}_p^n$ for this message vector is the list of the first n values of the polynomial $P(x)$:

$$C(m) = [P(z_1), P(z_2), \ldots, P(z_n)].$$

Since any two distinct polynomials of degree $t-1$ agree on at most $t-1$ points, the minimum Hamming distance between any two distinct codewords in the code $RS_{n,t,p,z}$ is $n-t+1$. This allows deterministic unique error-correction of a noisy codeword if at most $(n-t+1)/2$ coordinates are incorrect, i.e. at least $t' = t+(n-t-1)/2$ coordinates

are correct. However, in our application, we wish to be able to recover the codeword when t' is as close to t as possible, where t defines a security threshold (so that $t-1$ correct coordinates give no information on the codeword), and the incorrect coordinates are uniformly random and independent in $\mathbb{Z}_p$. The celebrated Reed-Solomon list-decoding algorithm of Guruswami-Sudan [5] gives a unique solution with high probability in our setting, when the number of correct coordinates $t' \geq \sqrt{tn}$, but this is still not sufficiently close to t for our application. To reduce t' closer to t, we use the *Interleaved Reed-Solomon Code* defined as follows.

Definition 10. *Let p be a prime number, $r \geq 1$ an integer, $t \leq n \leq p$, and let $z = (z_1, \ldots, z_n) \in \mathbb{Z}_p^n$ be a vector of n distinct elements in $\mathbb{Z}_p$. The* Interleaved Reed-Solomon code $IRS_{n,t,p,r,z}$ *over the field $\mathbb{Z}_p$ with $r \cdot t$ message symbols and $r \cdot n$ code symbols is defined as follows. Given a message vector $m = (m_1, \ldots, m_r)$ with $m_\ell = [m_{\ell,0}, \ldots, m_{\ell,t-1}] \in \mathbb{Z}_p^t$ for $\ell \in [r]$, let $P_\ell(x)$ be the polynomial*

$$P_\ell(x) = m_{\ell,t-1}x^{t-1} + \ldots + m_{\ell,0}.$$

Then the codeword $C(m) \in (\mathbb{Z}_p^r)^n$ for this message is the vector

$$C(m) = [(P_1(z_1), \ldots, P_r(z_1)), \ldots, (P_1(z_n), \ldots, P_r(z_n))].$$

For $i \in [n]$, we refer to $(P_1(z_i), \ldots, P_r(z_i)) \in \mathbb{Z}_p^r$ as the ith coordinate of the codeword $C(m)$.

Bleichenbacher, Kiayias and Yung [2] showed the following.

Theorem 1 ([2]). *Fix integer parameters $t \leq n \leq p$ with p prime and $r \geq n - t + 1$, and a vector $z = (z_1, \ldots, z_n) \in \mathbb{Z}_p^n$ with $z_i \neq z_j$ for $i \neq j$. There exists an efficient (run-time $O(\mathrm{poly}(n, \log p))$) decoding algorithm D for code $IRS_{n,t,p,r,z}$ that, given n, t, p, r, z and a noisy codeword $Y = C + E \in (\mathbb{Z}_p^r)^n$ with $C \in IRS_{n,t,p,r,z}$ and $E \in (\mathbb{Z}_p^r)^n$ a noise vector with some subset I of $t' \geq t + 1$ coordinates fixed to 0^r and the remaining $n - t'$ coordinates chosen independently and uniformly at random in $\mathbb{Z}_p^r$, returns C with probability at least $1 - (n - t')/q$ over the choice of E and the random coins of D.*

The above algorithm works when the number of correct coordinates in Y is $t' \geq t + 1$, but not for $t' = t$: in that case it is easy to see that C cannot be uniquely decoded from Y if we allow C to be an arbitrary codeword in code $IRS_{n,t,p,r,z}$. To deal with this problem, our secret sharing scheme introduces additional redundancy by restricting C to a subset of codewords whose r polynomials all share the *same* constant coefficient (the secret), i.e. we are using a modified interleaved Reed Solomon code $IRS'_{n,t,p,r,z}$ in which the codewords satisfy $P_\ell(0) = P_1(0)$ for all $\ell \in [r]$. We also make sure that $z_i \neq 0$ for $i \in [r]$. Below, we show that a natural adaptation of the decoding algorithm from Theorem 1 to the code $IRS'_{n,t,p,r,z}$ provides a unique solution with high probability even for $t' = t$, as required.

We formalize our construction of a random-error correcting secret sharing scheme as follows.

Definition 11. *Let p be a prime number, $r \geq 1$ an integer, $t < n < p$, and let $z = (z_1, \ldots, z_n) \in \mathbb{Z}_p$ be a vector of n distinct non-zero elements in $\mathbb{Z}_p$. The t-of-n threshold secret sharing scheme $IRS'_{n,t,p,z,r}$ over the field $\mathbb{Z}_p$ with threshold t and n shares is defined as follows. Given a secret $s \in \mathbb{Z}_p$, the dealer chooses r random polynomials $P_\ell(x)$ of degree $\leq t-1$ with $P_\ell(0) = s$ for $\ell \in [r]$. The share vector $C(s)$ for secret s is*

$$C(s) = [(P_1(z_1), \ldots, P_r(z_1)), \ldots, (P_1(z_n), \ldots, P_r(z_n))],$$

where for $i \in [n]$, the ith share is $(P_1(z_i), \ldots, P_r(z_i)) \in \mathbb{Z}_p^r$.

