Secure Approximate String Matching for Privacy-preserving Record Linkage

Aleksander Essex
Department of Electrical and Computer Engineering
Western University, Canada
aessex@uwo.ca.

***Abstract*—Real-world applications of record linkage often require matching to be robust in spite of small variations in string fields. For example, two health-care providers should be able to detect a patient in common, even if one record contains a typo or transcription error. In the privacy-preserving setting, however, the problem of *approximate string matching* has been cast as a trade-off between security and practicality, and the literature has focused mainly on *Bloom filter encodings*, an approach which can leak significant information about the underlying records.**

We present a novel public-key construction for secure two-party evaluation of threshold functions in restricted domains based on embeddings found in the message spaces of additively homomorphic encryption schemes. We use this to construct an efficient two-party protocol for privately computing the threshold Dice coefficient. Relative to the approach of Bloom filter encodings, our proposal offers formal security guarantees and greater matching accuracy. We implement the protocol and demonstrate the feasibility of this approach in linking medium sized patient databases with tens of thousands of records.

***Index Terms*—Homomorphic encryption, secure computation, approximate string matching, privacy-preserving records linkage**

I. Introduction

RECORD linkage is an important activity in health data science which seeks to identify records across datasets belonging to a single entity. Common applications in the health setting include public-health surveillance, efficient allocation of resources, detecting double-enrollments in clinical trials, and creating high-quality medical research datasets. Legislative and policy restrictions such as the U.S. Health Insurance Portability and Accountability Act (HIPAA) restrict data sharing between organizations for privacy purposes. This gives rise to the challenge of privacy-preserving record linkage, i.e., a records-linkage functionality that reveals *only* the identities of patients who are present in both datasets.

This challenge is made considerably harder when linkage must be robust against errors and variations in string fields. In real-world applications, errors can originate from a variety of sources. Common sources include: character edits (a random character is substituted, inserted or deleted); keyboard edits due to human typing mistakes; optical character recognition (OCR) errors due to misclassifications by OCR software; and phonetic errors during speech transcription due to mispronunciation, or ambiguity between spoken names, and their corresponding spelling [1].

In this paper, we consider the problem of privacy-preserving *approximate matching*, i.e., linking records in the privacy-preserving setting in a manner that is robust against such variations.

A. Limitations of Existing Approaches

Solutions to secure approximate matching must account for a potentially staggering number of possible variations, making a naive public-key approach too computationally intensive to be practical in a real-world setting.

One popular method for approximate matching is Bloom filter encodings (BFEs) [2]. The approach involves parties running strings through a type of locality-preserving hash function in which similar strings produce similar hashes. Data holders send these hashes to a third party linker, who searches for pairs with similar hashes. The main advantage of this approach is speed—the matching algorithm can be run on the hash values directly.

This advantage, however, is simultaneously the cause of its two most significant limitations. One is that it can only approximate its ideal functionality; matching the similarity of hashes produces errors relative to matching the similarity of the strings outright. The second, and most important, is that Bloom filter encodings are fundamentally incompatible with formal security notions or guarantees. Localized features of the string always affect the same bits of the hash, making BFEs semantically insecure by their very design, and numerous papers have proposed cryptanalytic techniques to recover information about underlying patient data [3]–[6]. More recent work has sought to disrupt cryptanalytic efforts through increasingly elaborate counter-measures [4], [7], [8] to mitigate these limitations. Any improvements to security, however, can only be heuristically measured and are often accomplished at the expense of matching accuracy.

B. Contribution

In this paper, we seek a *public-key* solution to approximate string matching that provides formal security guarantees, but is practical for real-world linkage applications. We introduce a novel public-key construction for computing threshold functions in restricted domains based on embeddings found in

This paper is a preprint of the published version in IEEE Transactions on Information Forensics and Security (TIFS), 14(10) pp. 2623–2632, 2019. https://ieeexplore.ieee.org/document/8662592

the message spaces of additively homomorphic encryption schemes. We propose an efficient and secure protocol in the semi-honest model for computing the threshold Dice coefficient. We implement this protocol and demonstrate its feasibility by linking two $20,000$ patient record databases in under 2 hours at a cloud computing cost of around $2. The contributions of this paper include:

- A novel approach for homomorphically approximating threshold functions on limited domains in the integer factorization setting,
- An efficient protocol for approximate string matching,
- A simulation-based proof of security of the protocol in the honest-but-curious setting,
- Implementation and performance evaluation demonstrating the feasibility of linking patient name databases in the tens of thousands of records.

II. Background

A. Approximate Matching

A common approach to approximate string matching in the field of records linkage involves decomposing strings into sets of n-grams and applying a set similarity metric between the two sets [2], [4], [7], [9]–[11]. Without loss of generality, in this paper we will focus on *bigram* decompositions (i.e., $n = 2$).

Definition 1 (Bigrams)**.** *Let $s \in \Sigma^*$ be a string defined on an alphabet Σ, and let '_' be a character in which $\_ \notin \Sigma$, which will be used to create special bigrams representing the beginning and end of a string. Let $s(i)$ denote the i-th character of s. Let $\|$ denote string concatenation. The set of bigrams of an ℓ-character string s is defined as:*

$$\begin{aligned}\mathsf{Bigrams}(s) \quad &= \quad \{t(1)t(2), \ldots, t(\ell-1)t(\ell)\} \\ &\quad\; \textit{where } t = \_ \,\|\, s \,\|\, \_.\end{aligned}$$

For example,

$$\mathsf{Bigrams}(\texttt{CRYPTO}) = \{\texttt{\_C}, \texttt{CR}, \texttt{RY}, \texttt{YP}, \texttt{PT}, \texttt{TO}, \texttt{O\_}\}.$$

The set of all bigrams under Definition 1 is:

$$BG = \{\texttt{\_A}, \texttt{\_B}, \ldots, \texttt{\_Z}, \texttt{A\_}, \texttt{B\_}, \ldots, \texttt{Z\_}, \texttt{AA}, \texttt{AB}, \ldots, \texttt{ZZ}\}.$$

With the lexicographic ordering above, we can refer to the *first* bigram, i.e., $BG(1) = \texttt{\_A}$, the *second* bigram, $BG(2) = \texttt{\_B}$, and so on. The set of bigrams has cardinality $|BG| = 2 \cdot 26 + 26^2 = 728$. A bigram decomposition $\mathsf{Bigrams}(s)$ of a string s is a subset of BG, and thus for all $s \in \Sigma^*$, $|\mathsf{Bigrams}(s)| \leq 728$.

B. Set Similarity

A common metric of set similarity used in the approximate matching literature is the Dice coefficient. Given two sets $\mathbf{a}, \mathbf{b}$:

$$\mathsf{Dice}(\mathbf{a}, \mathbf{b}) = \frac{2 \cdot |\mathbf{a} \cap \mathbf{b}|}{|\mathbf{a}| + |\mathbf{b}|}. \tag{1}$$

As an example, let $\mathbf{a} = \mathsf{Bigrams}(\texttt{CRYPTO})$ and $\mathbf{b} = \mathsf{Bigrams}(\texttt{KRYPTO})$, the corresponding Dice coefficient would be

$$\mathsf{Dice}(\mathbf{a}, \mathbf{b}) = \frac{2 \cdot 5}{7 + 7} = 0.71$$

Grzebala and Cheatham [1] note that the Dice coefficient is typically preferable to the related Jaccard index in terms of matching accuracy, and thus we will concentrate primarily on the former for the remainder of this paper, though a protocol for computing the latter would require only a small modification to the protocol presented in this paper. In the secure approximate matching setting we are interested in computing *only* whether the similarity between two strings exceeds a given threshold.

Definition 2 (Threshold Dice coefficient)**.** *Given two sets $\mathbf{a}, \mathbf{b}$, and a match threshold $0 \leq t \leq 1$, we define the threshold Dice coefficient as follows:*

$$\mathsf{ThreshDice}(\mathbf{a}, \mathbf{b}, t) = \begin{cases} 1 & \mathsf{Dice}(\mathbf{a}, \mathbf{b}) \geq t \\ 0 & \textit{otherwise.} \end{cases} \tag{2}$$

C. Secure Approximate Matching

We define records linkage between two parties, P_A and P_B, whose goal is to identify all records in their respective databases referring to the same entity (i.e., same patient). Extensive literature exists on the topic of privacy-preserving records linkage [1], [10]–[13], especially identifying record fields that match *exactly*. In this paper, we focus instead on the *approximate* matching of string fields, and define our ideal functionality in Functionality 1 as follows.

