Partial Fraction Techniques for Cryptography

Charanjit S. Jutla[1], Rohit Nema[2], and Arnab Roy[3]

[1] IBM Research, USA csjutla@us.ibm.com
[2] Stanford University rnema@cs.stanford.edu
[3] Mysten Labs arnab@mystenlabs.com

Abstract. Partial fraction decomposition is a fundamental technique in mathematics where products of rational functions can be expressed as sums of fractions. While rational functions have been used in various cryptographic constructions, their rich algebraic structure has not been systematically explored as a direct foundation for building cryptographic primitives. In this work, we describe and exploit two key properties of partial fraction decomposition: (1) the decomposition property itself, which enables efficient set membership testing, and (2) a novel linear independence property arising from the non-singularity of Cauchy matrices, which enables threshold cryptography.

We present two main applications. First, we construct a key-value commitment scheme where a dictionary is represented as a linear combination of partial fractions. Our scheme achieves constant-size commitments (a single group element) and proofs, supports homomorphic updates enabling stateless operation, and provides efficient membership and non-membership proofs through simple pairing equations. We also introduce *Credential-based Key-Value Commitments*, where keys are registered via Boneh-Boyen signatures, enabling applications in permissioned settings.

Second, we construct a dynamic threshold encryption scheme leveraging the linear independence of partial fraction products. Our scheme achieves compact ciphertexts, supports public preprocessing of public keys to a succinct encryption key, enables dynamic threshold selection at encryption time, and provides robustness through share verification without random oracles. In particular, we achieve the shortest CPA-secure ciphertext size of 3 group elements, given logarithmic size preprocessed encryption key.

We prove security of our constructions in the standard model under new q-type assumptions and establish their generic hardness in the generic bilinear group model. Our work demonstrates that working directly with the algebraic structure of rational fractions, rather than converting to polynomial representations, yields elegant and efficient cryptographic constructions with concrete advantages over prior work.

1 Introduction

Many of us have encountered partial fraction decomposition in high school algebra, especially while working out calculus problems. The fundamental observation that products of distinct rational functions can be decomposed into sums of rational fractions has been a cornerstone of mathematical education for centuries:

$$\frac{1}{X+2} \cdot \frac{1}{X+3} = \frac{1}{X+2} - \frac{1}{X+3}$$

While rational functions have been used to construct cryptographic primitives [4, 13, 19, 22, 23, 32], the rich algebraic structure of rational functions, particularly their behavior

under multiplication and emergent linear independence properties, has remained largely unexplored as a direct foundation for cryptographic constructions.

In this work, we systematically explore partial fraction decomposition as a cryptographic building block. We describe and exploit two key mathematical properties: (1) the decomposition property itself, which enables set membership testing through the existence or non-existence of valid partial fraction expansions; and (2) a novel linear independence property based on Cauchy matrices[4], which states that when a fixed sum of rational functions is multiplied with distinct other rational functions, the resulting combinations are linearly independent over the base field. These properties enable elegant constructions of key-value commitments and dynamic threshold encryption with concrete efficiency advantages.

We explain the second property with a concrete example, while a formal treatment will be developed in the technical sections. Consider the following partial fraction decompositions:

$$\left(\frac{1}{X+2}+\frac{1}{X+3}+\frac{5}{6}\right)\cdot\frac{1}{X+5}=\frac{1}{3}\cdot\frac{1}{X+2}+\frac{1}{2}\cdot\frac{1}{X+3}$$
$$\left(\frac{1}{X+2}+\frac{1}{X+3}+\frac{9}{20}\right)\cdot\frac{1}{X+7}=\frac{1}{5}\cdot\frac{1}{X+2}+\frac{1}{4}\cdot\frac{1}{X+3}$$

It can be readily seen that the partial fraction decompositions on the right are linearly independent over the rational numbers. In general, when we have n rational fractions on the left, and we multiply with n distinct rational fractions which do not appear in those sums, then we get n linearly independent rational decompositions. The linear independence is connected to the non-singularity of related Cauchy matrices. Note that there are auxiliary rational numbers (5/6 and 9/20 in the above example) that appear as affine shifts that are determined by the particular rational functions in the multiplicands. Further, looking ahead to its use in bilinear pairing-based cryptography, over finite fields of prime order p, a partial fraction is equivalently[5] a polynomial in X of degree $p-2$, and each partial fraction then gives a linearly independent such polynomial. This allows us to formulate new powerful (falsifiable) q-type hardness assumptions (provable in the generic bilinear group model).

Key-Value Commitments from Rational Functions. We construct a Key-Value Commitment (KVC) scheme by representing a dictionary $D=\{(k_i,v_i)\}_{i\in[n]}$ as a linear combination of rational functions: $C_D(X):=\sum_{i\in[n]}\frac{v_i}{X+k_i}$. In the cryptographic instantiation using bilinear groups with secret x, the commitment is the single group element $[C_D(x)]_1$[6]. Membership of a key-value pair (k,v) can be verified by checking whether $\left(C_D(X)-\frac{v}{X+k}\right)\cdot\frac{1}{X+k}$ decomposes into a sum of linear rational functions, which translates to a simple pairing equation. Our scheme achieves:

– **Constant-size commitments and proofs:** The commitment is a single $\mathbb{G}_1$ element, and membership proofs consist of $2|\mathbb{G}_1|$ bits. This compares favorably with Agrawal and Raghuraman's KVaC [1], which requires 2 group elements (in groups of unknown order)

[4] We remark that Cauchy matrix codes as being maximum-distance-separable (MDS) has been used for designing symmetric-key block ciphers [28].
[5] If we consider the partial fraction modulo $X^{p-1}-1$, given that all $x\in\mathbb{Z}_p^*$ satisfy $X^{p-1}-1=0$.
[6] We adopt additive notations for groups, with $[a]_\ell$ indicating $a\cdot g_\ell\in\mathbb{G}_\ell$.

for commitments and 3 for proofs, but operates in RSA/class groups rather than prime-order pairing groups.[7]

- **Homomorphic updates:** To add a key-value pair (k, v) or update an existing value, we simply compute $\mathsf{com}' = \mathsf{com} + v \cdot \left[\frac{1}{x+k}\right]_1$. This homomorphism enables stateless operation after initialization—updates require only the current commitment, not the full dictionary state. Existing proofs can also be updated homomorphically using only the update information. Compared to the expressive set accumulator of Zhang et al. [33], which supports rich operations (intersection, union, SUM, COUNT, MIN, MAX) using polynomial commitments, our construction offers a more direct algebraic foundation specifically optimized for key-value operations with homomorphic properties. While their scheme targets general set expressions over polynomial-sized domains, ours focuses on efficient dictionary operations defined over arbitrary domains with simpler proofs.
- **Credential-based variant:** We introduce Credential-based KVC (CbKVC), where keys must first be registered with a registrar who provides Boneh-Boyen signatures [4], $\left[\frac{1}{x+k}\right]_1$, $\left[\frac{1}{x+k}\right]_2$, on the key k. This enables applications in permissioned settings and provides a connection to signature-based constructions.
- **Algebraic construction:** Our construction is overwhelmingly algebraic. While [15] was the first pairing-based KVC construction, we avoid additional non-algebraic operations like cuckoo-hashing, and support stateless updates as a result.

Dynamic Threshold Encryption via Linear Independence. Our second main contribution leverages the linear independence property of rational functions combined with Cauchy matrices. For an authorized set S and threshold t, we construct an encryption key from a sum of rational functions: $P(X) = \sum_{j \notin S} \frac{1}{X+j} + g(X)$, where $\frac{1}{X+j}$ are related to user keys, and $g(X)$ is another rational function that masks the message.

The key insight is that for each authorized party $i \in S$, the product $P(X) \cdot \frac{1}{X+i}$ can be decomposed to yield a linearly independent combination. Given enough such combinations, the combiner solves a fully defined linear system (structured as a *special* Cauchy matrix) to recover $g(X)$ and decrypt the message. Our dynamic threshold encryption scheme achieves:

- **Public preprocessing:** The encryption key for an authorized set S of size s can be preprocessed publicly into $O(\log s)$ group elements (using techniques from Waters and Wu [31]), independent of threshold t. The threshold can be chosen dynamically at encryption time. We can efficiently update the encryption key using a single group operation without having to rerun quadratic or even linear-time preprocessing, unlike [12, 17, 31].
- **Compact ciphertexts and shares:** Ciphertexts consist of $2|\mathbb{G}_1| + |\mathbb{G}_T|$ (vs. $9|\mathbb{G}| + |\mathbb{G}_T|$ for Garg et al. [17] and $3|\mathbb{G}| + |\mathbb{G}_T| + |\mathbb{F}|$ for Waters and Wu [31]). Partial decryption shares are $2|\mathbb{G}_1| + 2|\mathbb{G}_2|$ (vs. $1|\mathbb{G}|$ for Garg et al. and $2|\mathbb{G}|$ for Waters and Wu).
- **Robustness without random oracles:** We achieve share verification using pairing equations, without requiring random oracles (ROs)—unlike Delerablée-Pointcheval [12] which need RO for robustness.
- **Simple algebraic structure:** Our construction uses algebraic properties of partial fraction decomposition and linear independence via Cauchy matrices, rather than SNARKs, or polynomial commitments, for example as required by Garg et al. [17].

[7] In this work, we assume a key is not in the dictionary if its value is 0 while [1] does not have this simplification. We can easily remove this simplification which yields a commitment size of 2 $\mathbb{G}_1$ elements.

– **Security against adaptive corruptions:** We prove CPA security of our construction with *adaptive corruptions*. Briefly, the adversary is allowed to adaptively corrupt parties before and after seeing the ciphertext. In the appendix, we also show that our scheme can be made CCA2 secure, by adding a few group elements to the ciphertext (see Appendix C).

Comparison with silent setup schemes. The recent threshold encryption schemes with silent setup [17, 31] achieve non-interactive key generation by having parties independently sample keys, enabling asynchronous operation and arbitrary thresholds (not limited by $t < n/3$ as in DKG-based schemes). While we assume a trusted setup[8] (or efficient DKG) to establish the shared secret x, our approach offers complementary advantages: simpler algebraic structure, smaller ciphertexts, and more direct security proofs from q-type assumptions. The combining operation requires $t + 3$ pairings (vs. 1 for Garg et al. and 3 for Waters-Wu), but preprocessing is linear (not quadratic), and our combining key is independent of S. We summarize the comparison in Table 1.

Comparison with dynamic threshold encryption. Delerablée-Pointcheval [12] introduced dynamic threshold public-key encryption where users can join dynamically, senders choose authorized sets per ciphertext, and thresholds are set at encryption time—achieving the first constant-size ciphertexts for this setting. Our scheme provides similar dynamic properties but with public preprocessing capability, homomorphic encryption keys, and robustness in the standard model. Their security relies on the MSE-DDH assumption (Multi-Sequence of Exponents DDH), while we use novel q-type assumptions provably secure in the generic bilinear group model. We also include comparison with [12] in Table 1.