We now present our main result.

Theorem 2. *Let $IRS'_{n,t,p,z,r}$ be the t-of-n secret sharing scheme defined above. If $r \geq n - t + 1$, the scheme is a SHEC-TSS with error $\delta \leq n/p$.*

Proof. The share hiding property follows from the fact that the collection of $t' < t$ valid shares $c_{i,\ell}$ plus the secret s imposes $t' + 1 \leq t$ constraints of the form $P_\ell(0) = s$ and $P_\ell(z_i) = c_{i,\ell}$ for t' distinct non-zero z_i, on the degree $\leq t-1$ polynomials P_ℓ. Since there is a unique solution for P_ℓ passing through any t given points, there are exactly $p^{t-t'-1} \geq 1$ possible choices for each polynomial P_ℓ satisfying the given constraints, regardless of the values of the $c_{i,\ell}$; the result follows by the uniformly random choice of the P_ℓ.

We now establish the δ-sparse property. Fix a subset $I \subseteq [n]$ with $|I| = t$. The probability that a uniformly random vector $Y \in (Z_p^r)^n$ matches any valid share vector C of $IRS'_{n,t,p,r,z}$ on the shares with indices in I is $1/p^{r-1}$. This is because there is a unique polynomial P_1 of degree $\leq t-1$ satisfying $P_1(z_i) = y_{i,1}$ for $i \in I$, and unique polynomials $P_2, \ldots, P_r$ of degree $\leq t-1$ satisfying $P_\ell(0) = P_1(0)$ and $P_\ell(z_i) = c_{i,\ell}$ for $i \in I - \{i^*\}$, where i^* is one element of I. Hence, the random vector Y will match the valid codeword C also on the i^*th share if and only if $P_\ell(z_{i^*}) = c_{i^*,\ell}$ for $\ell \geq 2$, which holds with probability $1/p^{r-1}$. By taking a union bound over all subsets I of size t we conclude that δ-sparsity holds with $\delta \leq \binom{n}{t}/p^{r-1} = \binom{n}{n-t}/p^{r-1} \leq (n/p)^{n-t} \leq n/p$, using $r \geq n - t + 1$ and $\binom{n}{n-t} \leq n^{n-t}$.

Now we prove the random error-correcting property by explaining the appropriate modifications to the algorithm of Theorem 1 and its analysis in [2]. The decoding algorithm accepts as input n, t, p, z, r and a noisy share vector $Y = (y_1, \ldots, y_n)$ with $y_i = (y_{i,1}, \ldots, y_{i,r}) \in \mathbb{Z}_p^r$ for $i \in [n]$ sampled from the distribution $D_{C,I}$ as in Definition 5, where $C = (c_1, \ldots, c_n)$ is the share vector for some secret s using scheme $IRS'_{n,t,p,z,r}$, and I is a subset of $[n]$ with $|I| \geq t$. The decoding algorithm does not know $|I|$, and therefore tries to decode using a guess t' for $|I|$, until it succeeds. The algorithm works as follows.

- Repeat the following for $t' = t, t+1, \ldots, n$:
  - Randomize: Select r random polynomials $Q_1(x), \ldots, Q_r(x)$ of degree $\leq t-1$ with $Q_\ell(0) = Q_1(0) = s'$ for $\ell \in [r]$, and set $y_{i,\ell} := y_{i,\ell} + Q_\ell(z_i)$ for $i \in [n]$ and $\ell \in [r]$.

- Solve: Find a polynomial $E(x)$ of degree $\leq n - t'$ with constant term 1, and r polynomials $m_\ell(x)$ for $\ell \in [r]$ of degree $\leq n - t' + t - 1$ such that the following linear system of equations is satisfied:

$$m_\ell(z_i) = y_{i,\ell} E(z_i), m_\ell(0) = m_1(0), E(0) = 1, \text{ for } i \in [n], \ell \in [r]. \quad (1)$$

- If the solution $m_1, \ldots, m_r, E$ to (1) is unique and $m_\ell(x)$ is divisible by $E(x)$ for $\ell \in [r]$, then compute polynomials $P_\ell(x) = m_\ell(x)/E(x) - Q_\ell(x)$ for $\ell \in [r]$, and return share vector C determined by those polynomials (with $P_1(0) = m_1(0) - s'$ being the corresponding secret), and terminate.

– If no solution is found for any $t' \in \{t, \ldots, n\}$, terminate and return *failure*.

We note that the algorithm in [2] works in essentially the same way, except that it does not impose the constraints $m_\ell(0) = m_1(0)$ for $\ell \in [r]$. The run-time of each iteration of the algorithm is dominated by the Gaussian elimination procedure for solving the linear system of equations over $\mathbb{Z}_p$ of dimension $\leq r \cdot n$, which can be done in time $O((rn)^3 \log^2 p)$. Thus the overall run-time is $O(n \cdot (rn)^3 \log^2 p)$, which is polynomial in the input length, as required.