Functionality 1: Approximate string matching for records linkage

- **Public parameters:** Match threshold t, database sizes m, n.
- **Private inputs:** Party P_A holds a list of patient names $A = [a_1, \ldots, a_n]$. Party P_B holds a list of patient names $B = [b_1, \ldots, b_m]$.
- **Functionality:** For each $a_i \in A$ and $b_j \in B$:
  1) Let $\mathbf{a}_i \leftarrow \mathsf{Bigrams}(a_i), \mathbf{b}_i \leftarrow \mathsf{Bigrams}(b_i)$
  2) Let $d_{ij} \leftarrow \mathsf{ThreshDice}(\mathbf{a}_i, \mathbf{b}_i, t)$
- **Output:** For any $d_{ij} = 1$, P_a learns i.

III. Related Work

The literature on record linkage is extensive, as is the field of *privacy-preserving* record linkage. We direct the reader to the survey of private record linkage techniques by Vatsalan et al. [11], and the survey of techniques for approximate string matching by Grzebala and Cheatham [1].

A related problem in the literature is privacy-preserving *pattern matching* in which one party learns the locations in a long text where a given pattern appears. Hazay and Toft [14] propose a two-party pattern matching approach in the malicious model based on the additive homomorphic properties of Elgamal, which was extended by Vergnaud [15] based on the Fast Fourier Transform. Chase and Shen [16] approach

the problem of outsourced substring queries using searchable symmetric-key encryption. Kolesnikov et al. [17] approach the problem of secure wildcard matching with oblivious transfer extensions, while the work of Wei et al. [18] incorporates secret sharing.

Much of the work in the public-key setting toward secure evaluation of the threshold Dice coefficient has focused on computing Functionality 1 under *partial* encryption. Randall et al. [19] propose a fully homomorphic solution for computing the Dice coefficient based on the *ring learning with error* problem. The construction computes inner-products of Bloom filters under encryption, but the division and threshold steps of Equations 1 and 2 are computed in the clear. Vatsalan et al. [8] propose a similar construction based on additively homomorphic encryption for multiple parties where the threshold is also computed in the clear. Cheon et al. [20] proposed a circuit-based approach to the related problem of secure edit distance based on somewhat-homomorphic encryption.

Public-key approaches in general, however, are often regarded as too computationally intensive [4], [10], [11], [21], and much of the literature has chosen to trade security guarantees in favor of efficiency.

A. Bloom Filter Encodings

Bloom filter encodings (BFEs) were proposed by Schnell [2] as a computationally efficient method for approximate matching based on the original data structure of Bloom [22]. A Bloom filter is an ℓ-bit vector. k secret hash functions $h_i : \{0,1\}^* \rightarrow \mathbb{Z}_\ell$ are defined between the two parties. An element $x \in \{0,1\}^*$ is inserted into a filter by setting the bit positions pointed to by the respective outputs of $h_1(x) \ldots h_k(x)$.

Each party computes encodings of its respective names and sends the filters to a third-party linker L to be matched. The linker computes the threshold Dice coefficient between each filter's pair and reports any that exceed the threshold. Bloom filter encodings are not semantically secure *by design*; similar plaintexts result in similar Bloom filters allowing L to perform the matching. Despite being the subject of ongoing cryptanalytic efforts [3]–[6], [23], Bloom filters continue to be applied in approximate matching applications [21], [24]–[27].

Recently Lazrig et al. [28] proposed performing the Dice coefficient calculation of Bloom filter encodings using garbled circuits. However, Bloom filter encodings are noisy relative to the ideal functionality, which can lead to matching errors.

In the following section, we propose a solution that fully realizes Functionality 1 using a novel cryptographic construction that homomorphically computes the threshold in a single public-key operation.

IV. Definitions and Notations

We begin with several security definitions and notations.

Definition 3 (Computational indistinguishability)**.** *Given a security parameter ρ, a function $\mu(\rho)$ is* negligible in ρ *if for every polynomial $p(\cdot)$, there exists a ρ_0 such that $\mu(\rho) < \frac{1}{p(\rho)}$ for all $\rho > \rho_0$. We say two distribution ensembles X and Y are* computationally indistinguishable *if for every polynomial time distinguisher D there exists a negligible $\nu(\cdot)$ such that*

$$\Big|\Pr[D(X) = 1] - \Pr[D(Y) = 1]\Big| < \nu(\rho).$$

We use the notation $X \stackrel{c}{\equiv} Y$ to denote computational indistinguishability of X and Y.

Definition 4 (Semantic security under Chosen plaintext attack)**.** *Let $\mathcal{CS} = \{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}\}$ be a public key encryption scheme. Let $\mu(\rho)$ be a negligible function in ρ, and $\mathcal{A}$ be a non-uniform adversary. We define the chosen plaintext attack experiment $\mathsf{Exp}^{CPA}_{\mathcal{A},\mathcal{CS}}(\rho)$ in the standard way: $\mathcal{PK} \leftarrow \mathsf{Gen}(1^\rho)$ is run. $\mathcal{A}$ can sample the output of $\mathsf{Enc}(m)$ polynomially many times for chosen m. Finally, $\mathcal{A}$ chooses two messages m_0, m_1. $c = \mathsf{Enc}(m_b)$ is computed for $b \leftarrow_\$ \{0,1\}$ and given to $\mathcal{A}$, who outputs a guess g for b. The experiment returns 1 if $g = b$, and 0 otherwise. $\mathcal{CS}$ is semantically secure against chosen plaintext attack if, for every polynomial adversary $\mathcal{A}$:*

$$\Pr\Big[\mathsf{Exp}^{CPA}_{\mathcal{A},\mathcal{CS}}(\rho) = 1\Big] \leq \frac{1}{2} + \mu(\rho).$$

Definition 5 (Semi-honest setting)**.** *In this paper we work the semi-honest (honest-but-curious) model: two parties P_A, P_B honestly follow the protocol but might try to analyze the transcript to infer more information about the other party's inputs. Let $f = (f_A, f_B)$ be a two-party functionality, and let π be a protocol for computing f. Let P_A, P_B have respective inputs x_A, x_B. The view of party P_A of the execution of π on their combined input $x = (x_A, x_B)$ is denoted as*

$$\mathsf{View}_{P_A}(x) = (x_A, r_A, msg_A),$$

where x_A is P_A's input to the protocol, r_A is all random values generated by P_A, and msg_A is the set of all messages received by P_A during the execution of π. The output of P_A in the execution of π is denoted $\mathsf{Output}_{P_A}(x)$ and is computable from $\mathsf{View}_{P_A}(x)$. Similarly, $\mathsf{View}_{P_B}(x) = (x_B, r_B, msg_B)$ and $\mathsf{Output}_{P_B}(x)$ denotes P_B's view and output of protocol π on input x.

Definition 6 (Secure evaluation)**.** *Protocol π securely evaluates f in the presence of a semi-honest adversary if there exists probabilistic polynomial-time algorithms P^*_A and P^*_B such that*

$$\big[P^*_A\big(x_A, f_A(x)\big), f_B(x)\big] \stackrel{c}{\equiv} \big[\mathsf{View}_{P_A}(x), \mathsf{Output}_{P_B}(x)\big]$$

and

$$\big[P^*_B\big(x_B, f_B(x)\big), f_A(x)\big] \stackrel{c}{\equiv} \big[\mathsf{View}_{P_B}(x), \mathsf{Output}_{P_A}(x)\big].$$

Let the notation $[\![x]\!]$ denote the encryption of a plaintext x in a semantically-secure additively homomorphic public-key encryption scheme. Let the notation $r \leftarrow_\$ S$ denote an element r sampled uniformly at random from a set S. Let $\mathbb{Z}, \mathbb{Z}_n$ and $\mathbb{Z}^*_n$ respectively denote the set of integers, the set of integers modulo an integer $n > 0$, and the set of integers relatively prime to n. Throughout this paper we use the notation $\mathbb{Z}_p$ and $\mathbb{Z}^*_p$ in the context of a prime p. Let $q|(p-1)$, and let $\mathbb{G}_q$ denote the cyclic subgroup of $\mathbb{Z}^*_p$ of order q.

x	0	1	2	3	4	5	6	7	8	9	10
$\mathsf{QR}_{277}(x+179)$	0	0	0	0	0	0	1	1	1	1	1
$\tau_6(x)$	0	0	0	0	0	0	1	1	1	1	1

TABLE I
EXAMPLE 10-APPROXIMATION OF $\tau_6(x)$ IN $\mathbb{Z}_{277}$ AT 179.