Related work on logarithmic derivatives. The connection between rational functions and logarithmic derivatives has been observed in various contexts, including Haböck's LogUp protocol [22]. However, prior work primarily uses this connection conceptually or converts to polynomial formulations for actual constructions. We are the first to work directly with the algebraic structure of rational functions to achieve cryptographic goals. The summation formulation enables easy unions in bilinear groups (which products cannot achieve directly), while the linear independence properties enable threshold constructions. Our work demonstrates that partial fraction decompositions offer unique advantages as a native cryptographic primitive rather than merely an alternative representation.

Additional Contributions and Applications. Beyond the core constructions, our work makes several additional contributions:

– We prove security of our schemes under new q-type assumptions (q-SqPole, q-DistPole and q-RankDef) and establish their hardness in the generic bilinear group model.
– In Appendix C, we show that our dynamic threshold encryption scheme can be made CCA2-secure, by adding a few group elements to the ciphertext using one-time simulation-sound NIZKs [29]. This even allows public verification of "valid" ciphertexts.[9]

[8] While the silent setup schemes such as [17, 31] allow pre-processing for a set S without the need of a trusted party, it can be argued that its utility for a dynamically changing S is contingent upon an authenticated bulletin-board maintaining this pre-processed information.

[9] To formally define publicly-verifiable CCA2-encryption schemes, one just extends the standard IND-CCA2 definition of encryption with a public verification function V which takes the public key and a potential ciphertext as arguments, and it returns true iff the decryption function when supplied with the same ciphertext does not return "invalid ciphertext" [24].

	Delerablée and Pointcheval [12]	Garg et al. [17]	Waters and Wu [31]	This work
$\lvert\mathsf{ek}\rvert$	$O(n)$	$O(1)$	$O(\log n)$	$O(\log s)$
$\lvert\mathsf{pp}\rvert$	–	$O(n)$	$O(n \log n)$	$O(\log n)$
$\lvert\mathsf{pk}\rvert$	$O(1)$	$O(n)$	$O(n)$	$O(1)$
$\lvert\mathsf{ct}\rvert$	$2\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$	$2\lvert\mathbb{G}_1\rvert + 7\lvert\mathbb{G}_2\rvert + \lvert G_T\rvert$	$3\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	$2\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$ ($+3\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$ for CCA)
$\lvert\partial\rvert$	$3\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	$1\lvert\mathbb{G}_2\rvert$	$2\lvert\mathbb{G}\rvert$	$2(\lvert G_1\rvert + \lvert\mathbb{G}_2\rvert)$
ciphertext validation?	✓	✗	✗	✓
$\lvert\mathsf{ck}\rvert$	$O(n)$	$O(s)$	$O(s)$	$O(n)$
# of pairings for decryption	$O(t)$	0	3	$O(t)$
# of pairings for verifying share	2	2	2	3
No RO?	✗	✗	✓	✓
Adaptive?	✗	✓	✗	✓
Assumption	(static) q-type	generic group	(static) q-type	(adaptive) q-type

Table 1: Comparison of Dynamic Threshold Encryption schemes. Here, n is the maximum number of users, s is the size of the authorized set, and t is the decryption threshold, ek is the (preprocessed, if supported) encryption key, pp is the public parameters used for preprocessing (where for [17] and [31], this is the crs), ct is the ciphertext, ∂ is the partial decryption share, and ck is the combining/aggregation key. We count the number of pairings for decryption across partial decryption and combining/aggregation.

– We describe a couple of other constructions based on partial fraction decomposition in Appendix D: Hadamard product proofs, and registration-based encryption (RBE). While we do not fully develop these applications in this paper, we hope the patterns are instructive for future work.

2 Technical Overview

We begin by providing an overview of partial fraction decomposition and its properties that we leverage in this work. We then proceed to apply it to build existing and novel cryptographic primitives as described below. Each construction highlights a characteristic feature that we hope inspires future interest and work.

2.1 Partial Fraction Decomposition

Partial fraction decomposition is a fundamental technique in mathematics with applications spanning calculus, complex analysis, and algebra. The technique allows us to express rational functions as sums of simpler rational functions, making complex computations more tractable.

Mathematical Foundation. Let $\mathbb{F}$ be a field and consider the field of fractions $\mathbb{F}(X)$ of its polynomial ring $\mathbb{F}[X]$. The principal observation is that any product of distinct linear rational functions can be decomposed into a sum of simpler rational functions. Specifically, for distinct elements $a_1, \ldots, a_n \in \mathbb{F}$, we have the following decomposition:

Lemma 1 (Partial Fraction Decomposition). *Suppose $n \geq 2$. Let $a_1, \ldots, a_n \in \mathbb{F}$ be distinct. Then, there exist unique coefficients $c_1, \ldots, c_n \in \mathbb{F}$ such that,*

$$\prod_{i=1}^{n} \frac{1}{X + a_i} = \sum_{i=1}^{n} \frac{c_i}{X + a_i}$$

where $c_i = \prod_{j \neq i} \frac{1}{a_j - a_i}$ for each $i \in [n]$.

Proof. We prove the existence and uniqueness of the decomposition.

Existence: Let $g(X) = \sum_{i=1}^{n} \frac{c_i}{X+a_i}$, with c_i as given in the lemma. Now, we have $g(X) \prod_{i=1}^{n}(X + a_i) = \sum_{i=1}^{n} c_i \prod_{j \neq i}(X + a_j)$. Observe that $\sum_{i=1}^{n} c_i \prod_{j \neq i}(X + a_j)$ is a degree $(n-1)$ polynomial, which is equal to 1 at the n distinct points $X = -a_1, \cdots, -a_n$. Hence it must be identically 1, and we are done.

Uniqueness: Suppose there exist coefficients $d_1, \ldots, d_n$ such that:

$$\sum_{i=1}^{n} \frac{c_i}{X + a_i} = \sum_{i=1}^{n} \frac{d_i}{X + a_i}$$

Then $\sum_{i=1}^{n} \frac{c_i - d_i}{X+a_i} = 0$. Since the functions $\{1/(X + a_i)\}_{i=1}^{n}$ are linearly independent over $\mathbb{F}$ (as they have distinct poles), we must have $c_i = d_i$ for all $i \in [n]$. □

Ring of Linear Rational Fractions. We now describe the algebraic structure of the $\mathbb{F}$-algebra generated by the linear rational fractions. Consider the subalgebra of the rational function field $\mathbb{F}(X)$ generated by the elements $\{(X + a_i)^{-1}\}_{i=1}^{n}$:

$$R = \mathbb{F}\left[\frac{1}{X + a_1}, \ldots, \frac{1}{X + a_n}\right] \subseteq \mathbb{F}(X).$$

While it might seem natural to expect $R \cong \mathbb{F}[X_1, \ldots, X_n]$ (identifying each $\frac{1}{X+a_i}$ with a formal variable X_i), the generators are *not* algebraically independent. They satisfy the bilinear relations

$$\frac{1}{X + a_i} - \frac{1}{X + a_j} = (a_j - a_i) \cdot \frac{1}{X + a_i} \cdot \frac{1}{X + a_j}$$

for all $i \neq j$. These relations define the structure of R. We have the following ring isomorphism:

$$\mathbb{F}\left[\frac{1}{X + a_1}, \ldots, \frac{1}{X + a_n}\right] \cong \frac{\mathbb{F}[X_1, \ldots, X_n]}{\langle (a_i - a_j)X_iX_j + X_i - X_j : i \neq j \rangle},$$

where the right-hand side is the quotient of the polynomial ring $\mathbb{F}[X_1, \ldots, X_n]$ by the ideal generated by $(a_i - a_j)X_iX_j + X_i - X_j$ for all pairs $i \neq j$. This quotient ring is the coordinate ring of the algebraic variety defined by these relations.

Set Membership Testing. The partial fraction decomposition provides a natural mechanism for set membership testing. Given a set $S = \{a_1, \ldots, a_n\}$ and an element $b \in \mathbb{F}$, we can test whether $b \in S$ by examining the decomposition of the product $\left(\sum_{i=1}^{n} \frac{1}{X+a_i}\right)\left(\frac{1}{X+b}\right)$.

Property 1 (Set (Non-)Membership via Partial Fractions).

Let $S = \{a_1, \ldots, a_n\}$ be a set of distinct elements in $\mathbb{F}$ and let $b \in \mathbb{F}$. Then,

- If $b \notin S$, then $\left(\sum_{i=1}^{n} \frac{1}{X+a_i}\right)\left(\frac{1}{X+b}\right)$ can be expressed as a sum of partial fractions $\sum_{i=1}^{n} \frac{h_i}{X+a_i} + \frac{h_{n+1}}{X+b}$ for some coefficients $h_i \in \mathbb{F}$.
- If $b \in S$, then no such decomposition exists.

This property forms the foundation for our key-value commitment scheme, where we can efficiently prove membership or non-membership of key-value pairs in a committed dictionary.

Linear Independence Property. Another important property for our applications is the linear independence of certain products of partial fractions:

Lemma 2 (Linear Independence of Partial Fraction Products).
Let $\{c_1, \ldots, c_n\} \in \mathbb{F}$ be arbitrary. Let $S = \{a_1, \ldots, a_n\}$ be a set of distinct elements in $\mathbb{F}$ and let $b_1, \ldots, b_m \in \mathbb{F}$ be distinct elements such that $S \cap \{b_1, \ldots, b_m\} = \emptyset$. Then the products:

$$\left\{ \left(\sum_{i=1}^{n} \frac{c_i}{X + a_i} + \sum_{i=1}^{n} \frac{c_i}{b_j - a_i} \right) \cdot \frac{1}{X + b_j} \right\}_{j=1}^{m} = \left\{ \sum_{i=1}^{n} \frac{h_{ij}}{X + a_i} \right\}_{j=1}^{m}$$

are linearly independent over $\mathbb{F}$ if $m \leq n$. Moreover, $h_{ij} = \frac{c_i}{b_j - a_i}$.

Proof. We show that the rational functions $\left\{\sum_{i=1}^{n} \frac{h_{ij}}{X+a_i}\right\}_{j=1}^{m}$ are linearly independent over $\mathbb{F}$ by demonstrating that the coefficient matrix $\boldsymbol{H} = (h_{ij})$ has full column rank when $m \leq n$.