The randomization step randomizes the r solution polynomials $P_1, \ldots, P_r$ corresponding to the vector Y, i.e. after this step, we know that there exists a subset $I \subseteq [n]$ of size $|I| \geq t$ and r random polynomials $P'_1, \ldots, P'_r$ of degree $\leq t - 1$ such that $y_{i,\ell} = P'_\ell(z_i)$ for $i \in I$ and $\ell \in [r]$, and $y_{i,\ell}$ uniformly random for $i \in [n] - I$ and $\ell \in [r]$, and $P'_\ell(0) = P'_1(0) = s + s'$ for $\ell \in [r]$. When $t' \leq |I|$, the polynomials P'_ℓ give rise to the following desired solution $m^*_1, \ldots, m^*_r, E^*$ to system (1):

$$E^*(x) = (-1)^{n-|I|} \prod_{i \in [n]-I} (x/z_i - 1), m^*_\ell(x) = P'_\ell(x) \cdot E^*(x), \ell \in [r].$$

However, note that when $t' < |I|$, the system (1) will not have a unique solution, so the algorithm will increment t' until it reaches $t' = |I|$ (indeed, when $t' < |I|$, one can take any subset I' of I of size $|I'| = t'$ and construct a distinct solution to (1) associated with I' by replacing I with I' in the above definition of E^*).

Our goal is to show that when $t' = |I|$, the above desired solution is indeed the unique one. We note that (1) is a linear system with $r \cdot n$ equations and $r \cdot (n - t' + t) + (n - t') - (r - 1)$ variables. A necessary condition for the system to have a unique solution is that it is not under determined; that is, the number of equations is at least equal to the number of variables. It is easy to see that if $r \geq n - t' + 1$, the system (1) is not under determined when $t' \geq t$ (whereas the system in [2] has $r - 1$ additional variables, and is not under determined only for $t' \geq t + 1$).

We now explain how to modify the argument in [2] to show that when the system (1) is not under determined, it has a unique solution with probability at least $1 - (n - t)/p$ over the random choice of the P_ℓ for $\ell \in [r]$ and $y_{i,\ell}$ for $i \in [n] - I$ and $\ell \in [r]$ - we call those the *random variables*.

The argument works in three steps as follows. Starting from the matrix A of the system (1), the first step removes some rows from A to obtain a square matrix $\hat{A}$. The second step is a rearrangement of the rows of $\hat{A}$ to give a matrix $\hat{A}^*$. The final step is to show that the determinant of $\hat{A}^*$ is a *non-zero* polynomial of degree $\leq n - t$ in

the random variables, by showing that the determinant is non-zero for *some* choice of values for the random variables. It then follows by the Schwartz Lemma [11] that the determinant of $\hat{A}^*$ is non-zero (and hence the original system has a unique solution) with probability at least $1-(n-t)/p$ over the uniform choice of the random variables.

First Step. The matrix A for our system (1) has the following form (where we arrange the variables with the $n-t'+t$ coefficients of m_1 first, followed by the $n-t'+t-1$ non-constant coefficients of m_ℓ for $\ell = 2, \ldots, r$, and finally the non-constant coefficients of E):

$$A = \begin{bmatrix} M & 0 & 0 & \cdots & 0 & -M_1 \\ K & \bar{M} & 0 & \cdots & 0 & -M_2 \\ K & 0 & \bar{M} & \cdots & 0 & -M_3 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ K & \cdots & \cdots & \cdots & \bar{M} & -M_r \end{bmatrix},$$

where

$$M = \begin{bmatrix} 1 & z_1 & z_1^2 & \cdots & z_1^{n-t'+t-1} \\ 1 & z_2 & z_2^2 & \cdots & z_2^{n-t'+t-1} \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ 1 & z_n & z_n^2 & \cdots & z_n^{n-t'+t-1} \end{bmatrix},$$

$\bar{M}$ is the $n \times n-t'+t-1$ submatrix of M with the first of column of M (all ones) removed, K is a $n \times (n-t'+t)$ matrix whose elements are all zero except for the leftmost column whose entries are all 1, and for $\ell \in [r]$, M_ℓ is a $n \times (n-t')$ matrix whose (i,j)th element $M_\ell[i,j]$ is related to the (i,j)th element of $\bar{M}$ as follows:

$$M_\ell[i,j] = y_{i,\ell} \cdot \bar{M}[i,j], i \in [n], j \in [n-t']. \tag{2}$$

Since the number of rows of A exceeds the number of columns by $N = r \cdot (t'-t+1)-(n-t'+1)$, we need to remove N rows from A to make it square. The rows of A are naturally divided into r blocks of n rows each, indexed from 1 to r from top to bottom. Similarly to [2], we remove from A the bottom $t'-t+1 \geq 1$ rows of the last $c < r$ blocks of A (note that this makes the c diagonal block matrices in the corresponding blocks, square matrices of dimension $n-t'+t-1$), where $c = \lfloor N/(t'-t+1) \rfloor$. This leaves $N \bmod (t'-t+1)$ remaining rows to remove – they are removed from the bottom of block $r-c \geq 1$. This gives the square matrix $\hat{A}$.