Definition 7 (Quadratic residuosity)**.** *Recall the Legendre symbol of an integer x modulo prime p:*

$$\left(\frac{x}{p}\right) \equiv x^{\frac{p-1}{2}} \bmod p \tag{3}$$

where $\left(\frac{x}{p}\right) \in \{-1, 0, 1\}$. An integer x is a quadratic residue modulo p if there exists some integer y such that $y^2 \equiv x \bmod p$. Let $\mathsf{QR}_p : \mathbb{Z} \times \mathbb{Z} \to \{0,1\}$ be a function denoting the quadratic residuosity of an integer $x \in \mathbb{Z}$ modulo $p \in \mathbb{Z}$:

$$\mathsf{QR}_p(x) = \begin{cases} 1 & \text{if } x \text{ is a quadratic residue modulo } p. \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Definition 8 (Threshold function)**.** *For a threshold $t \in \mathbb{Z}^+$, let $\tau_t : \mathbb{Z}^+ \to \{0,1\}$ be a threshold function defined as follows:*

$$\tau_t(x) = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

A. Approximating Threshold Functions in $\mathbb{Z}_p$

Consider the Legendre symbols of a sequence of successive elements, $x, (x+1), \ldots, (x+k)$ in $\mathbb{Z}_p$. Except for squares in $\mathbb{Z}$, which always have a Legendre symbol of 1, such a sequence takes on an irregular, mostly random-looking appearance.

The intuition of our approach comes from the observation that, for certain primes, this sequence may contain subsequences of regularity that locally approximate the output of some well-behaved function. Of course, such an approximation, if it were present, would be entirely coincidental. Its presence, nevertheless, can be exploited to effectively evaluate the function over a *restricted* domain. Specifically, we exploit the presence of such "well-behaved" subsequences to approximate threshold functions. The intuition is that for a threshold function τ_t, there may exist a prime s for which a subsequence generated by QR_s locally approximates τ_t.

Definition 9 (d-approximation of τ_t)**.** *Let s be a prime and let $0 < f \leq s$ be a value such that*

$$\mathsf{QR}_s(x+f) = \tau_t(x) \tag{6}$$

for all x in the range $0 \leq x \leq d$. We say the quadratic residuosity function QR_s d-approximates threshold function τ_t at f.

For example, QR_{227} 10-approximates the threshold function τ_6 at 179 (see Table I). Using this approach in conjunction with a semantically secure encryption scheme that is additively homomorphic modulo s, we will show how this can be used to evaluate the threshold Dice coefficient homomorphically.

V. ENCRYPTION SCHEME

We now discuss our public-key method for homomorphically evaluating a d-approximation of τ_t. Let $\mathcal{CS} = \{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}\}$ be a semantically secure, additively homomorphic public-key encryption scheme. Let $\mathcal{M}$ and $\mathcal{C}$ respectively denote the plaintext and ciphertext spaces of $\mathcal{CS}$. For any two messages $m_1, m_2 \in \mathcal{M}$ let,

$$\mathsf{Dec}\big(\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2)\big) = (m_1 + m_2) \bmod s$$

for a prime s. The value s will be chosen such that QR_s d-approximates τ_t. For concreteness and efficiency we will specify $\mathcal{CS}$ as special case of the cryptosystem due to Damgård et al. [29], however other options include the cryptosystems due to Benaloh [30], and Groth [31], [32], or more broadly, any additively homomorphic cryptosystem that can accommodate a message space of order s.

Let $\mathcal{G}$ be an algorithm that accepts a security parameter $\rho \in \mathbb{Z}^+$, threshold $t \in \mathbb{Z}^+$ and domain bound $d \in \mathbb{Z}^+$ and outputs:

1) A pair of integers (ℓ, w), where the factorization of the product of two random ℓ-bit primes is computationally infeasible, and where the discrete logarithm in a group of prime w-bit order is computationally infeasible in the security parameter,
2) A prime s, and offset $0 < f < s$ such that $f + d < s$ and $\mathsf{QR}_s(x+f) = \tau_t(x)$ for $0 \leq x < d$.

See Table V for computationally derived examples of (f, s) for 2μ-approximating a threshold function τ_μ. We now specify key generation, encryption, and decryption functions:

$\mathsf{Gen}(\rho)$**:** Given security parameter $\rho > 0$, run $\mathcal{G}(\rho, t, d)$ to obtain (ℓ, w, s, f) as defined above. Pick two random w-bit primes u, v. Let $n = pq$ for primes p and q such that $su|(p-1)$, $sv|(q-1)$. Let $\mathbb{G}_{suv}$ denote the unique subgroup of $\mathbb{Z}_n^*$ which has order su in $\mathbb{Z}_p^*$ and order sv in $\mathbb{Z}_q^*$. Let $\mathbb{G}_{uv}$ denote the subgroup of $\mathbb{Z}_n^*$ with order uv. Let g be a generator of $\mathbb{G}_{suv}$ and let h be a generator of $\mathbb{G}_{uv}$. Compute $\mu = (uv)^{-1} \bmod s$, and set $\lambda = uv\mu$. The public key is $\mathcal{PK} = (n, g, h, w, d, f, s, t)$. The private key is $\mathcal{SK} = \lambda$.

$\mathsf{Enc}(\mathcal{PK}, m)$**:** For a message m in the range $0 \leq m \leq d$, pick a random element $r \leftarrow\!\!\$\, [1, \ldots, 2^w - 1]$ and compute

$$C = g^m h^r \bmod n.$$

Output ciphertext C.

$\mathsf{Dec}(\mathcal{SK}, C)$**:** To decrypt a ciphertext C using private key $\mathcal{SK}$, compute

$$C^\lambda = C^{uv\mu} = (g^m)^{uv\mu}(h^r)^{uv\mu} = (g^m)^{uv(uv)^{-1}} = g^m.$$

Recover m by computing the discrete logarithm $\log_s(g^m)$. This is efficient for small values of s.[1] Output $\mathsf{QR}_s(m)$.

[1] s would typically be small enough for $\log_s(g^m)$ to be precomputed.

Theorem 1 (Semantic security of $\mathcal{CS}$)**.** *The public-key encryption scheme defined by $\mathcal{CS} = \{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}\}$ is semantically secure under chosen plaintext attack.*

Proof. We begin with the assumption that the public-key scheme due to Dargård et al. [29] (i.e., DGK) is semantically secure, and then show a reduction from their scheme to ours.

Suppose there existed an algorithm $\mathcal{A}'$ that accepted an element $x \in \mathbb{Z}_n^*$ and a public-key $\mathcal{PK} = (n, g, h, w, d, f, s, t)$, and had a non-negligible advantage in guessing whether x was a generator of $\mathbb{G}_{suv}$, i.e., whether x is a valid non-zero ciphertext in $\mathcal{CS}$.

We briefly recall the particulars of the DGK cryptosystem, maintaining its original variable naming conventions for the duration of this proof. Let $\mathcal{PK}_{DGK} = (n, g, h, u, t)$ and $\mathcal{SK} = (p, q, v_p, v_q)$ such that $v_q|(p-1)$, $v_q|(q-1)$, $n = pq$. Let be and g, h generate subgroups of $\mathbb{Z}_n^*$ as follows: g has order $uv_p \bmod p$ and order $uv_q \bmod q$. h has order $v_p \bmod p$ and order $v_q \bmod q$. $\mathsf{Enc}_{DGK}(m) = g^m h^r$ for $m \in \mathbb{Z}_u$ and $r \leftarrow_\$ \mathbb{Z}_{2^t}$. Observe $\mathsf{Enc}_{DGK}(0) = h^r$ is a generator of $\mathbb{G}_{v_p v_u}$ whereas $\mathsf{Enc}_{DGK}(m \neq 0) = g^m h^r$ is not. Thus the semantic security of DGK is broken by an algorithm guessing with advantage whether an element $x \in \mathbb{Z}_n^*$ is a generator $\mathbb{G}_{v_p v_u}$.