First, we verify the formula for h_{ij}. Using partial fraction decomposition:

$$\begin{aligned}
&\left(\sum_{i=1}^{n} \frac{c_i}{X + a_i} + \sum_{i=1}^{n} \frac{c_i}{b_j - a_i} \right) \cdot \frac{1}{X + b_j} \\
=&\sum_{i=1}^{n} \frac{c_i}{(X + a_i)(X + b_j)} + \frac{1}{X + b_j} \sum_{i=1}^{n} \frac{c_i}{b_j - a_i}. \\
=&\sum_{i=1}^{n} \frac{c_i}{b_j - a_i} \left(\frac{1}{X + a_i} - \frac{1}{X + b_j} \right) + \left(\sum_{i=1}^{n} \frac{c_i}{b_j - a_i} \right) \frac{1}{X + b_j} \\
=&\sum_{i=1}^{n} \frac{c_i}{b_j - a_i} \left(\frac{1}{X + a_i} \right)
\end{aligned}$$

Hence $h_{ij} = \frac{c_i}{b_j - a_i}$, confirming our formula. We can write $\boldsymbol{H} = \boldsymbol{D} \cdot \mathcal{C}(\boldsymbol{a}, \boldsymbol{b})^T$, where:

- $\boldsymbol{D} = \mathsf{diag}(c_1, \ldots, c_n) \in \mathbb{F}^{n \times n}$ is a diagonal matrix with the coefficients c_i on the diagonal,
- $\mathcal{C}(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{F}^{m \times n}$ is the Cauchy matrix with entries $\frac{1}{b_j - a_i}$ for $j \in [m], i \in [n]$.

Note that $\mathcal{C}(\boldsymbol{a}, \boldsymbol{b})^T \in \mathbb{F}^{n \times m}$ has entries $\frac{1}{b_j - a_i}$ in row i and column j.

When $m \leq n$, the Cauchy matrix $\mathcal{C}(\boldsymbol{a}, \boldsymbol{b})$ has full row rank m (see Lemma 3). Therefore, $\mathcal{C}(\boldsymbol{a}, \boldsymbol{b})^T$ has full column rank m.

If not all c_i are zero (which we can assume without loss of generality, as otherwise the statement is vacuous), then $\boldsymbol{D}$ is either invertible (if all $c_i \neq 0$) or has rank equal to the number of nonzero c_i. In either case, $\boldsymbol{H} = \boldsymbol{D} \cdot \mathcal{C}(\boldsymbol{a}, \boldsymbol{b})^T$ has rank $\min(m, \text{rank}(\boldsymbol{D}))$. When $m \leq n$ and sufficiently many c_i are nonzero, $\boldsymbol{H}$ has full column rank m. □

This linear independence property is essential for our dynamic threshold encryption scheme, where we need to ensure that different threshold combinations produce linearly independent decryption shares.

2.2 Key-Value Commitments: Using Membership Testing

Our first application of partial fractions is Key-Value Commitments (KVCs). A KVC scheme [1, 6] allows a prover to (succinctly) commit to a dictionary—which can be represented as a map or equivalently, a set of key-value pairs where no key is repeated—such that the prover can generate (short) proofs of (non-)membership. We show that partial fractions can be used to efficiently and succinctly accumulate key-value pairs into a constant-sized commitment. Moreover, the construction is "doubly-efficient" as both the prover and verifier must do very little work.

We first explain the high-level idea without any cryptography. Let $\mathbb{F}$ be a finite field of prime order p. Let $D \coloneqq \{(k_i, v_i)\}_{i \in [n]}$ be a dictionary of size n where the key space $\mathcal{K} = \mathbb{F}$ and the value space $\mathcal{V} = \mathbb{F} \setminus \{0\}$.[10] We consider the following function in $\mathbb{F}$, which is a linear combination of simple rational functions with linear denominator or linear "partial fractions",

$$C_D(X) \coloneqq \sum_{i \in [n]} \frac{v_i}{X + k_i}$$

For a key-value pair (k, v), observe that if $(k, v) \in D$, then the product, $\left(C_D(X) - \frac{v}{X+k}\right)\left(\frac{1}{X+k}\right)$ can be expressed as a sum of linear rational fractions. This directly follows from applying Lemma 1 to each summand in the expansion of the product. That is, we can compute h_i for $i \in [n]$ such that,

$$\sum_{i \in [n]} \frac{h_i}{X + k_i} = \left(C_D(X) - \frac{v}{X + k}\right)\left(\frac{1}{X + k}\right) \tag{1}$$

In fact, the h_i have an efficiently computable closed form. Suppose $k = k_j$ for some $j \in [n]$. It is easy to verify that,

$$h_i = \frac{v_i}{k_j - k_i} \text{ for } i \neq j, \text{ and } h_j = -\sum_{i \in [n] \setminus \{j\}} h_i$$

If $(k, v') \in D$ for $v' \neq v$, then the expansion contains a non-zero multiple of a quadratic denominator rational function, $\frac{v' - v}{(X+k)^2}$ which cannot be written as a combination of linear

[10] We treat the default value of each key as 0 and assume a key is *not in* D if its value is 0. This is merely a technicality and can be easily removed by separately accumulating the keys only. Throughout this work, we continue with the simplification.

rational fractions. Thus, partial fractions provide a sound mechanism to test for key-value membership. We can also easily test for non-membership of key k as well by observing that the linear combination $\{h_i\}$ for the equation,

$$\sum_{i\in[n]} \frac{h_i}{X+k_i} = C_D(X)\left(\frac{1}{X+k}\right)$$

exists if and only if $(k,v) \notin D$ by Property 1. (Note that this is simply a membership test for $(k,0)$ as we set 0 to be the default value for each key.)

To realize this cryptographically, we leverage prime-order bilinear groups. Suppose we have an asymmetric bilinear group of prime order p, $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, where g_i is the generator for $\mathbb{G}_i$ and e is an efficient pairing, $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to G_T$.[11] We represent the partial fractions using elements in $\mathbb{G}_1$ and $\mathbb{G}_2$. Let $x \xleftarrow{\$} \mathbb{F}$ be a uniformly random field element. Then, we can represent the fraction $\frac{1}{X+k}$ by $\left[\frac{1}{x+k}\right]_i$ in $\mathbb{G}_i$, which is simply the partial fraction evaluated at x. Due to linearity, we can represent $C_D(X)$ in $\mathbb{G}_1$ by,

$$\sum_{i\in[n]} v_i\left(\left[\frac{1}{x+k_i}\right]_1\right) = \left[\sum_{i\in[n]} \frac{v_i}{x+k_i}\right]_1 = [C_D(x)]_1$$

Let $\mathsf{com} := [C_D(x)]_1$ be the commitment to D. Membership of pair (k,v) can now be checked using the following pairing equation,

$$e\left(\mathsf{com} - \left[\frac{v}{x+k}\right]_1, \left[\frac{1}{x+k}\right]_2\right) \stackrel{?}{=} e\left(\sum_{i\in[n]} h_i\left(\left[\frac{1}{x+k_i}\right]\right)_1, [1]_2\right)$$

where the h_i are the same as before. Informally, the security relies on the fact that if x is secret and $(k,v) \notin D$ then it is computationally infeasible to compute a linear combination of the group elements such that the pairing equation is satisfied except with negligible probability. We formally define security and prove our scheme secure in the *plain* model using two new q-type assumptions we denote as the "Computational Bilinear Square-Pole Diffie-Hellman" (q-SqPole) and the "Computational Bilinear Distinct Pole Diffie-Hellman" (q-DistPole) assumptions. We show that both assumptions hold in the Generic Bilinear Group Model (GBGM).

Our construction satisfies several desirable properties of both theoretical and practical interest. First, the commitment is *homomorphic*—one can obtain a commitment to the disjoint union of two dictionaries by simply adding the two corresponding commitments. If the dictionaries are not disjoint, then the duplicated keys have their values added together. We believe this could be a desirable property in several applications and draw connections to polynomial-based constructions in Section 2.4. We leave further exploration of this property to future work.

Practically, due to the homomorphism, it is extremely efficient to update the commitment to include a new key-value pair (k', v') by simply computing $\mathsf{com} + v'\left[\frac{1}{x+k'}\right]_1$. If key k' is already in the dictionary, then this corresponds to an additive update by value v'. In fact, existing (non-)membership proofs can also be updated homomorphically. This property

[11] We refer the reader to the preliminaries in Section 3 for more details on bilinear groups.

effectively makes the commitment *stateless* given the initial commitment and proof as updating the commitment and proofs only requires knowing the current state which need not be remembered for future updates. We remark that the initial proof generation *is* stateful but stateless thereafter.

Second, the commitment can be seen as an accumulation of *scaled* Boneh-Boyen (BB) signatures [4].[12] Recall that in BB signatures, the secret signing key is a uniformly random $x \xleftarrow{\$} \mathbb{F}$ and the public key is $[x]_2$. To sign message $m \in \mathbb{F}$, the signer generates signature $\sigma := \left[\frac{1}{x+m}\right]_1$, which can be verified by the pairing equation, $e(\sigma, [x]_2 + [m]_2) \stackrel{?}{=} e([1]_1, [1]_2)$. Observe that our commitment is a linear combination of BB signatures on keys k_i, $\sigma_i := \left[\frac{1}{x+k_i}\right]_1$ where the scalars are precisely the keys' corresponding values v_i in the dictionary. We use this observation to define a novel cryptographic primitive we call *Credential-based* Key-Value Commitments (CbKVC). In CbKVC, keys must first be registered by a party known as the Registrar before they can be added to the dictionary. For example, a key could be the identity of a user in the system that must first register itself with the Registrar who may verify the identity for example and subsequently provide a BB signature on it (in both groups). Such a primitive is useful in decentralized permissioned settings where not everyone may be permitted to participate. Additionally, our scheme supports *revoking* registered keys by publishing a constant-sized commitment with which any party may check if a key is revoked before updating their proofs. Thus, we can also interpret our scheme as instantiating Key-Value Commitments over pre-existing BB signatures.

We formally define CbKVC, present our pairing-based construction, and prove security in the standard model using new q-type assumptions in Section 4.

2.3 Dynamic Threshold Encryption: Using Linear Independence

We apply our technique to construct a *Dynamic* Threshold Encryption scheme using the linear independence property (Lemma 2). In particular, we support encrypting a message to an arbitrary threshold t and authorized set S. Given S, we can (publicly) preprocess the authorized public keys (and public parameters) into a short, logarithmic-sized encryption key ek. The secret key and ciphertext are constant-sized. Our construction is concise and simple to understand—it does not employ any heavy machinery such as polynomial commitments or SNARKs and is entirely algebraic.