Second Step. In this step, we make the diagonal block matrices of the top $r-c$ blocks square (and hence all block matrices along the diagonal square, thanks to Step 1) by swapping some rows from those blocks to the bottom of the matrix. As in [2], we assume, without loss of generality, that $I = \{n-t'+1, \ldots, n\}$, and we define the *surplus* s_ℓ of block $\ell \in [r-c]$ of $\hat{A}$ as the number of rows that should be swapped from the ℓth block to the bottom of the matrix, in order to make the the corresponding diagonal block matrix square, i.e. $s_1 = t'-t-x_1$ and $s_\ell = t'-t+1-x_\ell$ for $\ell \geq 2$, where x_ℓ is the number of rows removed from block ℓ in Step 1. We observe that, since matrix $\hat{A}$ is square and the number of columns of the M_ℓ matrices on the right is $n-t'$, we have $\sum_{\ell \in [r-c]} s_\ell = n-t'$. We swap rows $1, \ldots, s_1$ of block 1 to the bottom, then rows $s_1+1, \ldots, s_1+s_2$ of block 2 to the bottom, and so on until rows

$\sum_{\ell<r-c} s_\ell + 1, \ldots, \sum_{\ell \le r-c} s_\ell = n - t$ of block $r - c$. The resulting matrix $\hat{A}^*$ has the form:

$$\hat{A}^* = \begin{bmatrix} N_1 & 0 & 0 & \cdots & 0 & -M'_1 \\ K & \bar{N}_2 & 0 & \cdots & 0 & -M'_2 \\ K & 0 & \bar{N}_3 & \cdots & 0 & -M'_3 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ K & \cdots & \cdots & \cdots & \bar{N}_r & -M'_r \\ V_1 & V_2 & \cdots & \cdots & V_r & -\hat{M} \end{bmatrix},$$

where N_1 is a Vandermonde matrix relative to a subset of the z_i's, and for $\ell \ge 2$, N'_r is a scaled Vandermonde matrix relative to a subset of the z_i's (we recall that a Vandermonde matrix of dimension k relative to $(z_1, \ldots, z_k)$ has $(1, z_i, z_i^2, \ldots, z_i^{k-1})$ as its ith row for $i = 1, \ldots, k$. If the ith row is of the form $(z_i, z_i^2, \ldots, z_i^k)$ for $i = 1, \ldots, k$, we call the matrix a *scaled* Vandermonde matrix). Similarly, for $\ell \ge 1$, M'_ℓ is M_ℓ with some rows removed. The matrices $V_1, \ldots, V_r$ and $\hat{M}$ consist of the rows of M, $\bar{M}$, and $M_1, \ldots, M_r$ swapped to the bottom in this step.

Step 3. We show that $\det(\hat{A}^*)$ is non-zero polynomial D in the random variables of degree $n - t'$. The degree follows from the fact that only the last $n - t'$ columns of $\hat{A}^*$ depend on the random variables, and each element in those columns is linear in the random variables. To show that D is a non-zero polynomial, we show that it evaluates to a non-zero value for certain values of the random variables. Namely, we set the polynomials $P_\ell = 1$ for $\ell \in [r]$ (note that this satisfies the constraint that all P_ℓ have the same constant coefficient). This implies $y_{i,\ell} = 1$ for all $i \in \{n-t'+1, \ldots, n\}$ (since $I = \{n - t' + t, \ldots, b\}$) and $\ell \in [r]$. We also set $y_{i,\ell} = 1$ for all rows i, ℓ which have not been moved to the bottom of the matrix in Step 2. On the other hand, we set $y_{i,\ell} = 0$ for all rows i, ℓ which have been moved to the bottom in Step 2 (note that these rows have $i \le n - t'$ by construction, therefore the corresponding random variables $y_{i,\ell}$ can take on arbitrary values, independent of the P_ℓ's). For this setting of the random variables, we have that M'_1 is equal to the submatrix of N_1 consisting of columns 2 to $n - t + 1$, and for $\ell \ge 2$, M'_ℓ is equal to the submatrix of N_ℓ consisting of the first $n - t$ columns. We also have that $\hat{M}$ is the zero matrix.

We now perform elementary row operations on $\hat{A}^*$ (with the above setting of the random variables) to zero out all elements below the square block matrices $N_1, \ldots, N_r$ along the diagonal. Since N_ℓ for $\ell \in [r]$ is a Vandermonde (or scaled Vandermonde) matrix, and the z_i's are distinct and non-zero, then N_ℓ is full rank and has non-zero determinant (it is well known that a Vandermonde matrix relative to $z_1, \ldots, z_k$ has a non-zero determinant when the z_i are distinct; the scaled Vandermonde matrix can be obtained by multiplying each row of a Vandermonde matrix by the corresponding z_i, so the scaled Vandermonde matrix has a non-zero determinant when the z_i are all *non-zero* and distinct). First, we eliminate all 1 elements in the first column of $\hat{A}^*$. Since N_1 is full rank, we can express each row $(1, 0, \ldots, 0)$ of K as a linear combination of the rows of N_1. Subtracting this linear combination of the first $n - t$ rows of $\hat{A}^*$ from the rows below, we eliminate the 1 elements in the first column of these rows. Furthermore, since M'_1 consists of the submatrix of N_1 *except the first column*, this operation has no effect on the elements of M'_ℓ for $\ell \ge 2$. Next, we similarly eliminate the elements in V_1 by expressing each row of V_1 as a linear combination of the rows of N_1 and subtracting