Consider an algorithm $\mathcal{A}$ that accepts a DGK public key $\mathcal{PK}_{DGK}$ and an element $x \in \mathbb{Z}_n^*$ which outputs a guess whether x is a generator of $\mathbb{G}_{v_p v_u}$. Suppose $\mathcal{A}(\mathcal{PK}_{DGK}, x)$ calls $\mathcal{A}'(\mathcal{PK}', x)$ where $\mathcal{PK}' = (n, g, h, t, 1, 1, u, 1)$ and outputs the result. $\mathcal{A}'$ guesses if x generates $\mathcal{G}_{v_p, v_q}$ with advantage, which allows $\mathcal{A}$ to guess with advantage. By the assumed semantic security of DGK, $\mathcal{A}$ does not exist implying neither can $\mathcal{A}'$. □

A. Homomorphic Properties

First, we observe that $\mathcal{CS}$ is additively homomorphic:

$$\begin{aligned} \mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2) &= g^{m_1} h^{r_1} \cdot g^{m_2} h^{r_2} \\ &= g^{m_1 + m_2} \cdot h^{r_1 + r_2} \\ &= \mathsf{Enc}(m_1 + m_2 \bmod s) \end{aligned}$$

We now define the functionality for homomorphically d-approximating the threshold function τ_t.

$\mathsf{Eval}(\mathcal{PK}, c)$**:** Given ciphertext $c = \mathsf{Enc}(m)$, sample blinding factor $x \leftarrow_\$ [1, \ldots, s-1]$ and compute

$$c' = \big(c \cdot \mathsf{Enc}(f)\big)^{x^2} \bmod n.$$

Output c'.

Theorem 2 (Homomorphic d-approximation of τ_t)**.** *For $0 \leq m < d$ and ciphertext $c = \mathsf{Enc}(m)$, $\mathsf{Eval}(c)$ homomorphically evaluates $\tau_t(m)$, i.e.,*

$$\mathsf{Dec}\big(\mathsf{Eval}(\mathsf{Enc}(m))\big) = \tau_t(m).$$

Proof. Expanding $\mathsf{Eval}\big(\mathsf{Enc}(m)\big)$ we have:

$$\begin{aligned} \mathsf{Eval}\big(\mathsf{Enc}(m)\big) &= \big(\mathsf{Enc}(m) \cdot \mathsf{Enc}(f)\big)^{x^2} \\ &= \big(g^m h^{r_1} \cdot g^f h^{r_2}\big)^{x^2} \\ &= g^{(m+f)x^2} h^{r'} \\ &= \mathsf{Enc}\big((m+f) \cdot x^2\big) \\ &= c'. \end{aligned}$$

By the law of quadratic reciprocity, the product of two elements $a, b \in \mathbb{Z}_s$ has the following property:

$$\mathsf{QR}_s(ab) = \mathsf{QR}_s(a) \cdot \mathsf{QR}_s(b).$$

Therefore decrypting c' gives:

$$\begin{aligned} \mathsf{Dec}(c') &= \mathsf{QR}_s\big((m+f) \cdot x^2\big) \\ &= \mathsf{QR}_s(m+f) \cdot \mathsf{QR}_s(x^2) \\ &= \mathsf{QR}_s(m+f). \end{aligned}$$

Recall given (t, d), Gen produces $s, f \in \mathcal{PK}$ such that $\mathsf{QR}_s(m+f) = \tau_t(m)$ for $0 \leq x \leq d$. Therefore $\mathsf{Dec}(c') = \mathsf{QR}_s(m+f) = \tau_t(m)$. □

Theorem 3 (Blinding hiding)**.** *Decrypting $\mathsf{Eval}\big(\mathsf{Enc}(m)\big)$ reveals no information about m beyond what is revealed by $\tau_t(m)$.*

Proof. Let $c' = \mathsf{Enc}\big((m+f) \cdot x^2\big)$. Using $\mathcal{SK} = \lambda$, the private-key holder decrypts c' by computing:

$$\begin{aligned} \mathsf{Dec}(c') &= (c')^\lambda \bmod n \\ &= (g^{(m+f) \cdot x^2} h^{r'})^\lambda \bmod n \\ &= g^{(m+f) \cdot x^2} \bmod n. \end{aligned}$$

The private-key holder then computes the discrete logarithm base g to recover $m' = (m+f) \cdot x^2 \bmod s$, and outputs $\mathsf{QR}_s(m')$.

The goal of the proof is to show that the x^2 factor blinds $(m+f)$, and our approach is to show m' is uniform in the set of quadratic residues modulo s if $(m+f)$ is a quadratic residue, and m' is uniform in the set of quadratic non-residues otherwise.

Recall that m, f, d, x were defined such that $f > 0$, $0 \leq m \leq d$, $f + d < s$, and $1 \leq x \leq s$. Therefore $(m+f) \cdot x^2 \not\equiv 0 \bmod s$. Let $s = 2^j \cdot k + 1$ for odd k. There are two subgroups of $\mathbb{Z}_s^*$ of interest: the subgroup $\mathbb{G}_{2^j}$ of order 2^j, and the subgroup $\mathbb{G}_k$ or order k. Let g_j and g_k be generators of $\mathbb{G}_{2^j}$ and $\mathbb{G}_k$ respectively. Any element $a \in \mathbb{Z}_s^*$ can be expressed as

$$a = g_j^y g_k^z \bmod s$$

for some $0 \leq y < 2^j$ and $0 \leq z < k$. Suppose y is even. Then a is a quadratic residue, since there exists a b such that $b^2 = a \bmod s$. By the same argument, if y is odd, a is a quadratic non-residue. With this approach let us rewrite $(m+f)$ and blind factor x as follows:

$$(m+f) = g_j^{y_1} g_k^{z_1} \bmod s$$

$$x = g_j^{y_2} g_k^{z_2} \bmod s.$$

Therefore,

$$\begin{aligned}(m+f)\cdot x^2 &= g_j^{y_1} g_k^{z_1} \cdot (g_j^{y_2} g_k^{z_2})^2 \bmod s \\ &= g_j^{y_1+2y_2} g_k^{z_1+2z_2} \bmod s.\end{aligned}$$

Since x is uniform in $\mathbb{Z}_s^*$, then z_2 is uniform in $\mathbb{Z}_k^*$, and thus z_1+2z_2 is uniform in $\mathbb{Z}_k^*$. Similarly, y_2 is uniform in $\mathbb{Z}_{2^j}^*$, and thus if y_1 is even, then y_1+2y_2 is uniform in the set of non-zero even numbers modulo 2^j, and is uniform in the set of odd numbers modulo 2^j otherwise. Therefore $m' = (m+f)\cdot x^2$ is uniform among the quadratic residues modulo s if $(m+f)$ is quadratic residue, and is uniform among the quadratic non-residues otherwise. □

VI. Protocol for Private Threshold Approximate Matching

In this section, we present our semi-honest secure protocol for computing the approximate string matching for records linkage functionality (Functionality 1).

The protocol consists of two sub-protocols. The first sub-protocol is a protocol to compute the private set intersection cardinality of string bigrams homomorphically. The second sub-protocol is a protocol to homomorphically compute the threshold Dice coefficient ThreshDice using the homomorphic d-approximation properties of $\mathcal{CS}$.