Idea. We first elaborate on the technique and then provide an overview of the construction. Suppose we had some linear combination of linear rational fractions. For example, let $n \geq 1$, and define $p_i(X) := \frac{1}{X+i}$ for $i \in [n]$, and consider the "uniform" combination (sum) $P(X) := \sum_{i \in I} p_i(X)$ for some $I \subset [n]$. Then, we can derive a $\mathbb{F}$-*linearly independent* combination of $\{p_i(X)\}$ with respect to the uniform combination. Let $j \notin I$. Using Lemma 2, we obtain,

$$L_j(X) := \left(P(X) + \sum_{i \in I} \frac{1}{j-i}\right) p_j(X) = \sum_{i \in I} \frac{1}{j-i} p_i(X)$$

Moreover, for distinct $j, j' \notin I$, L_j and $L_{j'}$ are linearly independent if $|I| \geq 2$. Thus, given any $|I| - 1$ distinct $j \notin I$, we obtain a full-rank $\mathbb{F}$-linear system over the p_i, by including

[12] Throughout this work, BB signatures refer to the signature scheme in [4] and not the signature scheme derived from [3].

the original $P(X)$. We employ this fact to build our threshold cryptosystem. Briefly, we will use $P(X)$ to *aggregate* public keys of users in a set I, which we describe in a moment—this will serve as the encryption key. The partial decryption share of party j will be the linearly independent combinations of the p_i with respect to p_j, which will represent party j's secret key.

Observe that for linear independence to hold, we need $j \notin I$. This suggests that we should use $I = \bar{S}$, the *complement* of S. Let s be the size of S. Then, our resultant linear system has $|I| = \left|\bar{S}\right| = n - s$ variables. Thus, we may only support a decryption threshold of $n - s - 1$. (Again, we get one linearly independent equation in the form of the encryption key, $P(X)$.)

To extend our approach to arbitrary $t \leq n$, we introduce "dummy" partial fractions. Specifically, we define $2n+1$ partial fractions of the form, $d_i(X) \coloneqq \frac{1}{X+n+i}$ for $i = 1, \ldots, 2n$, and $g(X) \coloneqq \frac{1}{X-1}$. We single out one of them as $g(X)$ as it serve a special purpose—we will use it to mask the message. Then, we add $s+t+1$ of them to the encryption key, and redefine $P(X) = \sum_{j \notin S} p_j(X) + \sum_{j \in [s+t]} d_j(X) + g(X)$, a sum over $(n-s) + (s+t) + 1 = n + t + 1$ partial fractions. (Note that $s + t \leq 2n$.)

Now, every party $j \in S$ with their "secret" $p_j(X)$ can derive a linearly independent combination over $n + t + 1$ fractions. For threshold decryption, we would like any t parties to be able to decrypt; however we cannot solve the system using only $t + 1$ linearly independent equations. (Recall we get one for "free" as the encryption key.) To address this, we introduce another set of n dummy partial fractions, $f_i(X) \coloneqq \frac{1}{X+3n+i}$ that can be used to derive another n linearly independent equations. In fact, anyone can derive these additional equations but by themselves they are not sufficient to decrypt as we need t more "secret" equations which are provided as decryption shares from parties with authorized keys.

Finally, we can solve the linear system with $n+t+1$ equations; in particular, we can solve for $g(X)$ with which the combiner can unmask the message and thus successfully decrypt. For security, we multiply the entire linear system by a random $r \in \mathbb{F}$. Thus, we solve for $r \cdot g(X)$. In our pairing-based construction below, the d_i will correspond to dummy "public" keys and f_i will correspond to dummy "secret" keys. Both d_i and f_i are however public and the distinction is merely to emphasize in which group the elements lie. We remark that generating dummy elements is a popular technique in threshold cryptography and is not unique to this work.

Construction. Briefly, the ciphertext will include the encryption key multiplied by a random mask where the random mask will also mask the message. The ciphertext is "ElGamal-like" lending itself to generic and efficient extensions for standard CCA-2 security. (We elaborate below.)

Given the ciphertext, the parties use their secret key (also a linear rational fraction) to generate a linearly independent combination of the masked encryption key, which they publish as a partial decryption share. The combining key will contain the n dummy partial fractions, f_i. Thus given the combining key, the combiner can derive n decryption shares by itself. Once t more decryption shares are obtained from other parties, the combiner can solve for a linear system on $n+t+1$ randomly scaled partial fractions and recover the mask used on the message.

Like before, we succinctly represent the linear fractions as group elements using the secret x. Suppose we have n parties indexed by $i \in [n]$. (In fact, such an indexing is not necessary and can be replaced by a hash function applied to arbitrary identities for example.) We set

party i's keys as follows.

$$\mathsf{pk}_i \coloneqq \left[\frac{1}{x+i}\right]_1 = [p_i(x)]_1, \text{ and } \mathsf{sk}_i \coloneqq [p_i(x)]_2$$

We set an additional $2n$ dummy public keys as follows.

$$\mathsf{dpk}_i \coloneqq \left[\frac{1}{x+n+i}\right]_1 = [d_i(x)]_1 \text{ for } i \in [2n]$$

Additionally, we set n more dummies as follows.

$$\mathsf{dsk}_i \coloneqq \left[\frac{1}{x+3n+i}\right]_2 = [f_i(x)]_2$$

While dsk_i are not secret, it is important to note that they are elements in $\mathbb{G}_2$ unlike the dummy public keys.

Lastly, we set one more dummy public key we denote as the *target* key,

$$\mathsf{tk} \coloneqq \left[\frac{1}{x-1}\right]_1 = [g(x)]_1$$

Let $S \subseteq [n]$ be a set of s parties we wish to encrypt to. Define the encryption key (with respect to S) as,

$$\mathsf{ek}_S \coloneqq \sum_{j\in[n]\setminus S} \mathsf{pk}_j + \sum_{j=1}^{s+t} \mathsf{dpk}_j + \mathsf{tk} = \sum_{j\in[n]\setminus S} [p_j(x)]_1 + \sum_{j=1}^{s+t} [d_j(x)]_1 + [g(x)]_1$$

And the combining key, a tuple of size n as,

$$\mathsf{ck} \coloneqq (\mathsf{dsk}_1, \ldots, \mathsf{dsk}_n)$$

Note that the combining key is independent of S.

To encrypt a message $m \in \mathbb{G}_T$ to S with decryption threshold t, the encrypting party samples $r \xleftarrow{\$} \mathbb{F}$ and generates the ciphertext,

$$\mathsf{ct} \coloneqq ([r]_1, r \cdot \mathsf{ek}_S, r \cdot e(\mathsf{tk}, [1]_2) + m)$$

To partially decrypt, party $i \in S$ computes,

$$\begin{aligned}\partial_i &\coloneqq e\left(\mathsf{ct}_2 + \left(\sum_{j\in[n]\setminus S} \frac{1}{j-i} + \sum_{j=1}^{s+t} \frac{1}{n+j-i} + \frac{1}{-1-i}\right)\mathsf{ct}_1, \mathsf{sk}_i\right) \\ &= r\left[\left(\left(\sum_{j\in[n]\setminus S} p_j(x) + \sum_{j=1}^{s+t} d_j(x) + g(x)\right) + \left(\sum_{j\in[n]\setminus S} \frac{1}{j-i} + \sum_{j=1}^{s+t} \frac{1}{n+j-i} + \frac{1}{-1-i}\right)\right) p_i(x)\right]_T\end{aligned}$$

Using Lemma 2, we obtain,

$$\begin{aligned}\partial_i &= r\left[\sum_{j\in[n]\setminus S} \frac{1}{j-i} p_j(x) + \sum_{j=1}^{s+t} \frac{1}{n+j-i} d_j(x) + \frac{1}{-1-i} g(x)\right]_T \\ &= r\left(\sum_{j\in[n]\setminus S} \frac{1}{j-i} [p_j(x)]_T + \sum_{j=1}^{s+t} \frac{1}{n+j-i} [d_j(x)]_T + \frac{1}{-1-i} [g(x)]_T\right)\end{aligned}$$

Given partial decryptions from a subset of t parties, $T \subseteq S$[13] , $\{\partial_{T_i}\}_{i\in[t]}$, the ciphertext ct, and the combining key $\mathsf{ck} = (\mathsf{dsk}_i)_i$, the combining party first computes for $i \in [n]$,

$$F_i \coloneqq e\left(\mathsf{ct}_2 + \left(\sum_{j\in[n]\setminus S} \frac{1}{j-(3n+i)} + \sum_{j=1}^{s+t} \frac{1}{(n+j)-(3n+i)} + \frac{1}{-1-(3n+i)}\right)\mathsf{ct}_1, \mathsf{dsk}_i\right)$$
$$= r\left(\sum_{j\in[n]\setminus S} \frac{1}{j-(3n+i)}[p_j(x)]_T + \sum_{j=1}^{s+t} \frac{1}{(n+j)-(3n+i)}[d_j(x)]_T + \frac{1}{-1-(3n+i)}[g(x)]_T\right)$$

Let $\bar{S} \coloneqq [n] \setminus S$ and $k \coloneqq \left|\bar{S}\right| = n - s$. We write $\bar{S} = (s'_1, \ldots, s'_k)$. Define a matrix $H_{n,S,T} = (h_{ij})_{i,j} \in \mathbb{F}^{(n+t+1)\times(n+t+1)}$ where,

$$H_{n,S,T} = \begin{bmatrix} 1 & \cdots & 1 & 1 & \cdots & 1 & 1 \\ \frac{1}{s'_1-T_1} & \cdots & \frac{1}{s'_k-T_1} & \frac{1}{n+1-T_1} & \cdots & \frac{1}{(n+s+t)-T_1} & \frac{1}{-1-T_1} \\ \vdots & & & \vdots & & & \vdots \\ \frac{1}{s'_1-T_t} & \cdots & \frac{1}{\bar{S}'_k-T_t} & \frac{1}{n+1-T_t} & \cdots & \frac{1}{(n+s+t)-T_t} & \frac{1}{-1-T_t} \\ \frac{1}{s'_1-(3n+1)} & \cdots & \frac{1}{s'_k-(3n+1)} & \frac{1}{n+1-(3n+1)} & \cdots & \frac{1}{(n+s+t)-(3n+1)} & \frac{1}{-1-(3n+1)} \\ \vdots & & & \vdots & & & \vdots \\ \frac{1}{s'_1-4n} & \cdots & \frac{1}{s'_k-4n} & \frac{1}{n+1-4n} & \cdots & \frac{1}{(n+s+t)-4n} & \frac{1}{-1-4n} \end{bmatrix}$$

Observe that $H_{n,S,T}$ is the concatenation of $\mathbf{1} \in \mathbb{F}^{1\times(n+t+1)}$, the row of all 1's, with a $(n+t)\times(n+t+1)$ Cauchy matrix with vectors $\boldsymbol{a} = -(T_1 \ldots, T_t, 3n+1, \ldots, 4n)$ and $\boldsymbol{b} = -(s'_1, \ldots, s'_k, n+1, \ldots, n+s+t, -1)$. We call $H_{n,S,T}$ a *special* Cauchy matrix. $H_{n,S,T}$ correspond exactly to the linear combinations we obtain from each of ct_2, ∂_{T_i}, and F_i.