this combination of rows of $\hat{A}^*$ from the non-zero first s_1 rows of V_1. Due to the relation of M'_1 and N_1 explained above and the fact that $\hat{M} = 0$, the first s_1 rows of $\hat{M}$ after this operation are the first s_1 rows of V_1 before this operation (without the first column). Similarly, we eliminate the s_ℓ non-zero rows of V_ℓ using N_ℓ for $\ell = 2, \ldots, r$. At the end, we get all zero elements below the diagonal square block matrices $N_1, \ldots, N_r$, and the matrix $\hat{M}$ has the form of a scaled Vandermonde matrix relative to a subset of the z_i's, and therefore has a non-zero determinant. It follows that $\det(\hat{A}^*)$ is the product of the non-zero determinants of the N_ℓ for $\ell \in [r]$ and $\hat{M}$, so $\det(\hat{A}^*)$ is non-zero for the above setting of the random variables, as claimed. □

4 Private Fuzzy Matching Protocol

We show how to combine our our error correcting secret sharing scheme with homomorphic encryption to get a simple protocol secure in the passive case, which has quadratic communication complexity in the size of the datasets. Our protocol is similar to a fuzzy matching protocol of [3]. However, the protocol of [3] requires computation *exponential* in the size of the datasets. In contrast, our protocol makes use of error correction techniques to improve computation complexity to be *polynomial* in the size of the datasets.

Our simple protocol is shown in Fig. 1.

Theorem 3. *The protocol Γ_1 is a* passive-secure *t-fuzzy matching protocol, assuming that the underlying homomorphic cryptosystem $\mathcal{E}$ and one-time symmetric cryptosystem E are semantically secure.*

Proof. We first show completeness. For each $i \in [m]$, $j \in [n]$, let $I_{ij} = \{k \in [T] : a_{ik} = b_{jk}\}$. Note that if $k \in I_{ij}$, i.e. A_i matches B_j on the kth letter, then the share $\bar{C}_{ijk}$ decrypted by Alice matches the corresponding share C_{ijk} for secret B_j created by Bob, whereas if $k \notin I_{ij}$ the share $\bar{C}_{ijk}$ is independent and uniformly random in $(\mathbb{Z}_p)^r$, thanks to the uniform independent choice of γ^ℓ_{ijk} for $\ell \in [r]$. There are two cases to consider.

The first case is that $A_i \approx_t B_j$, so that $|I_{ij}| \geq t$. In this case, $\bar{C}_{ij}$ is sampled from the noisy share vector distribution $D_{C_{ij},I_{ij}}$ defined in Def. 5. Since $|I_{ij}| \geq t$ and $r \geq T - t + 1$, we conclude from the δ-random error correcting property of $IRS'_{T,t,p,r,z}$ that in this case, except with probability $\delta \leq (T-t)/p \leq T/p$, Alice recovers secret key $s = k_j^{sym}$ so that $\bar{B}_j = D(k_j^{sym}, \rho_j) = B_j$ and Alice correctly adds (A_i, B_j) to the output set S.

The second case is that A_i is not t-fuzzy equal to B_j so that $|I_{ij}| < t$. In this case, we claim that Alice correctly concludes that $(A_i, B_j) \notin A \cap_t B$, except with probability at most T/p. This is because, in order for Alice to make a mistake, $\bar{C}_{ij}$ would have to match *some* valid share vector $\hat{C}$ of scheme $IRS'_{T,t,p,r,z}$ on at least t shares. To bound the probability of this bad event B, we first observe that $|I_{ij}| < t$ implies, by the share-hiding property of $IRS'_{T,t,p,r,z}$, that the probability distribution of $\bar{C}_{ij}$ is uniform on $(\mathbb{Z}_p^r)^T$, independent of the secret k_j^{sym}. It follows from the δ-sparse property of $IRS'_{T,t,p,r,z}$ that event B occurs with probability $\leq \delta \leq T/p$. We conclude that, for each i, j, Alice makes a mistake with probability at most T/p, hence the overall

Input: Security parameter k, Alice has a dataset $A = \{A_1, \ldots, A_m\}$ and Bob owns a dataset $B = \{B_1, \ldots, B_n\}$, where $A_i = (a_{i1}, \ldots, a_{iT}) \in \mathbb{Z}_p^T$ and $B_j = (b_{j1}, \ldots, b_{jT}) \in \mathbb{Z}_p^T$,for some prime $p \geq 2^k mnT$.
Output: Alice learns $A \cap_t B$.