A. Privately Computing Set Intersection Cardinality

The first sub-protocol takes private inputs, $A = [a_1, \ldots a_n,], B = [b_1, \ldots, b_m]$, and for each pair $a_i \in A, b_j \in B$, produces an encryption of the set intersection cardinality of their bigrams, i.e.,

$$r_{ij} = [\![\, |\mathsf{Bigrams}(a_i) \cap \mathsf{Bigrams}(b_j)| \,]\!].$$

Other public-key approaches exist for computing private set intersection cardinalities from unrestricted domains (see e.g. [33], [34]), however, given the restricted domain of this application, we are able to take a more efficient approach by exploiting the additive homomorphic properties of cryptosystem $\mathcal{CS}$. Let EncBigrams be a function that accepts a string and produces an encrypted bigram vector, in which the i-th ciphertext encrypts the presence (or absence) of the i-th bigram in the string. The function is defined as follows:

EncBigrams(String s, Public-key $\mathcal{PK}$)

for $1 \le i \le 728$:
  if $BG(i) \in \mathsf{Bigrams}(s)$:
    $c_i \leftarrow \mathsf{Enc}(1)$
  else:
    $c_i \leftarrow \mathsf{Enc}(0)$
return $C = [c_1, \ldots, c_{728}]$

Let SetIntCard be a function that accepts a string s and an encrypted bigram vector and outputs an encryption of the set intersection cardinality between the bigrams of s and the bigrams in the encrypted bigram vector. The function is defined as follows:

SetIntCard(String s, Encrypted vector C, Public-key $\mathcal{PK}$)

$r \leftarrow 1$
for each $bg \in \mathsf{Bigrams}(s)$:
  $i \leftarrow$ Index of bg in BG
  $r \leftarrow r \cdot c_i \bmod n$, (where c_i is the i-th ciphertext in C).
return r

An example of SetIntCard is shown in Table II. We define the sub-protocol for privately computing set intersection cardinalities in Sub-protocol 1.

Sub-protocol 1: Private Records Set Intersection Cardinality

- **Public parameters:** Public key PK.
- **Private inputs:** Party P_A holds a list of strings $A = [a_1, \ldots, a_n]$. Party P_B holds a list of strings $B = [b_1, \ldots, b_m]$.
- **The protocol:**
  1) For each $a_i \in A$, P_A computes: $C_i \leftarrow \mathsf{EncBigrams}(a_i)$ and sends each C_i to P_B
  2) For each C_i and $b_j \in B$, P_B computes: $r_{ij} \leftarrow \mathsf{SetIntCard}(b_j, C_i)$
- **Output:** P_B outputs all encrypted set intersection cardinalities r_{ij}

Theorem 4 (Correctness (Sub-protocol 1))**.** *For each $a_i \in A$ and $b_j \in B$, Sub-protocol 1 outputs*

$$r_{ij} = [\![\, |\mathsf{Bigrams}(a_i) \cap \mathsf{Bigrams}(b_j)| \,]\!].$$

Proof. For each $a_i \in A$ and $b_j \in B$, r_{ij} steps 1 and 2 compute:

$$r_{ij} = \mathsf{SetIntCard}(b_j, \mathsf{EncBigrams}(a_i)).$$

SetIntCard is run by P_B, who selects encrypted bigram $c_i \in C$ if and only if the i-th bigram is in $\mathsf{Bigrams}(b_j)$. The selected ciphertext is $[\![1]\!]$ if the i-th bigram is also in $\mathsf{Bigrams}(a_i)$, and is $[\![0]\!]$ otherwise. By the additive homomorphic property of CS, the multiplication of all ciphertexts produces an encryption of the sum of the respective plaintexts.

For each bigram $bg \in BG$, SetIntCard selects ciphertexts of the form $[\![1]\!]$ if and only if $bg \in \mathsf{Bigrams}(a_i)$ and $\mathsf{Bigrams}(b_j)$, and selects either $[\![0]\!]$, or no ciphertext otherwise. Therefore, the product of all the ciphertexts will be the encryption of the sum of all bigrams $bg \in BG$ where $bg \in \mathsf{Bigrams}(a_i)$ and $\mathsf{Bigrams}(b_j)$, i.e., the encryption of $|\mathsf{Bigrams}(a_i) \cap \mathsf{Bigrams}(b_j)|$. □

B. Privately Computing the Threshold Dice Coefficient

The next step is to define a sub-protocol for computing the threshold dice coefficient given two privately input set cardinalities $\ell_a = |\mathbf{a}|, \ell_b = |\mathbf{b}|$, and the output from Sub-protocol 1, i.e., $\mathsf{Enc}(|\mathbf{a} \cap \mathbf{b}|)$. We rearrange the threshold Dice equation into an instance of the threshold function τ_t, and use the properties of $\mathcal{CS}$ to homomorphically evaluate τ_t.

TABLE II
EXAMPLE OF SetIntCard COMPUTING THE ENCRYPTED SET INTERSECTION CARDINALITY OF KRYPTO AND EncBigrams(CRYPTO).

_C	...	_J	_K	...	O_	P_	...	KR	KS	...	PT	PU	...	RY	...	TO	...	YP	...
$[\![1]\!]$		$[\![0]\!]$	$[\![0]\!]$		$[\![1]\!]$	$[\![0]\!]$		$[\![0]\!]$	$[\![0]\!]$		$[\![1]\!]$	$[\![0]\!]$		$[\![1]\!]$		$[\![1]\!]$		$[\![1]\!]$	
			↓		↓			↓			↓			↓		↓		↓	
			$[\![0]\!]$		$[\![1]\!]$			$[\![0]\!]$			$[\![1]\!]$			$[\![1]\!]$		$[\![1]\!]$		$[\![1]\!]$	

$$[\![0]\!] \cdot [\![1]\!] \cdot [\![0]\!] \cdot [\![1]\!] \cdot [\![1]\!] \cdot [\![1]\!] \cdot [\![1]\!] \bmod n = [\![5]\!]$$

We construct our d-approximation of τ_t to handle all possible inputs to ThreshDice up to a maximum set intersection cardinality. Let μ be defined as the maximal allowable set cardinality size, i.e., the respective upper bounds of ℓ_a and ℓ_b, and $|\mathbf{a} \cap \mathbf{b}|$. Parameterize $d, t \in \mathcal{PK}$ such that $d = 2 \cdot \mu$ and $t = \mu$.

We can then use $\mathcal{CS}$ to homomorphically evaluate whether an encrypted plaintext is less than, or equal to μ for a plaintext in the range $1 \ldots 2\mu$. Our approach is to pre-compute a value θ such that $\theta + |\mathbf{a} \cap \mathbf{b}| \geq \mu$ if and only if ThreshDice would have yielded a match given the two strings as input. Let

$$\mathsf{DiceOffset}(\ell_a, \ell_b, t, \mu) = \mu - \left\lceil \frac{t}{2}(\ell_a + \ell_b) \right\rceil + 1.$$

We now define the second sub-protocol (Sub-protocol 2), which accepts P_A's and P_B's strings as private inputs, and a ciphertext of their set intersection cardinality, and outputs an encryption of ThreshDice evaluated on the input strings.

Sub-protocol 2: Private Threshold Dice Coefficient

- **Public parameters:** Public key PK. Maximum set cardinality μ. Encrypted set-intersection cardinality $c = [\![\mathsf{Bigrams}(a) \cap \mathsf{Bigrams}(b)]\!]$
- **Private inputs:** P_A's string a, P_B's string b
- **The protocol:**
  1) P_A computes: $\ell_a = |\mathsf{Bigrams}(a)|$
  2) For $2 \leq i \leq \mu$, P_A computes:
     $$[\![\theta_i]\!] = \mathsf{Enc}\big(\mathsf{DiceOffset}(\ell_a, i, t, \mu)\big)$$
  3) P_A sends $[\![\theta_2]\!] \ldots [\![\theta_\mu]\!]$ to P_B in order.
  4) P_B computes: $\ell_b = |\mathsf{Bigrams}(b)|$
  5) P_B selects $[\![\theta_{\ell_b}]\!]$ from the list
  6) P_B computes $d \leftarrow \mathsf{Eval}\big([\![\theta_{\ell_b}]\!] \cdot c\big)$
- **Output:** P_B outputs d.