Informally, since each a_i and b_j are distinct, the Cauchy matrix has rank $n+t$. Furthermore, since $\mathbf{1}$ is linearly independent, $H_{n,S,T}$ is full-rank and thus, invertible. We formally define Cauchy matrices and prove $H_{n,S,T}$ is invertible in the preliminaries (Section 3). Thus, the combiner can compute R as below, and finally decrypt the message as $m = \mathsf{ct}_3 - R$.

$$R \coloneqq \begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}^\top H^{-1}_{n,S,T} \begin{bmatrix} e(\mathsf{ct}_2, [1]_2) \\ \partial_{T_1} \\ \vdots \\ \partial_{T_t} \\ F_1 \\ \vdots \\ F_n \end{bmatrix} = \begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}^\top r \begin{bmatrix} [g(x)]_T \\ [p_{s'_1}(x)]_T \\ \vdots \\ [p_{s'_k}(x)]_T \\ [d_1(x)]_T \\ \vdots \\ [d_{s+t}(x)]_T \end{bmatrix} = r[g(x)]_T = r \cdot e(\mathsf{tk}, [1]_2)$$

The above construction highlights the novelty of Cauchy matrices and partial fractions. We make several optimizations to the scheme above to obtain a concretely efficient dynamic threshold encryption scheme. In particular, we remove the dependence on S in partial decryption, reduce the number of additional pairings the combiner needs to make to only three (from n), add *robust* correctness by adding a verification procedure for partial decryption

[13] If $|T| > t$, pick the first t decryption shares in lexicographical order.

shares, and add support for preprocessing the encryption key succinctly with respect to S. In the process, we reduce the computation and bandwidth required to generate and send partial decryption shares by sending elements in the source groups, $\mathbb{G}_1$ and $\mathbb{G}_2$, instead of the target group, $\mathbb{G}_T$. We describe our final construction in Section 5.2 and prove security using in the standard model using a new q-type assumption we present in Section 5.3.

Remark 1. Our construction can be thought of as encrypting *against* an *unauthorized* set. That is, instead of specifying the set of authorized parties, we can specify parties that are disallowed from partially decrypting. In this case, we would use the unauthorized set as-is during encryption (and not take its complement). If a new party joins, it will be able to contribute a partial decryption share with respect to an existing ciphertext as it was not in the unauthorized set during encryption time.

In our full construction, we take the complement of S with respect to only the set of all parties that have joined. Unlike existing constructions, our construction is slightly more favorable for scenarios where the encryptor might want to blacklist recipients out of a large set, rather than choose a small set of recipients.

2.4 Relation to Logarithmic Derivatives

The connection between partial fractions and logarithmic derivatives provides important context for understanding our approach and its relationship to existing work. This well-established mathematical relationship reveals that taking the derivative of the logarithm of a product of linear terms naturally yields a sum of partial fractions:

$$\frac{d}{dX} \log \prod_{i=1}^{n} (X + a_i)^{v_i} = \sum_{i=1}^{n} \frac{v_i}{X + a_i}$$

This fundamental connection has been observed and used in various cryptographic contexts, including the work of Haböck [22]. More specifically, this relationship also intuitively connects our key-value commitment scheme to the approach of [1, 33], who employ a power product formulation in their constructions.

The choice between summation and product formulations has interesting implications for cryptographic applications. The summation formulation that we adopt enables straightforward union operations in bilinear map groups, which would be challenging to achieve directly with the product formulation. However, this trade-off comes with limitations: while the product construction supports intersection computations through random linear combinations, such operations are not easily achievable in our summation-based approach.

3 Preliminaries

We denote the security parameter as λ; the set of natural numbers as $\mathbb{N}$; the set of integers as $\mathbb{Z}$; the set of integers modulo a prime p as $\mathbb{Z}_p$; and the prime p order finite field as $\mathbb{F}_p$, which we may write simply as $\mathbb{F}$. We assume that $p = \Omega(2^\lambda)$. For any $n \in \mathbb{N}$, let $[n] \coloneqq \{1, \ldots, n\}$. Denote a negligible function in λ (i.e., $o(\lambda^{-c}$ for all $c \in \mathbb{N})$ as $\mathsf{negl}(\lambda)$.

Bilinear Groups. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be cyclic (additive) groups of prime order p that admit an efficient (non-degenerate) bilinear pairing map, $e \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let g_1, g_2 be generators of groups $\mathbb{G}_1, \mathbb{G}_2$ respectively. We call $\mathcal{G} \coloneqq (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ a prime-order bilinear group

We assume there exists an efficient, randomized algorithm $\mathsf{BGrpGen}(1^\lambda)$ that can output descriptions of $\mathcal{G}$. We denote group elements using bracket notation [14]: for $x, y, z \in \mathbb{Z}_p$, let $[x]_1$ denote g_1^x, let $[y]_2$ denote g_2^y, and $[z]_T$ denote $e(g_1, g_2)^z$.

Vectors and Dictionaries. We denote vectors using boldface e.g. $\boldsymbol{a}, \boldsymbol{b}, \ldots$ We may explicitly list the components of the vector as, $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ for $\boldsymbol{a} \in \mathbb{F}^n$. We denote dictionaries as a set of pairs, $D = \{(k_i, v_i)\}_{i\in[n]}$ and denote $D[k_i]$ as the second component of the pair in which k_i is the first component. So $D[k_i] = v_i$.

Cauchy Matrices. We define cauchy matrices and highlight their relevant properties.

Definition 1 (Cauchy matrix). *Let $m, n \in \mathbb{Z}_{\geq 1}$. Let $\boldsymbol{a} \in \mathbb{F}^m, \boldsymbol{b} \in \mathbb{F}^n$ such that each element is distinct, i.e. the a_i's and b_j's are together all mutually unequal. Define $\mathcal{C}(\boldsymbol{a}, \boldsymbol{b}) = (c_{ij})_{i\in[m],j\in[n]} \in \mathbb{F}^{m\times n}$ as,*

$$c_{ij} = \frac{1}{a_i - b_j}$$

We denote $\mathcal{C}(\boldsymbol{a}, \boldsymbol{b})$ as the Cauchy matrix for vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. $\mathcal{C}(\boldsymbol{a}, \boldsymbol{b})$ is square if $m = n$.

Square Cauchy matrices have non-zero determinant and are therefore, invertible.

Lemma 3. *For any distinct $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}^m$, $\mathcal{C}(\boldsymbol{a}, \boldsymbol{b})$ is invertible.*

In fact, the inverse has a closed form that can be computed in $O(m^2)$ time using barycentric interpolation [2]. The closed form is given in the lemma below.

Lemma 4. *For any distinct $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}^m$, the inverse of $\mathcal{C}(\boldsymbol{a}, \boldsymbol{b})$ denoted by $\mathcal{C}^{-1}(\boldsymbol{a}, \boldsymbol{b}) = (c'_{ij})_{i,j\in[m]}$ is given by,*

$$c'_{ij} = \frac{\prod_{k\in[m]}(a_k - b_i)}{\prod_{\substack{k\in[m]\\k\neq j}}(a_j - a_k)} \prod_{\substack{k\in[m]\\k\neq i}} \frac{(a_j - b_k)}{(b_k - b_i)}$$

In this work, we are only interested in computing a single row of the inverse. This can be done in time $\tilde{O}(m)$ also using barycentric interpolation. Proofs of Lemmas 3 and 4 can be found in [8, 27].

Special *Cauchy matrix.* In this work, we consider the concatenated matrix, $H \in \mathbb{F}^{m\times m}$,

$$H(\boldsymbol{a}, \boldsymbol{b}) = \begin{bmatrix} \mathbf{1}_{1\times m} \\ \mathcal{C}(\boldsymbol{a}, \boldsymbol{b}) \end{bmatrix}$$

for $\boldsymbol{a} \in \mathbb{F}^{m-1}, \boldsymbol{b} \in \mathbb{F}^m$. It can be shown that this is just a limiting Cauchy matrix, and hence invertible, and with a nice closed form for each entry as in Lemma 5.

Lemma 5. *The (i, j)-th entry c^*_{ij} $(i, j \in [m])$ of inverse of matrix $H(\boldsymbol{a}, \boldsymbol{b})$ is given by,*

$$c^*_{ij} = \begin{cases} \frac{\prod_{k\in[m-1]}(a_k - b_i)}{\prod_{k\in[m-1],\ k\neq j-1}(a_{j-1} - a_k)} \cdot \prod_{k\in[m],\ k\neq i} \frac{(a_{j-1} - b_k)}{(b_k - b_i)} & \text{for } j \neq 1 \\ \frac{\prod_{k\in[m-1]}(a_k - b_i)}{\prod_{k\in[m],\ k\neq i}(b_k - b_i)} & \text{for } j = 1 \end{cases}$$

See Appendix B for a proof.

4 Credential-based Key-Value Commitments

We formally introduce CbKVC schemes in Section 4.1. We present our algebraic construction in Section 4.2. We introduce our new assumptions and prove security in Section 4.3. We fix the key space $\mathcal{K} = \mathbb{F}$ and the value space $\mathcal{V} = \mathbb{F} \setminus \{0\}$. In general, we can set the key space to be arbitrary strings $\mathcal{K} = \{0,1\}^*$ and use a collision-resistant hash function to map it to $\mathbb{F}$.

4.1 Definitions

Definition 2 (Credential-based KVC (CbKVC) Scheme). *A CbKVC scheme is a tuple of algorithms,*
$(\mathsf{KeyGen}, \mathsf{Register}, \mathsf{Insert}, \mathsf{Update}, \mathsf{PfUpdate}, \mathsf{ProveNonMem}, \mathsf{Verify}, \mathsf{VerifyNonMem})$,

- $(\mathsf{vk}, \mathsf{rk}, \mathsf{com}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$*: on input the security parameter* λ*, outputs public verification key* vk*, secret registration key* rk*, and the initial commitment to the empty key-value map* com*.*
- $\mathsf{reg} \leftarrow \mathsf{Register}(\mathsf{rk}, k, \mathsf{aux})$*: on input registration key* rk*, key* k*, and auxiliary information* aux*, output registration information* reg*.*
- $(\mathsf{com}', \pi_{\text{mem}}, \mathsf{upd}) \leftarrow \mathsf{Insert}(\mathsf{reg}, \mathsf{com}, (k, v))$*: on input registration information* reg*, a commitment* com*, KV pair* (k, v)*, outputs (1) an updated commitment* com′*, (2) a membership proof of the new KV pair* π_{mem}*, and (3) update information* upd*.*
- $(\mathsf{com}', \mathsf{upd}) \leftarrow \mathsf{Update}(\mathsf{reg}, \mathsf{com}, (k, \delta))$*: on input registration information* reg*, a commitment* com *and key* k*, whose corresponding value is to be updated by an additive value* δ*, outputs the updated commitment* com′ *and update information* upd*.*
- $\pi' \leftarrow \mathsf{PfUpdate}(\mathsf{reg}, k, \pi, \mathsf{upd})$*: on input registration information* reg *for key* k*, (non-)membership proof* π*, and update information* upd*, outputs an updated proof* π'*.*
- $\pi_{\text{non-mem}} \leftarrow \mathsf{ProveNonMem}(\mathsf{reg}, \mathsf{com}, k)$*: on input registration information* reg *for key* k*, and a KV commitment* com*, outputs a non-membership proof for* k *with respect to* com*.*
- $0/1 \leftarrow \mathsf{VerifyMem}(\mathsf{vk}, \mathsf{com}, (k, v), \pi_{\text{mem}})$*: on input verification key* vk*, KV commitment* com*, KV pair* (k, v)*, and membership proof* π_{mem}*, outputs 0 or 1.*
- $0/1 \leftarrow \mathsf{VerifyNonMem}(\mathsf{vk}, \mathsf{com}, k, \pi_{\text{non-mem}})$*: on input verification key* vk*, KV commitment* com*, key* k*, and non-membership proof* $\pi_{\text{non-mem}}$*, outputs 0 or 1.*

Remark 2. In the above definition, we specify an algorithm to generate a proof of non-membership (ProveNonMem) but not its analogue for proofs of membership. This is because Insert outputs a membership proof and so no explicit algorithm is needed.