0. **Setup.** Alice and Bob agree on T distinct points $z = (z_1, \ldots, z_T) \in (\mathbb{Z}_p \setminus \{0\})^n$ and parameter $r \geq T - t + 2$ for the t-of-T secret sharing scheme $IRS'_{T,t,p,r,z}$ over $\mathbb{Z}_p$, a homomorphic public key cryptosystem $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ with plaintext space $\mathbb{Z}_p$, a one-time symmetric cryptosystem (E, D) with key space K^{sym} and plaintext space $\mathbb{Z}_p^T$. All arithmetic below is defined over $\mathbb{Z}_p$.
1. Bob
   (a) for $j \in [n]$ generates random symmetric key $k_j^{sym} \in K^{sym}$ and computes ciphertext $\rho_j = E(k_j^{sym}, B_j)$.
   (b) sends $(\rho_1, \ldots, \rho_n)$ to Alice (the B_j's are indexed in a random order).
2. Alice
   (a) generates a homomorphic cryptosystem key pair (k_p, k_s) using $\mathcal{K}(1^l)$,
   (b) sends public key k_p and ciphertexts $c_{ik} = \mathcal{E}(a_{ik})$ to Bob for $i \in [m]$ and $k \in [T]$.
3. Bob, for $i \in [m]$, $j \in [n]$:
   (a) computes a random share vector $C_{ij} = (C_{ij1}, \ldots, C_{ijT}) \in (\mathbb{Z}_p^r)^T$ for secret k_j^{sym} using the t-of-T secret sharing scheme $IRS'_{T,t,p,z,r}$, i.e. $C_{ijk} = (C_{ijk}^1, \ldots, C_{ijk}^r)$ with $C_{ijk}^\ell = P_{ij}^\ell(z_k) \in \mathbb{Z}_p$ and $P_{ij}^1(x), \ldots, P_{ij}^r(x)$ are random polynomials over $\mathbb{Z}_p$ of degree $\leq t - 1$ with $P_{ij}^\ell(0) = k_j^{sym}$ for $\ell \in [r]$.
   (b) using c_{ik}, and the homomorphic properties of $\mathcal{E}$, computes ciphertext $\eta_{ijk}^\ell = \mathcal{E}(\bar{C}_{ijk}^\ell)$ for $\bar{C}_{ijk}^\ell = C_{ijk}^\ell + \gamma_{ijk}^\ell \cdot (a_{ik} - b_{jk})$, where γ_{ijk}^ℓ is chosen uniformly and independently from $\mathbb{Z}_p$.
   (c) sends all η_{ijk}^ℓ's to Alice.
4. Alice
   (a) initializes output set S to empty.
   (b) decrypts η_{ijk}^ℓ's to $\bar{C}_{ijk}^\ell = \mathcal{D}(\eta_{ijk}^\ell)$ for $i \in [m]$, $j \in [n]$, $k \in [T]$ and $\ell \in [r]$.
   (c) for $i \in [m]$ and $j \in [n]$,
      i. runs the decoding algorithm from Theorem 2 for secret sharing scheme $IRS'_{T,t,p,r,z}$ on noisy share vector $\bar{C}_{ij} = (\bar{C}_{ij1}, \ldots, \bar{C}_{ijT})$, where $\bar{C}_{ijk} = (\bar{C}_{ijk}^1, \ldots, \bar{C}_{ijk}^r) \in (\mathbb{Z}_p)^r$ for $k \in [T]$. If the decoding algorithm succeeds to recover share vector C_{ij} matching $\bar{C}_{ij}$ on $\geq t$ shares, conclude $A_i \cap_t B_j$ and add $(A_i, \bar{B}_j)$ to output set S, where $\bar{B}_j = D(s, \rho_j)$ and s is the secret corresponding to share vector C_{ij}.
      ii. otherwise, conclude $(A_i, B_j) \notin A \cap_t B$.
   (d) return $S = A \cap_t B$.

Fig. 1. Protocol Γ_1: Computation-Efficient Fuzzy Private Matching Protocol

protocol success probability is at least $1 - mnT/p \geq 1 - 2^{-k}$ using $p \geq 2^k \cdot mnT$. This completes the completeness proof.

The security against passive attacks is shown as follows.

Bob's protocol view consists of just the public key k_p and ciphertexts c_{ik} for Alice's dataset. Accordingly, Bob's view simulator S_B simply generates a key pair (k_p, k_s) for $\mathcal{E}$ to get public key k_p, and simulates ciphertexts $c_{ik} = \mathcal{E}(0)$ (i.e. by encrypting

0 messages). By a standard hybrid argument, the semantic security of the encryption scheme $\mathcal{E}$ implies that this simulation is computationally indistinguishable from the view of Bob in the real protocol (in which $c_{ik} = \mathcal{E}(a_{ik})$).