Theorem 5 (Correctness (Sub-protocol 2))**.** *Given strings a, b as private inputs, and an encryption of their set-intersection cardinality c, Sub-protocol 2 outputs an encryption of the threshold Dice coefficient, i.e.,*

$$[\![\mathsf{ThreshDice}(\mathsf{Bigrams}(a), \mathsf{Bigrams}(b), t)]\!].$$

Proof. In Step 2 of Sub-protocol 2, P_A computes $\mathsf{DiceOffset}(\ell_a, \ell_b, t, \mu)$ for each possible set cardinality of $\ell_b = |\mathsf{Bigrams}(b)|$, and in Step 3, P_B selects the corresponding $[\![\theta_{\ell_b}]\!]$. Therefore,

$$\begin{aligned} \theta_{\ell_b} &= \mathsf{DiceOffset}(\ell_a, \ell_b, t, \mu) \\ &= \mu - \left\lceil \frac{t}{2}(\ell_a + \ell_b) \right\rceil + 1. \end{aligned}$$

By Theorem 2, $\mathsf{Dec}(\mathsf{Eval}(\mathsf{Enc}(m))) = \tau_\mu(m)$. Let $c = [\![\mathsf{Bigrams}(a) \cap \mathsf{Bigrams}(b)]\!] = [\![\ell_{a \cap b}]\!]$. Next recall that $d, t \in \mathcal{PK}$ were selected such that $d = 2 \cdot \mu$ and $t = \mu$. Therefore

$$\begin{aligned} \mathsf{Dec}\big(\mathsf{Eval}([\![\theta_{\ell_b}]\!] \cdot c)\big) &= \mathsf{Dec}\big(\mathsf{Eval}([\![\theta_{\ell_b}]\!] \cdot [\![\ell_{a \cap b}]\!])\big) \\ &= \mathsf{Dec}\big(\mathsf{Eval}([\![\theta_{\ell_b} + \ell_{a \cap b}]\!])\big) \\ &= \tau_\mu(\theta_{\ell_b} + \ell_{a \cap b}) \\ &= \tau_\mu\big(\mu - \left\lceil \frac{t}{2}(\ell_a + \ell_b) \right\rceil + 1 + \ell_{a \cap b}\big). \end{aligned}$$

By the definition of τ in Equation 5, $\tau_\mu(\theta_{\ell_b} + \ell_{a \cap b}) = 1$ if $\theta_{\ell_b} + \ell_{a \cap b} \geq \mu$ (and 0 otherwise). This condition, in turn, is satisfied if and only if $\ell_{a \cap b} - \left\lceil \frac{t}{2}(\ell_a + \ell_b) \right\rceil \geq 0$, i.e., $\ell_{a \cap b} \geq \left\lceil \frac{t}{2}(\ell_a + \ell_b) \right\rceil - 1$. Similarly, ThreshDice() outputs 1 if $\frac{2\ell_{a \cap b}}{\ell_a + \ell_b} \geq t$ (and 0 otherwise), i.e., $\ell_{a \cap b} \geq \frac{t}{2}(\ell_a + \ell_b)$. Therefore,

$$\mathsf{Dec}\big(\mathsf{Eval}([\![\theta_{\ell_b}]\!] \cdot c)\big) = \mathsf{ThreshDice}(\mathsf{Bigrams}(a), \mathsf{Bigrams}(b)).$$

□

C. Private approximate matching for records linkage

We realize Functionality 1 in Protocol 1 using Sub-protocols 1 and 2.

Protocol 1: Implementing Functionality 1 (Approximate string matching for records linkage)

- **Public parameters:** Public key $\mathcal{PK}$. Maximum set cardinality μ. Encrypted set-intersection cardinality $c = [\![\mathsf{Bigrams}(a) \cap \mathsf{Bigrams}(b)]\!]$
- **Private inputs:** Party P_A holds a list of strings $A = [a_1, \ldots, a_n]$, and private key $\mathcal{SK}$. Party P_B holds a list of strings $B = [b_1, \ldots, b_m]$.
- **The protocol:**
  1) P_A and P_B run Sub-protocol 1 on their respective private inputs A, B, producing encrypted set intersection cardinalities r_{ij} for $1 \leq i \leq n$ and $1 \leq j \leq m$.
  2) For each r_{ij}, P_A and P_B run Sub-protocol 2 on their respective private inputs a_i, b_j, producing encrypted threshold Dice coefficient d_{ij}.
  3) For each d_{ij}, P_B sends $\langle d_{ij}, i \rangle$ to P_A in randomly shuffled order.
- **Output:** For all $\langle d_{ij}, i \rangle$ where $\mathsf{Dec}(d_{ij}) = 1$, P_A outputs i.

Theorem 6 (Correctness (Protocol 1))**.** *Protocol 1 implements Functionality 1 for approximate string matching for records linkage, i.e., for all strings $a_i \in A$ and $b_j \in B$, P_A learns the index i of any string $a_i \in A$ for which* $\mathsf{ThreshDice}\big(\mathsf{Bigrams}(a_i), \mathsf{Bigrams}(b_i), t\big) = 1$.

Proof. The proof follows from the proofs of Theorems 4 and 5. □

D. Privacy of approximate string matching protocol

We now prove Protocol 1 securely realizes Functionality 1 in the semi-honest adversary model. We employ a simulation-based approach to prove that, for all possible inputs A by P_A (resp. B by P_B), all possible messages received from P_B (resp. P_A), and all internally generated randomness, P_A's (resp. P_B's) view of the protocol execution can be efficiently simulated given only P_A's (resp. P_B's) own input and output.

Theorem 7 (P_A's privacy)**.** *Let P_A's input to the protocol $x_A = (A, \mathcal{PK}, \mathcal{SK}, \mu)$ and P_B's input to the protocol be $x_B = B$, and their combined input be denoted as x. Let $\mathsf{View}_{P_B}(x)$ represent P_B's view of the protocol during execution of Protocol 1. P_B has no output from Functionality 1 which we denote as $f_B(x) = \perp$. Let $F_1(x)$ denote the output of Functionality 1 on the combined input x. P_A outputs $f_A(x) = f_1(x)$. There exists a probabilistic polynomial-time algorithm P_B^* such that*

$$\big[P_B^*(x_B, \perp), F_1(x)\big] \stackrel{c}{\equiv} \big[\mathsf{View}_{P_B}(x), \mathsf{Output}_{P_A}(x)\big].$$

Proof. Proving P_A's privacy is straightforward, as P_B only ever receives a fixed number of ciphertexts from P_A. By the semantic security of the encryption scheme (see Theorem 1), a ciphertext is computationally indistinguishable from a uniform value in $\mathbb{Z}_n^*$. To simulate P_B's view in Sub-protocol 1, for each $a_i \in A$, P_B^* samples $C_=[r_1, \ldots, r_{728}]$ where each $r_j \leftarrow_\$ \mathbb{Z}_n^*$. The rest of P_B's view of Sub-protocol 1 (Step 2 and Output) is simulated by computing the r_{ij}s directly from the respective C_i's, P_B's input B, and $\mathcal{PK}$. To simulate P_B's view in each invocation of Sub-protocol 2, for each $[\![\theta_i]\!]$, P_B^* samples $r_i \leftarrow_\$ \mathbb{Z}_n^*$. The rest of P_B's view of Sub-protocol 2 is simulated by computing $\mathsf{Eval}(r_i \cdot c)$ from b, c and r_i. P_B's view of Steps 1 and 2 of Protocol 1, therefore, can be simulated by simulating Sub-protocols 1 and 2 as described above. The remainder of P_B's view of Protocol 1 (i.e., Step 3) can be computed directly from the d_{ij}'s output of Sub-protocol 2. □

Theorem 8 (P_B's privacy)**.** *Let $\mathsf{View}_{P_A}(x)$ represent P_A's view of the protocol during execution of Protocol 1, with all other inputs and outputs defined as in P_A's case above. There exists a probabilistic polynomial-time algorithm P_A^* such that*

$$\big[P_A^*(x_A, F_1(x)), \perp\big] \stackrel{c}{\equiv} \big[\mathsf{View}_{P_A}(x), \perp\big].$$