Similar to prior work, we define correctness using a game. We base our definition on [1] and modify it to include the nuances of registration. Informally, if the commitment and proofs are honestly generated then all proofs of (non-)membership should verify after every update.

Definition 3 (CbKVC Correctness). *Let* CbKVC *be a CbKVC scheme,* $\mathcal{A}$ *be a PPT adversary, and* CH *be the challenger. We define the following game,*

$\mathsf{Game}^{\mathsf{correct}}$*:*

1. CH *samples* $(\mathsf{vk}, \mathsf{rk}, \mathsf{com}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and sends* vk *and* com *to* $\mathcal{A}$*.* CH *also maintains internal state which consists of (1)* R*, a key-registration dictionary initialized to empty, (2)* D*, a KV dictionary initialized to empty; (3)* P*, a key-proof dictionary initialized to empty; (4)* rk*, the registration secret key; and (5)* com*, the initial commitment.*

2. *$\mathcal{A}$ issues queries of one of the following forms,*
   (a) $(\mathsf{Register}, k)$*:*
      – CH *computes registration information* $\mathsf{CbKVC.Register}(\mathsf{rp}, k, D) \to \mathsf{reg}$ *and sends* reg *to* $\mathcal{A}$*.*
      – *Update* $R \leftarrow R \cup \{(k, \mathsf{reg})\}$*, and* $P \leftarrow P \cup \{(k, \mathsf{CbKVC.ProveNonMem}(\mathsf{reg}, k)\}$*.*
   (b) $(\mathsf{Insert}, (\mathsf{reg}, k, v))$*:*
      – CH *checks if* $(k, \mathsf{reg}) \notin R$ *or* $(k, \cdot) \in D$*. If so, respond with* $\perp$*.*
      – *Else execute* $\mathsf{CbKVC.Insert}(\mathsf{reg}, \mathsf{com}, (k, v)) \to (\mathsf{com}', \pi_{\mathrm{mem}}, \mathsf{upd})$*.*
      – *Update the KV dictionary* $D \leftarrow D \cup \{(k, v)\}$*, commitment* $\mathsf{com} \leftarrow \mathsf{com}'$*, and key-proof dictionary* $P \leftarrow P \cup \{(k, \pi_{\mathrm{mem}})\} \setminus \{(k, \pi_{\text{non-mem}})\}$ *where* $\pi_{\text{non-mem}} \leftarrow \mathsf{CbKVC.ProveNonMem}(\mathsf{reg}, k)$*.*
   (c) $(\mathsf{Update}, (\mathsf{reg}, k, \delta))$*:*
      – CH *checks if* $(k, \mathsf{reg}) \notin R$ *or* $(k, \cdot) \notin D$*. If so, respond with* $\perp$*.*
      – *Else execute* $\mathsf{CbKVC.Update}(\mathsf{com}, (k, \delta)) \to (\mathsf{com}', \pi_{\mathrm{mem}}, \mathsf{upd})$*.*
      – *Let* v *be such that* $(k, v) \in D$*. Update commitment* $\mathsf{com} \leftarrow \mathsf{com}'$*, KV dictionary,* $(D \cup (k, v + \delta)) \setminus \{(k, v)\}$*,*
3. *After every* Insert *and* Update *query,* CH *performs the following additional operations,*
   (a) *For each* $(k, \pi) \in P$*,* CH *executes* $\mathsf{CbKVC.PfUpdate}(R[k], k, \pi, \mathsf{upd}) \to \pi'$ *and updates* $(P \leftarrow P \cup \{(k, \pi')\}) \setminus \{(k, \pi)\}$*.*
   (b) *For each* $(k, \cdot) \in D$*,* CH *executes* $\mathsf{VerifyMem}(\mathsf{vk}, \mathsf{com}, (k, D[k]), P[k]) \to b$*. If any* $b = 0$*, then* CH *outputs* fail *and terminates.*
   (c) *For each* $(k, \cdot) \in P$ *but* $(k, \cdot) \notin D$*,* CH *executes* $\mathsf{VerifyNonMem}(\mathsf{vk}, \mathsf{com}, k, P[k]) \to b$*. If any* $b = 0$*, then* CH *outputs* fail *and terminates.*
4. CH *outputs* $\mathsf{success}$*.*

CbKVC *is* correct *if for every adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{correct}}_{\mathsf{CbKVC},\mathcal{A}}(\lambda) \coloneqq \Pr\left[\mathsf{CH} \text{ outputs } \mathsf{fail} \text{ in } \mathsf{Game}^{\mathsf{correct}}\right] = 0$$

We capture the "security" of our scheme using a modification of key-binding—we incorporate registration and capture adversaries that attempt to prove membership of *unregistered* keys. Informally, an adversary should be unable to generate valid proofs for differing values of registered keys (the usual notion of key-binding), and furthermore cannot generate valid (non-)membership proofs for unregistered keys. We call this property *Registration Robust Key-binding*, which is relevant for permissioned settings.

Definition 4 (CbKVC Registration Robust Key-binding). *Let* CbKVC *be a CbKVC scheme,* $\mathcal{A}$ *be a PPT adversary, and* CH *be the challenger. We define the following game,*

$\mathsf{Game}^{\mathsf{robust\text{-}bind}}$*:*

1. CH *samples* $(\mathsf{vk}, \mathsf{rk}, \mathsf{com}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and sends* vk *and* com *to* $\mathcal{A}$*.* CH *also maintains internal state which consists of (1)* R*, a key-registration dictionary initialized to empty, (2)* D*, a KV dictionary initialized to empty; (3)* rk*, the registration key; and (4)* com*, the initial commitment.*
2. *$\mathcal{A}$ issues queries of one of the following forms,*
   (a) $(\mathsf{Register}, k)$*:*
      – CH *computes registration information* $\mathsf{CbKVC.Register}(\mathsf{rk}, k, D) \to \mathsf{reg}$ *and sends* reg *to* $\mathcal{A}$*.*

– *Update* $R \leftarrow R \cup \{(k, \mathsf{reg})\}$.

(b) $(\mathsf{Insert}, (\mathsf{reg}, k, v))$*:*
– CH *checks if* $(k, \mathsf{reg}) \notin R$ *or* $(k, \cdot) \in D$*. If so, respond with* $\perp$*.*
– *Else execute* $\mathsf{CbKVC.Insert}(\mathsf{reg}, \mathsf{com}, (k, v)) \to (\mathsf{com}', \cdot, \cdot)$*.*
– *Update the KV dictionary* $D \leftarrow D \cup \{(k, v)\}$*, and commitment* $\mathsf{com} \leftarrow \mathsf{com}'$*.*

(c) $(\mathsf{Update}, (\mathsf{reg}, k, \delta))$*:*
– CH *checks if* $(k, \mathsf{reg}) \notin R$ *or* $(k, \cdot) \notin D$*. If so, respond with* $\perp$*.*
– *Else execute* $\mathsf{CbKVC.Update}(\mathsf{com}, (k, \delta)) \to (\mathsf{com}', \cdot, \cdot)$*.*
– *Let* v *be such that* $(k, v) \in D$*. Update commitment* $\mathsf{com} \leftarrow \mathsf{com}'$*, and dictionary* $(D \cup (k, v + \delta)) \setminus \{(k, v)\}$*,*

3. $\mathcal{A}$ *sends a final output to* CH *of one of the following forms,*
(a) a key k *such that* $(k, \cdot) \notin R$*, and a* non-membership *proof* π*,*
(b) a key k *such that* $(k, \cdot) \notin R$*, a value* v*, and a proof* π*,*
(c) a key k *such that* $(k, \cdot) \in R$ *but* $(k, \cdot) \notin D$*, a value* v*, and a proof* π*,*
(d) a key k *such that* $(k, \cdot) \in R$*,* $(k, \cdot) \in D$*, and a* non-membership *proof* π*,*
(e) a key k *such that* $(k, \cdot) \in D$*, a pair of values* v, v' *where* $v \neq v'$*, and a pair of corresponding proofs* π, π'*.*

4. CH *performs the following checks corresponding to* $\mathcal{A}$*'s output,*
(a) $\mathsf{CbKVC.VerifyNonMem}(\mathsf{vk}, k, \pi) \stackrel{?}{=} 1$*,*
(b) $\mathsf{CbKVC.VerifyMem}(\mathsf{vk}, (k, v), \pi) \stackrel{?}{=} 1$*,*
(c) $\mathsf{CbKVC.VerifyMem}(\mathsf{vk}, (k, v), \pi) \stackrel{?}{=} 1$*,*
(d) $\mathsf{CbKVC.VerifyNonMem}(\mathsf{vk}, k, \pi) \stackrel{?}{=} 1$*, or*
(e) $\mathsf{CbKVC.VerifyMem}(\mathsf{vk}, (k, v), \pi) \stackrel{?}{=} 1$ *and* $\mathsf{CbKVC.VerifyMem}(\mathsf{vk}, (k, v'), \pi') \stackrel{?}{=} 1$

If the check passes, CH *outputs* fail*, else* $\mathsf{success}$*.*

CbKVC *is* registration robust key-binding *if for every adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A}}(\lambda) \coloneqq \Pr\left[\mathsf{CH} \text{ outputs } \mathsf{fail} \text{ in } \mathsf{Game}^{\mathsf{robust\text{-}bind}}\right] = \mathsf{negl}(\lambda)$$

Remark 3. We assume the commitment is *honestly generated* in the definition above, i.e., the commitment is consistent with the final dictionary after $\mathcal{A}$'s insertions and updates. It would be interesting to explore whether this is necessary. We leave this to future work.

4.2 Construction

We construct our algebraic, pairing-based CbKVC scheme in Figure 1.