Alice's protocol view consists of the symmetric key ciphertexts ρ_j and the public key ciphertexts η^ℓ_{ijk}. On input $(A, A \cap_t B)$, Alice's view simulator S_A works as follows. First, S_A generates random keys $k^{sym}_j \in K^{sym}$ for $j \in [n]$. Then, S_A determines from $A \cap_t B$ the number N of distinct words B_j such that $(A_i, B_j) \in A \cap_t B$ for some $i \in [m]$, and chooses a random subset V of N indices $j \in [n]$ to assign to those N words B_j. Now for each $j \in V$, S_A computes ciphertexts $\rho_j = E(k^{sym}_j, B_j)$, and C^ℓ_{ijk} for $\ell \in [r]$ and $k \in [T]$ in exactly the same way as Bob computes them in the real protocol (this is possible since Bob knows the corresponding A_is and B_js). Finally, for each $j \in [n] - V$, S_A computes $\rho_j = E(k^{sym}_j, 0)$ (where $0 \in (\mathbb{Z}^r_p)^T$) and $\eta^\ell_{ijk} = \mathcal{E}(\bar{C}^\ell_{ijk})$ for independent uniformly random $\bar{C}^\ell_{ijk} \in \mathbb{Z}_p$. To analyse this simulation, note that for (i, j) with $j \in V$ the simulation of the ρ_j and C^ℓ_{ijk} is perfect since the simulation exactly follows the protocol. For all other (i, j), we know that A_i is not fuzzy t-equal to B_j, and in this case in the real protocol, as shown in the correctness proof above, the noisy share vector $\bar{C}_{ij}$ is uniformly random in $(\mathbb{Z}^r_p)^T$, independent of the secret k^{sym}_j, perfectly matching the simulation of the η^ℓ_{ijk} for $j \in [n] - V$. Finally, a standard hybrid argument shows that the semantic security of the one-time symmetric encryption scheme E implies that the simulation of the ρ_j (as encryption of zero under a random key) for $j \in [n] - V$ is computationally indistinguishable from the real protocol (encryption of B_j under a random key). This completes the security proof. □

Implementation Remarks. For simplicity, we assumed in the version of the protocol presented above that we use a homomorphic encryption scheme with plaintext space $\mathbb{Z}_p$ for p prime (the Okamoto-Uchiyama [8] cryptosystem is one such example). Our protocol can also directly work with the Paillier cryptosystem [9], in which the plaintext space is $\mathbb{Z}_N$, where $N = pq$ and p, q are distinct primes. The correctness and security analysis of our protocol naturally extend to this case, as long as the z_i,$z_i - z_j$ and $a_{i,\ell} - b_{j,\ell}$ are non-zero mod p and mod q; this can be easily ensured by restricting $z_i, a_{i,\ell}, b_{j,\ell} < \min(p, q)$ (or just relying on the hardness of factoring n). With the same assumptions on the z_i's, the proof of Theorem 2 also extends (by analysing the linear system of equations mod n separately mod p and mod q), except that unique decoding may fail with probability at most $\delta \leq (n-t) \cdot (1/p + 1/q)$. This leads to the correctness condition $N/(p+q) > 2^k mnT$.

In practice, there are actually three separate parameters in our protocol which were assumed to be equal above: the size of the dataset letter space p_d, the encryption scheme plaintext space N, and the secret sharing modulus p. Our protocol correctness only requires $p > 2^k mnT$ for security parameter k, which may typically be much smaller than N for the same security parameter (e.g. for security parameter $k = 80$, $m, n < 2^{20}$ and $T < 2^{10}$ we have $2^k mnT < 2^{130}$ while $N \approx 2^{1024}$). In this case we may improve the efficiency of the protocol by taking $p \approx 2^k mnT$ much smaller than N (assuming that $p_d < p$), which reduces the complexity of error correction. To maintain security of our protocol in this case, Step 3(b) would have to be modified so that η^ℓ_{ijk} are computed as ciphertexts of $\bar{C}^\ell_{ijk} = C^\ell_{ijk} + \gamma^\ell_{ijk} \cdot (a_{ik} - b_{jk}) + w^\ell_{ijk} \cdot p$, where γ^ℓ_{ijk} is chosen uniformly

and independently in $\mathbb{Z}_N$, and w_{ijk}^{ℓ} is chosen uniformly and independently in $\mathbb{Z}_s$ where $s = \lfloor N/p \rfloor$. In step 4(c), the decrypted plaintexts in $\mathbb{Z}_N$ would be reduced modulo p before proceeding with the decoding in $\mathbb{Z}_p$. With this modification, for the case when A_i is not fuzzy t-equal to B_j in the simulation proof of Theorem 3, the noisy share vectors $\bar{C}_{ij}$ decrypted by Alice in Step 4 of the real protocol have coordinates uniformly random in $\mathbb{Z}_N$ for unmatching positions and coordinates uniformly random in $\mathbb{Z}_{k \cdot p}$ for matching positions. The latter are statistically indistinguishable from uniform on $\mathbb{Z}_N$ if N/p is sufficiently large (namely the statistical distance is $\leq mnTp/N$); thus ensuring that $N > 2^k mnTp$, maintains statistical security of the protocol (the simulation consists of choosing the coordinates of $\bar{C}_{ij}$ uniformly and independently at random from $\mathbb{Z}_N$). If the condition $p_d < p$ does not hold, a possible solution is to hash the letters from $\mathbb{Z}_{p_d}$ to $\mathbb{Z}_p$ using a collision-resistant hash function, and then apply the previous protocol.

Efficiency. The communication and computation complexity of our scheme are summarised in Table 1, which also includes the values for previous protocols.