Proof. P_B's privacy relies on the blinding properties of the Eval function of CS. To simulate P_A's view of Sub-protocol 1, P_A^* computes each C_i directly from each a_i. P_A does not see step 2 or the output, concluding P_A's view of Sub-protocol 1. To simulate P_A's view of an instance of Sub-protocol 2, P_A^* computes steps 1-3 directly from its private input string a and public inputs. P_A does not see steps 4-6. In Step 3 of Protocol 1, P_A receives and decrypts all d_{ij}s received from P_B, and outputs the index i of any d_{ij} for which $\mathsf{Dec}(d_{ij}) = 1$. To simulate this, P_A^* consults the output $F_1(x)$ to see which records $a_i \in A$ were found to be an approximate match of some record in B. For all such a_i, it chooses a uniform random $1 \le j \le m$, and a random $\alpha \leftarrow_\$ \mathbb{Z}_s^*$ such that $\mathsf{QR}_s(\alpha) = 1$ and computes $d_{ij} \leftarrow \mathsf{Enc}(\alpha) = g^\alpha h^r \bmod n$. For all other d_{ij}, it chooses a uniform random $1 \le j \le m$, and a random $\beta \leftarrow_\$ \mathbb{Z}_s^*$ such that $\mathsf{QR}_s(\alpha) = 0$ and computes $d_{ij} \leftarrow \mathsf{Enc}(\beta) = g^\beta h^r \bmod n$. By Theorem 6, $\mathsf{Dec}(d_{ij}) = 1$ if records $a_i \in A$ and $b_j \in B$ approximately match ($\mathsf{Dec}(d_{ij}) = 0$ otherwise). By Theorem 3 (blinding hiding), for all $d_{ij} = g^m h^r \bmod n$, m is a uniform quadratic residue in $\mathbb{Z}_s^*$ if the records match, and therefore indistinguishable from the simulated value α. Similarly m is a uniform quadratic non-residue if the records do not match, and therefore indistinguishable from β. □

TABLE III
PERFORMANCE OF CRYPTOSYSTEM $\mathcal{CS}$.

Security level (bits)	Time (ms) Enc	Blind	Dec
112 ($\lvert n\rvert = 2048, \lvert u\rvert = 224$)	0.30	0.35	0.11
128 ($\lvert n\rvert = 3072, \lvert u\rvert = 256$)	0.61	0.74	0.28
192 ($\lvert n\rvert = 7680, \lvert u\rvert = 384$)	3.9	4.1	2.1
256 ($\lvert n\rvert = 15360, \lvert u\rvert = 512$)	15.5	16.6	8.8

VII. Implementation and Performance

In line with related work [4], [10] we used the North Carolina Voter Registration dataset[2] maintained by the North Carolina State Board of Elections. The list contained 7.8M records at the time of writing. Records consist of a voter's name, address, and other demographic data. We used the `last_name` column to create our dataset.

A. Implementing $\mathcal{CS}$ and Protocol 1

We implemented the encryption scheme $\mathcal{CS}$ from Section V and Protocol 1 in Python.[3] We used the `gmpy2` library for faster modular exponentiation. We implemented the outer encryption using DGK, which offers fast encryption/decryption due to small randomizer subgroups h, as well as the ability to perform decryption modulo p instead of modulo n. We precomputed powers of g and h for faster exponentiation and precomputed $\log_s g^m$ and $\mathsf{QR}_s(m)$ for faster decryption. We benchmarked on an Intel Xeon E5-2697A @ 2.60GHz. For key size $|n| = 2048$, we achieved an encryption/decryption round trip time of under $500\,\mu s$. With $|n| = 15,360$, the round trip time was in the low milliseconds. The performance of $\mathcal{CS}$ is given in Table III.

[2]https://www.ncsbe.gov/data-stats/other-election-related-data
[3]https://github.com/aleksessex/Residue-Homomorphic-Encryption

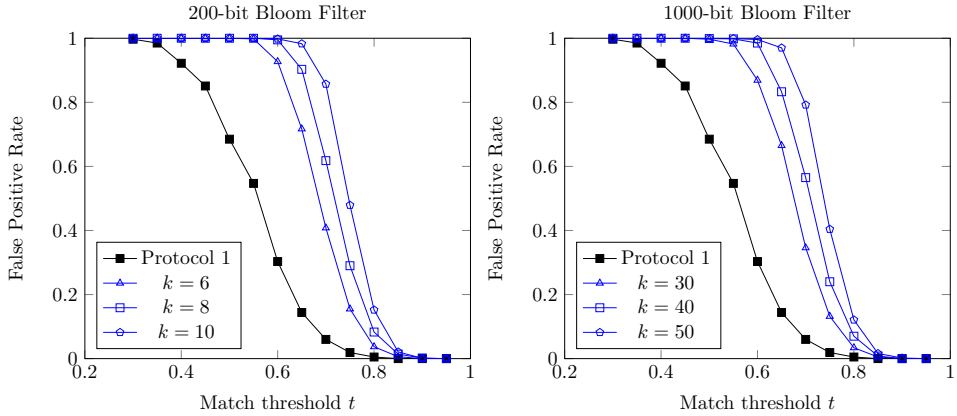

Fig. 1. Matching accuracy of Protocol 1 relative to various Bloom filter encoding parameterizations.

B. Computing Parameters to Approximate τ_t

Protocol 1 requires suitable parameterizations to 2μ-approximate threshold function τ_μ. Finding appropriate group orders s and offsets f was done through a brute-force search by examining successively larger primes for suitable subsequences of quadratic residues. For each successively larger prime s, the quadratic residuosity of $2, \ldots, (s-2)$ modulo s was computed, and the resulting sequence was searched for any subsequences consisting of μ quadratic non-residues, followed by μ quadratic residues. We found instead that approximating the *complement* of τ consistently yielded smaller minimum group orders. Table V lists computationally derived minimal parameters for 2μ-approximating threshold function $\overline{\tau_\mu}$, i.e., the *complement* of τ_μ. These parameters can be used to implement Protocol 1 by changing the output such that P_A outputs i if $\mathsf{Dec}(d_{ij}) = 0$ (as opposed to 1).

The longest surname in the NC voter registry is 25 characters, and Table V accommodates approximate matching up to $\mu = 26$. Although our approach can accommodate all real-world surnames found in the dataset, future work will focus on developing new methods to extend the upper bound on μ, to allow matching of longer strings in other applications.

C. Matching Accuracy

The Bloom filter encoding literature typically employs stochastic "data corrupting" algorithms to introduce realistic typographic and transcription errors. As our protocol implements the threshold Dice coefficient, by definition there are no matching errors relative to the ideal functionality, and thus we do not examine the issue of corruptions.

A more meaningful comparison is the matching accuracy of Bloom filter encodings against the ideal functionality in the *absence* of corruptions. Therefore we are interested only in how often distinct records trigger a match (false positives). We randomly selected $|A| = |B| = 1000$ unique names from the North Carolina list and matched using Protocol 1 with BFEs of different sizes. Each record $a_i \in A$ was compared against each record $b_i \in B$. If a_i was found to match one or more records in B, a false positive was recorded. The number of records in A that generated a false positive was divided by $|A|$ to give the false positive rate. For BFEs, we set the number of hash functions k and filter bit length ℓ based on the same ratios of k/ℓ explored by Kuzu et al. [4]. The results of the comparison are shown in Figure 1, demonstrating the accuracy of our scheme relative to BFEs at the same match threshold, especially in Grzebala and Cheatham's recommended range of $0.55 \leq t \leq 0.75$.

D. Record Linking Experiment

Finally, we conducted a proof-of-concept linkage experiment on two medium-sized databases of $|A|, |B| = 20,000$ records. We used a Digital Ocean high-cpu droplet with Intel Xeon CPU E5-2697A @ 2.60GHz, 32CPUs, 46GB memory, and 1 TB drive. We capped name lengths at 20 characters, and used a Dice coefficient threshold of 0.9 with a $|n| = 2048$-bit modulus. As an additional optimization, P_A sent $|\mathsf{Bigrams}(a_i)|$ in plaintext, allowing P_B to decide whether a match would be possible for two records of length $|A|, |B|$ at the given threshold T, and to skip it otherwise.