$\mathsf{KeyGen}(1^\lambda)$*:*
1. Sample a bilinear group, $\mathcal{G} \coloneqq (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$.
2. Sample $x \xleftarrow{\$} \mathbb{Z}_p$.
3. Output, $(\mathsf{vk} \coloneqq ([x]_1, [x]_2), \mathsf{rk} \coloneqq x, \mathsf{com} \coloneqq [1]_1)$.

$\mathsf{Register}(\mathsf{rk}, k, \mathsf{aux})$*:*
1. Parse $\mathsf{aux} \to \{(k_i, v_i)\}_{i \in I}$ where I is an indexing set.
2. Let $z_i \coloneqq \dfrac{v_i}{k - k_i}$ for $i \in I$.
3. Let $h^{(1)} \coloneqq \left[\dfrac{1}{\mathsf{rk} + k}\right]_1$, $h^{(2)} \coloneqq \left[\dfrac{1}{\mathsf{rk} + k}\right]_2$, and $h^{(1)}_i \coloneqq \left[\dfrac{1}{\mathsf{rk} + k_i}\right]_1$ for $i \in I$.

4. Let $\Lambda \coloneqq \sum_{i \in I} z_i \left(h_i^{(1)} - h^{(1)}\right)$.
5. Output $\mathsf{reg} \coloneqq \left(h^{(1)}, h^{(2)}, \Lambda\right)$.

$\mathsf{Insert}(\mathsf{reg}, \mathsf{com}, (k, v))$:
1. Parse $\mathsf{reg} =: \left(h^{(1)}, h^{(2)}, \Lambda\right)$.
2. Set $\mathsf{com}' \leftarrow \mathsf{com} + v \cdot h^{(1)}$.
3. Let $\pi_{\text{mem}} \coloneqq \left(h^{(1)}, h^{(2)}, \Lambda\right)$.
4. Let $\mathsf{upd} \coloneqq \left(k, v, h^{(1)}\right)$.
5. Output $(\mathsf{com}', \pi_{\text{mem}}, \mathsf{upd})$.

$\mathsf{Update}(\mathsf{reg}, \mathsf{com}, (k, \delta))$:
1. Parse $\mathsf{reg} =: \left(h^{(1)}, \cdot, \cdot\right)$.
2. Set $\mathsf{com}' \leftarrow \mathsf{com} + \delta \cdot h^{(1)}$.
3. Let $\mathsf{upd} \coloneqq (k, \delta, h^{(1)})$.
4. Output upd.

$\mathsf{PfUpdate}(\mathsf{reg}, k, \pi, \mathsf{upd})$:
1. Parse $\mathsf{reg} =: (h, \cdot)$.
2. Parse $\mathsf{upd} =: (k', v', h')$.
3. Parse $\pi =: \left(\Lambda, h^{(1)}, h^{(2)}\right)$.
4. If $k = k'$, let $\Lambda_{\mathsf{upd}} \leftarrow \Lambda$.
5. Else set,
$$\Lambda_{\mathsf{upd}} \leftarrow \Lambda + \frac{v'}{k' - k}\left(h^{(1)} - h'\right).$$
6. Output $\pi_{\mathsf{upd}} \coloneqq \left(\Lambda_{\mathsf{upd}}, h^{(1)}, h^{(2)}\right)$.

$\mathsf{ProveNonMem}(\mathsf{reg}, k)$:
1. Parse $\mathsf{reg} =: \left(\cdot, h^{(2)}, \Lambda\right)$.
2. Output $\pi_{\text{non-mem}} \coloneqq \left(h^{(2)}, \Lambda\right)$.

$\mathsf{Verify}(\mathsf{vk}, \mathsf{com}, (k, v), \pi_{\text{mem}})$:
1. Parse $\mathsf{vk} =: \left(x^{(1)}, x^{(2)}\right)$.
2. Parse $\pi_{\text{mem}} =: \left(h^{(1)}, h^{(2)}, \Lambda\right)$.
3. Check
$e\left(h^{(1)}, x^{(2)} + [k]_2\right) \stackrel{?}{=} e([1]_1, [1]_2)$, and
$e\left(x^{(1)} + [k]_1, h^{(2)}\right) \stackrel{?}{=} e([1]_1, [1]_2)$.
If any unequal, output 0.
4. Output $e\left(\mathsf{com} - v \cdot h^{(1)}, h^{(2)}\right) \stackrel{?}{=} e(\Lambda, [1]_2)$.

$\mathsf{VerifyNonMem}(\mathsf{vk}, \mathsf{com}, k, \pi_{\text{non-mem}})$:
1. Let $j \in I$ be such that $k = k_j$.
2. Parse $\pi_{\text{non-mem}} =: \left(h^{(2)}, \Lambda\right)$.
3. Check
$e\left(x^{(1)} + [k]_1, h^{(2)}\right) \stackrel{?}{=} e([1]_1, [1]_2)$.
If unequal, output 0.
4. Output $e\left(\mathsf{com}, h^{(2)}\right) \stackrel{?}{=} e(\Lambda, [1]_2)$.

Fig. 1: CbKVC: Credential-based Key-Value Commitment construction

Correctness. It is easy to verify by inspection that our CbKVC construction is correct, i.e., if the commitments, proofs and update information are honestly generated then the updated proofs of (non-membership) are valid.

Efficiency. The construction above is concretely efficient. The commitment, keys, update information and registration information are all constant-sized. The verification key is only two group elements, $|\mathbb{G}_1| + |\mathbb{G}_2|$ bits. The commitment is only $|\mathbb{G}_1|$ bits.[14] Each party must store $2|\mathbb{G}_1| + |\mathbb{G}_2|$ bits of registration information. Membership proofs are also $2|\mathbb{G}_1| + |\mathbb{G}_2|$ bits while non-membership proofs are slightly smaller with $|\mathbb{G}_1| + |\mathbb{G}_2|$ bits. Lastly the update information is $2|\mathbb{F}| + |\mathbb{G}_1|$ bits.

4.3 Security of CbKVC

To prove our CbKVC construction satisfies registration robust key-binding, we first introduce two new q-type assumptions we call q-Computational-Bilinear-Square-Pole-Diffie-Hellman (Assumption 1) and q-Computational-Bilinear-Distinct-Pole-Diffie-Hellman (Assumption 2). Informally, Assumption 1 says given linear partial fractions in $\mathbb{G}_1$ and $\mathbb{G}_2$, it is hard to compute a rational function with *square* denominator, or in other words a rational

[14] Adding a separate commitment to just the keys increase this cost to a total of $2|\mathbb{G}_1|$ bits.

function with a square pole, in $\mathbb{G}_1$. Our second assumption is reminiscent of the existential unforgeability of BB signatures. Essentially, given "BB-like" signatures in both $\mathbb{G}_1$ and $\mathbb{G}_2$, it is hard to generate a signature on a new "message" given only the verification key. This new message is in fact the pole of the linear partial fraction.
We formally present the assumptions below and establish their generic hardness in Appendix A.

Assumption 1 (Computational Bilinear Square-Pole Diffie-Hellman)**.** Let $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ be a bilinear group. We define the q-Computational-Bilinear-Square-Pole-Diffie-Hellman (or q-SqPole for short) problem using a game between the challenger CH and adversary $\mathcal{A}$,

1. CH samples $x \xleftarrow{\$} \mathbb{Z}_p$ and sends $[x]_1, [x]_2$ to $\mathcal{A}$.
2. For $i = 1, ..., q$, $\mathcal{A}$ sends $\rho_i \in \mathbb{Z}_p$ and CH responds with $\left[\frac{1}{x+\rho_i}\right]_1, \left[\frac{1}{x+\rho_i}\right]_2$.
3. $\mathcal{A}$ outputs $u \in \mathbb{G}_1$.

Let,

$$\mathsf{Adv}^{\mathsf{SqPole}}_{q,\mathcal{A}} := \Pr\left[u = \left[\frac{1}{(x+\rho_j)^2}\right]_1 \text{ for any } j \in [q] \;\middle|\; \begin{array}{c}\mathcal{G} \xleftarrow{\$} \mathsf{BGrpGen}(1^\lambda),\\ x \xleftarrow{\$} \mathbb{Z}_p\end{array}\right]$$

We say that the q-SqPole assumption holds in $\mathcal{G}$ if for every PPT algorithm $\mathcal{A}$, and for every polynomially bounded $q\colon \mathbb{Z} \to \mathbb{Z}$,

$$\mathsf{Adv}^{\mathsf{SqPole}}_{q(\lambda),\mathcal{A}(\lambda)} = \mathsf{negl}(\lambda)$$

Assumption 2 (Computational Bilinear Distinct Pole Diffie-Hellman)**.** Let $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ be a bilinear group. We define q-Computational-Bilinear-Distinct-Pole-Diffie-Hellman (or q-DistPole for short) problem using a game between the challenger CH and adversary $\mathcal{A}$,

1. CH samples $x \xleftarrow{\$} \mathbb{Z}_p$ and sends $[x]_1, [x]_2$ to $\mathcal{A}$.
2. For $i = 1, ..., q$, $\mathcal{A}$ sends $\rho_i \in \mathbb{Z}_p$ and CH responds with $\left[\frac{1}{x+\rho_i}\right]_1, \left[\frac{1}{x+\rho_i}\right]_2$.
3. $\mathcal{A}$ outputs $\rho \in \mathbb{Z}_p$ and $u \in \mathbb{G}_2$.

Let,

$$\mathsf{Adv}^{\mathsf{DistPole}}_{q,\mathcal{A}} := \Pr\left[u = \left[\frac{1}{(x+\rho)}\right]_2 \text{ and } \rho \notin \{\rho_i\}_{i\in[q]} \;\middle|\; \begin{array}{c}\mathcal{G} \xleftarrow{\$} \mathsf{BGrpGen}(1^\lambda),\\ x \xleftarrow{\$} \mathbb{Z}_p\end{array}\right]$$

We say that the q-DistPole assumption holds in $\mathcal{G}$ if for every PPT algorithm $\mathcal{A}$, and for every polynomially bounded $q\colon \mathbb{Z} \to \mathbb{Z}$,

$$\mathsf{Adv}^{\mathsf{DistPole}}_{q(\lambda),\mathcal{A}(\lambda)} = \mathsf{negl}(\lambda)$$

We are now ready to state the lemma below.

Lemma 6. *The* CbKVC *construction in Section 4.2 is registration robust key-binding assuming* q'-SqPole *(Assumption 1) and* q-DistPole *(Assumption 2) hold for some* q', q *that are polynomials in the security parameter.*

The proof is quite straightforward and follows almost directly from the two assumptions using an intermediate game hybrid to handle the case of unregistered keys. We provide the formal proof below.

Proof. We show that for every PPT adversary $\mathcal{A}$, its advantage, $\mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A}} = \mathsf{negl}(\lambda)$ through a sequence of game hybrids.