Table 1. Comparison of protocol efficiency with previous protocols, with $m = n$, ℓ_{pk} and ℓ_{sym} are the ciphertext lengths of the homomorphic encryption scheme $\mathcal{E}$ and symmetric encryption E, respectively, k is the security parameter. Also, $T_{\mathcal{E}}, T_{\mathcal{D}}, T_{\mathcal{H}}, T_{\mathcal{A}}$ denote the encryption/decryption time and time for a homomorphic scalar multiplication/homomorphic addition for $\mathcal{E}$, $T'_{\mathcal{E}} = T_{\mathcal{E}} + T_{\mathcal{D}} + T_{\mathcal{H}}$, and T_{sym} denotes the encryption time for E. Only dominant terms (proportional to n^2) are shown.

Scheme	Communication	Computation
Ours	$O(n^2 T^2 \ell_{pk})$	$O(n^2(\mathrm{poly}(T) + T^2 T'_{\mathcal{E}}))$
CH1 [3]	$O(n^2 T \ell_{pk})$	$O(n^2(\binom{T}{t}\mathrm{poly}(T) + T T'_{\mathcal{E}}))$
Yao [13]	$O(n^2 T(\log p + \log T)\ell_{sym})$	$O(n^2 T T_{sym})$
IW [6]	$O(n^2 k T_{\mathcal{D}} \ell_{sym})$	$O(n^2 k(T^2 T_{\mathcal{A}} + T_{\mathcal{D}} T_{sym}))$

Compared to the CH1 protocol [3], our protocol dramatically improves computation by a factor $O(\binom{T}{t}/\mathrm{poly}(T))$ but has larger communication by a linear factor $O(T)$ (due to our use of error correcting secret sharing). We also compare our protocol to two other protocols based on the generic Yao 'garbled circuit' protocol for two-party computation. Since one can choose $\ell_{pk} \approx \log p$, we see that compared to the generic Yao protocol [13], our protocol's communication is roughly a factor $O(T/\ell_{sym})$ times that of the Yao protocol, hence we expect an improvement in the case $T = O(\ell_{sym})$. Although this may not be a huge improvement, we believe it is still a useful, simpler and more natural alternative to the Yao protocol for this application. Note that Yao's protocol is generic and applies to any Boolean ciruit; to apply it to our problem, we represent the fuzzy matching function as a boolean circuit having the n^2 database words as input. Such a circuit can be implemented with $O(n^2 T(\log p + \log T))$ 2-input gates, giving the complexity estimate in Table 1 (in practice, one could use the Fairplay compiler [7] to generate the circuit). The last row in Table 1 corresponds to the fuzzy matching protocol of Indyk and Woodruff [6]. The latter protocol has a dominant communication complexity term (the n^2 term) independent of T, but uses (as a subprotocol) the Yao protocol applied to the decryption circuit $\mathcal{D}$ of a homomorphic encryption scheme, which typically has complexity $T_{\mathcal{D}} = O(\ell_{pk}^3)$, where ℓ_{pk} is the length of the public key. Thus we expect our protocol to be more efficient in the case $T^2 = O(k\ell_{pk}^2)$.

5 Conclusion

We presented a novel share hiding random error-correcting secret sharing scheme based on interleaved Reed-Solomon codes, and showed how to apply it to construct a simple protocol for private fuzzy matching. We believe our secret sharing scheme may find further cryptographic applications in future. The size of shares in our t-of-n scheme is $O((n-t)k)$, where k is the length of the secret. An interesting open problem is to find alternative constructions with smaller shares, as this will improve our protocol's communication efficiency further.

Acknowledgements. The work of Q. Ye, R. Steinfeld and J. Pieprzyk was supported in part by Australian Research Council grant DP0987734. The work of R. Steinfeld was also supported in part by a Macquarie University Research Fellowship (MQRF). The work of H. Wang is supported in part by the Australian Research Council under ARC Discovery Project DP0665035 the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

References

[1] B. Adida and D. Wikstrom. How to shuffle in public. In *4th Theory of Cryptography Conference (TCC '07),*, volume accepted of *LNCS*. Springer-Verlag, 2007.

[2] D. Bleichenbacher, A. Kiayias, and M. Yung. Decoding interleaved reed-solomon codes over noisy channels. *Theoretical Computer Science*, 379:348–360, 2007.

[3] L. Chmielewski and J.-H. Hoepman. Fuzzy private matching (extended abstract). In *The 3rd International Conference on Availability, Security and Reliability*, pages 327–334. IEEE CS Press, 2008.

[4] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - Eurocrypt '04*, volume 3024 of *LNCS*, pages 1–9. Springer-Verlag Berlin Heidelberg, 2004.

[5] V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45:1757–1767, 1999.

[6] P. Indyk and D. Woodruff. Polylogarithmic private approximations and efficient matching. In *TCC 2006*, 2006. See also ECCC, Report No. 117 (2005).

[7] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay – a secure two-party computation system. In *Proceedings of the 13th USENIX Security Symposium*. USENIX Association, 2004.

[8] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology - Eurocrypt '98*, volume 1403 of *LNCS*, pages 308–318. Springer-Verlag, 1998.

[9] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - Eurocrypt '99*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, 1999.

[10] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of Society for Industrial and Applied Mathematics*, 8(2):300 – 304, June 1960.

[11] J. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27:701–717, 1980.

[12] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[13] A. Yao. How to generate and exchange secrets. In *Proceedings of the 27th FOCS*, pages 162–167, 1986.