The protocol consists of three stages: encrypting bigram sets on P_A's end, homomorphic matching on P_B's end, and finally decryption on P_A's end. Each stage consists of numerous independent cryptographic operations, which we broke into multiple jobs and ran as separate Python processes. The performance of each stage is presented in Table IV.

Bigram encryption took just under 4 minutes and generated 5 GB of ciphertext data. Homomorphic matching took 1.35 hours and generated about 71 GB of ciphertext data. Decryption took 22 minutes. The total linking time was 1.8 hours at a total cloud computing cost of $1.90 (USD).

VIII. Conclusion

We introduced a novel cryptographic construction to homomorphically approximate a threshold function in a restricted domain, and a protocol to securely implement the threshold Dice coefficient. Relative to Bloom filter encodings, our scheme offers formal security guarantees and improved accuracy. Through this work, we hope to initiate a new avenue of research for the challenge of secure approximate matching.

TABLE IV
RUNNING TIME (S) OF PROTOCOL 1 AT VARYING MATCH THRESHOLDS t.

	$\|A\|, \|B\| = 100$ Records			$\|A\|, \|B\| = 1000$ Records		
	$t = 0.85$	$t = 0.9$	$t = 0.95$	$t = 0.85$	$t = 0.9$	$t = 0.95$
Encrypt record bigrams (P_A)	39	39	39	382	382	382
Homomorphic Dice threshold (P_B)	5.6	3.0	1.9	535	435	184
Decrypt results (P_A)	1.4	1.2	0.4	141	104	43

TABLE V
MINIMAL PARAMETERS FOR 2μ-APPROXIMATING $\overline{\tau_\mu}$.

Max set size (μ)	Offset (f)	Group order (s)
3	3	11
4	26	59
5	25	59
6	60	131
7	59	131
8	58	131
9	897	1811
10	1460	2939
11	1459	2939
12	5994	12011
13	5993	12011
14	5992	12011
15	5991	12011
16	33230	66491
17	33229	66491
18	33228	66491
19	74051	148139
20	74050	148139
21	137805	275651
22	475904	951851
23	475903	951851
24	1134846	2269739
25	1134845	2269739
26	1134844	2269739

REFERENCES

[1] P. Grzebala and M. Cheatham, "Private record linkage: Comparison of selected techniques for name matching," in *International Semantic Web Conference*. Springer, 2016, pp. 593–606.

[2] R. Schnell, T. Bachteler, and J. Reiher, "Privacy-preserving record linkage using bloom filters," *BMC medical informatics and decision making*, vol. 9, no. 1, p. 41, 2009.

[3] O. Papapetrou, W. Siberski, and W. Nejdl, "Cardinality estimation and dynamic length adaptation for bloom filters," *Distributed and Parallel Databases*, vol. 28, no. 2-3, pp. 119–156, 2010.

[4] M. Kuzu, M. Kantarcioglu, E. Durham, and B. Malin, "A constraint satisfaction cryptanalysis of bloom filters in private record linkage," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2011, pp. 226–245.

[5] M. Kroll and S. Steinmetzer, "Automated cryptanalysis of bloom filter encryptions of health records," *8th International Conference on Health Informatics*, 2014.

[6] P. Christen, R. Schnell, D. Vatsalan, and T. Ranbaduge, "Efficient cryptanalysis of bloom filters for privacy-preserving record linkage," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2017, pp. 628–640.

[7] E. A. Durham, M. Kantarcioglu, Y. Xue, C. Toth, M. Kuzu, and B. Malin, "Composite bloom filters for secure record linkage," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 12, pp. 2956–2968, 2014.

[8] D. Vatsalan, P. Christen, and E. Rahm, "Scalable privacy-preserving linking of multiple databases using counting bloom filters," in *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*. IEEE, 2016, pp. 882–889.

[9] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: an efficient and scalable protocol," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 789–800.

[10] M. Kuzu, M. Kantarcioglu, E. A. Durham, C. Toth, and B. Malin, "A practical approach to achieve private medical record linkage in light of public resources," *Journal of the American Medical Informatics Association*, vol. 20, no. 2, pp. 285–292, 2012.

[11] D. Vatsalan, P. Christen, and V. S. Verykios, "A taxonomy of privacy-preserving record linkage techniques," *Information Systems*, vol. 38, no. 6, pp. 946–969, 2013.

[12] K. El Emam and F. K. Dankar, "Protecting privacy using k-anonymity," *Journal of the American Medical Informatics Association*, vol. 15, no. 5, pp. 627–637, 2008.

[13] K. El Emam, *Guide to the De-Identification of Personal Health Information*. CRC Press., 2013.

[14] C. Hazay and T. Toft, "Computationally secure pattern matching in the presence of malicious adversaries," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2010, pp. 195–212.

[15] D. Vergnaud, "Efficient and secure generalized pattern matching via fast fourier transform," in *International Conference on Cryptology in Africa*. Springer, 2011, pp. 41–58.

[16] M. Chase and E. Shen, "Substring-searchable symmetric encryption," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 263–281, 2015.

[17] V. Kolesnikov, M. Rosulek, and N. Trieu, "Swim: Secure wildcard pattern matching from ot extension," Cryptology ePrint Archive, Report 2017/1150, 2017, https://eprint.iacr.org/2017/1150.

[18] X. Wei, M. Zhao, and Q. Xu, "Efficient and secure outsourced approximate pattern matching protocol," *Soft Computing*, vol. 22, no. 4, pp. 1175–1187, 2018.

[19] S. M. Randall, A. P. Brown, A. M. Ferrante, J. H. Boyd, and J. B. Semmens, "Privacy preserving record linkage using homomorphic encryption," *Population Informatics for Big Data, Sydney, Australia*, 2015.

[20] J. H. Cheon, M. Kim, and K. Lauter, "Homomorphic computation of edit distance," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 194–212.

[21] D. Vatsalan and P. Christen, "Scalable privacy-preserving record linkage for multiple databases," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 1795–1798.

[22] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[23] F. Niedermeyer, S. Steinmetzer, M. Kroll, and R. Schnell, "Cryptanalysis of basic bloom filters used for privacy preserving record linkage," *Journal of Privacy and Condentiality*, vol. 6, no. 2, pp. 59–79, 2014.

[24] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 2112–2120.

[25] S. M. Randall, A. M. Ferrante, J. H. Boyd, J. K. Bauer, and J. B. Semmens, "Privacy-preserving record linkage on large real world datasets," *Journal of biomedical informatics*, vol. 50, pp. 205–212, 2014.

[26] K. Schmidlin, K. M. Clough-Gorr, and A. Spoerri, "Privacy preserving probabilistic record linkage (p3rl): a novel method for linking existing health-related data and maintaining participant confidentiality," *BMC medical research methodology*, vol. 15, no. 1, p. 46, 2015.

[27] ——, "Privacy preserving probabilistic record linkage (p3rl): a novel method for linking existing health-related data and maintaining participant confidentiality," *BMC medical research methodology*, vol. 15, no. 1, p. 46, 2015.

[28] I. Lazrig, T. C. Ong, I. Ray, I. Ray, X. Jiang, and J. Vaidya, "Privacy preserving probabilistic record linkage without trusted third party," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2018, pp. 1–10.

[29] I. Damgård, M. Geisler, and M. Krøigaard, "Efficient and secure comparison for on-line auctions," in *Australasian Conference on Information Security and Privacy*. Springer, 2007, pp. 416–430.
[30] J. Benaloh, "Dense probabilistic encryption," in *Proceedings of the workshop on selected areas of cryptography*, 1994, pp. 120–128.
[31] J. Groth, "Cryptography in subgroups of $\mathbb{Z}_n^*$," in *Theory of Cryptography Conference*. Springer, 2005, pp. 50–65.
[32] J.-S. Coron, A. Joux, A. Mandal, D. Naccache, and M. Tibouchi, "Cryptanalysis of the rsa subgroup assumption from tcc 2005," in *International Workshop on Public Key Cryptography*. Springer, 2011, pp. 147–155.
[33] E. De Cristofaro, P. Gasti, and G. Tsudik, "Fast and private computation of cardinality of set intersection and union," in *International Conference on Cryptology and Network Security*. Springer, 2012, pp. 218–231.
[34] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 1–19.