Let $\mathcal{A}$ be a PPT adversary. For every Game_i below, denote by $\mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A},i}$ $\mathcal{A}$'s advantage, the probability that CH outputs fail, in Game_i. Let $\mathsf{Game}_0 \coloneqq \mathsf{Game}^{\mathsf{robust-bind}}$, the registration robust key-binding game in Definition 4. Define Game_1 to be the same as Game_0 except CH does *not* output fail if part (a) or (b) hold in Step 4 of Game_0.

First, we show that $\mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A},0} \leq \mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A},1} + \mathsf{Adv}^{\mathsf{DistPole}}_{q,\mathcal{B}}$ for an adversary $\mathcal{B}$ we construct below and some polynomial q that bounds the number of $\mathsf{Register}$ queries $\mathcal{A}$ makes in Game_0.

We partition the event of CH outputting fail in Game_0 into two events, (1) $\mathcal{A}$ responds with form (a) or (b) in Step 3 and checks (a) or (b) hold in Step 4 (respectively) of Definition 4; and (2) $\mathcal{A}$ responds with form (c) – (e) in Step 3 and checks (c) – (e) hold in Step 4 (respectively). Then, $\mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A},0}$ is equal to the probability of event (1) + the probability of event (2).

First, we analyze event (1). We construct adversary $\mathcal{B}$ for q-$\mathsf{DistPole}$ that interacts with $\mathcal{A}$.

1. $\mathcal{B}$ receives $([x]_1, [x]_2)$ from the q-$\mathsf{DistPole}$ challenger $\mathsf{CH}^{\mathsf{DistPole}}$. Send $\mathsf{vk} \coloneqq ([x]_1, [x]_2)$ and $\mathsf{com} \coloneqq [1]_1$ to $\mathcal{A}$. Initialize an empty KV dictionary, $D \leftarrow \{\}$ and registration dictionary, $R \leftarrow \{\}$.
2. $\mathcal{A}$ makes a series of registration, insertion and updation queries according to Definition 3.
   (a) For every registration query $(\mathsf{Register}, k)$, $\mathcal{B}$ sends k to $\mathsf{CH}^{\mathsf{DistPole}}$ and receives $h^{(1)}_k \in \mathbb{G}_1$ and $h^{(2)}_k \in \mathbb{G}_2$. Run $\mathsf{CbKVC.Register}$ as in the construction using $h^{(1)} = h^{(1)}_k$, $h^{(2)} = h^{(2)}_k$, $\mathsf{aux} = D$ and $h^{(1)}_i = R[k_i][1]$, the first entry of the value of the key in R. Output computed reg to $\mathcal{A}$. Update $R \leftarrow R \cup \{(k, \mathsf{reg})\}$.
   (b) For insertions and updates, $\mathcal{B}$ updates D and com accordingly. Note that $\mathcal{B}$ has all the information necessary to update the commitment.
3. Let $D = \{(k_i, v_i)\}_{i \in I}$, $R = \{(k_j, \mathsf{reg}_j)\}_{j \in J}$ be the final state of the dictionary and registration information respectively. Let com be D's corresponding final commitment.
4. If $\mathcal{A}$ aborts at any time, $\mathcal{B}$ also aborts. Else $\mathcal{A}$, conditioned on form (a) or (b), responds with a tuple of one of the following forms,
   (a) (k_j, π) where $j \notin J$—$\mathcal{A}$ did not register k_j.
   (b) (k_j, v, π) where $j \notin J$—$\mathcal{A}$ did not register k_j;

   We handle each case separately. In each case, parse $\mathsf{vk} =: ([x]_1, [x]_2)$.
   (a) $\mathsf{VerifyNonMem}(\mathsf{vk}, \mathsf{com}, k_j, \pi) = 1$. By construction, this implies that $e\big(x^{(1)} + [k_j]_1, h^{(2)}\big) = e([1]_1, [1]_2)$ where $\pi =: (h^{(2)}, \cdot)$. By the pairing equation, we conclude $h^{(2)} = \left[\frac{1}{x+k_j}\right]_2$. Set $\rho \coloneqq k_j$ and $u \coloneqq h^{(2)}$.
   (b) $\mathsf{VerifyMem}(\mathsf{vk}, \mathsf{com}, (k_j, v), \pi) = 1$. By construction, this implies that $e\big(x^{(1)} + [k_j]_1, h^{(2)}\big) = e([1]_1, [1]_2)$ where $\pi =: (\cdot, h^{(2)}, \cdot)$. By the pairing equation, we conclude $h^{(2)} = \left[\frac{1}{x+k_j}\right]_2$. Set $\rho \coloneqq k_j$ and $u \coloneqq h^{(2)}$.
5. $\mathcal{B}$ sends ρ and u to $\mathsf{CH}^{\mathsf{DistPole}}$.

Since $j \notin J$, $\mathcal{B}$ never queries $\mathsf{CH}^{\mathsf{DistPole}}$ with k_j. Therefore, the probability, $\Pr[\mathcal{B} \text{ outputs } \rho, u] = \mathsf{Adv}^{\mathsf{DistPole}}_{q,\mathcal{B}}$. Therefore, the probability of event (1) $= \mathsf{Adv}^{\mathsf{DistPole}}_{q,\mathcal{B}} \cdot \Pr[\mathcal{A} \text{ outputs form (a) or (b)}] \leq \mathsf{Adv}^{\mathsf{DistPole}}_{q,\mathcal{B}}$. Observe that probability of event (2) is exactly the probability that CH outputs fail in Game_1. Thus, we conclude, $\mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A},0} \leq \mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A},1} + \mathsf{Adv}^{\mathsf{DistPole}}_{q,\mathcal{B}}$.

Next, we show that $\mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A},1} \leq \mathsf{Adv}^{\mathsf{SqPole}}_{q',\mathcal{B}'}$ for an adversary $\mathcal{B}'$ we construct below and some polynomial q' that bounds the number of $\mathsf{Register}$ queries $\mathcal{A}$ makes in Game_1.

We construct adversary $\mathcal{B}'$ for q'-SqPole that interacts with $\mathcal{A}$.

1. $\mathcal{B}'$ receives $([x]_1, [x]_2)$ from the q-SqPole challenger $\mathsf{CH}^{\mathsf{SqPole}}$. Send $\mathsf{vk} := ([x]_1, [x]_2)$ and $\mathsf{com} := [1]_1$ to $\mathcal{A}$. Initialize an empty KV dictionary, $D \leftarrow \{\}$ and registration dictionary, $R \leftarrow \{\}$.
2. $\mathcal{A}$ makes a series of registration, insertion and updation queries according to Definition 3.
   (a) For every registration query $(\mathsf{Register}, k)$, $\mathcal{B}'$ sends k to $\mathsf{CH}^{\mathsf{SqPole}}$ and receives $h_k^{(1)} \in \mathbb{G}_1$ and $h_k^{(2)} \in \mathbb{G}_2$. Run $\mathsf{CbKVC.Register}$ as in the construction using $h^{(1)} = h_k^{(1)}$, $h^{(2)} = h_k^{(2)}$, $\mathsf{aux} = D$ and $h_i^{(1)} = R[k_i][1]$, the first entry of the value of the key in R. Output computed reg to $\mathcal{A}$. Update $R \leftarrow R \cup \{(k, \mathsf{reg})\}$.
   (b) For insertions and updates, $\mathcal{B}'$ updates D and com accordingly. Note that $\mathcal{B}'$ has all the information necessary to update the commitment.
3. Let $D = \{(k_i, v_i)\}_{i \in I}$, $R = \{(k_j, \mathsf{reg}_j)\}_{j \in J}$ be the final state of the dictionary and registration information respectively. Let com be D's corresponding final commitment.
4. If $\mathcal{A}$ aborts at any time, $\mathcal{B}'$ also aborts. Else $\mathcal{A}$, conditioned on form (c) – (e), responds with a tuple of one of the following forms,
   (c) (k_j, v, π) where $j \notin J$—$\mathcal{A}$ did not register k_j;
   (d) (k_i, v, π) where $i \notin I$—$\mathcal{A}$ did not insert k_i;
   (e) (k_i, v, v', π, π') where $i \in I$ and $v \neq v'$—$\mathcal{A}$ opened existing k_i to two different values;

   We handle each case separately. In each case, parse $\mathsf{vk} =: ([x]_1, [x]_2)$.

   (c) $\mathsf{VerifyMem}(\mathsf{vk}, \mathsf{com}, (k_j, v), \pi) = 1$ and $j \notin I$. Parse $\pi =: \left(h^{(1)}, h^{(2)}, \Lambda\right)$. By construction, this implies that (1) $e\left(\mathsf{com} - v \cdot h^{(1)}, h^{(2)}\right) = e(\Lambda, [1]_2)$, (2) $e\left(h^{(1)}, x^{(2)} + [k]_2\right) = e([1]_1, [1]_2)$, and (3) $e\left(x^{(1)} + [k]_1, h^{(2)}\right) = e([1]_1, [1]_2)$. From the pairing equations in (2) and (3), we obtain $h^{(i)} = \left[\frac{1}{x+k_j}\right]_i$. Let w be the DLOG of Λ, i.e., $\Lambda =: [w]_1$. Since commitment com is honestly generated, we have $\left(\sum_{i \in I} \frac{v_i}{x+k_i} - \frac{v}{x+k_j}\right)\left(\frac{1}{x+k_j}\right) = w \cdot 1$. Expanding, we obtain, $\frac{v}{(x+k_j)^2} = \left(\sum_{i \in I} \frac{v_i}{x+k_i}\right)\left(\frac{1}{x+k_j}\right) - w$.

   Let $z_i := \frac{v_i}{k_j - k_i}$ for $i \in I$. Then,

$$\sum_{i \in I} z_i \left(\frac{1}{x+k_i} - \frac{1}{x+k_j}\right) = \left(\sum_{i \in I} \frac{v_i}{x+k_i}\right)\left(\frac{1}{x+k_j}\right)$$

   Let $h_i^{(1)} = R[k_i][1]$, the first entry of the registration information of key k_i. Then, we obtain $\sum_{i \in I} z_i \left(h_i^{(1)} - h^{(1)}\right) - \Lambda = \left[\frac{v}{(x+k_j)^2}\right]_1$.

   Set $u := \frac{1}{v}\left(\sum_{i \in I} z_i \left(h_i^{(1)} - h^{(1)}\right) - \Lambda\right)$.

   (d) $\mathsf{VerifyNonMem}(\mathsf{vk}, \mathsf{com}, k_j, \pi) = 1$ and $j \in I$. Parse $\pi =: \left(h^{(2)}, \Lambda\right)$. By construction, this implies (1) $e\left(\mathsf{com}, h^{(2)}\right) = e(\Lambda, [1]_2)$, and (2) $e\left(x^{(1)} + [k_j]_1, h^{(2)}\right) = e([1]_1, [1]_2)$. By pairing equation (2), we conclude $h^{(2)} = \left[\frac{1}{x+k_j}\right]_1$.

Since com is honestly generated, we have $\left(\sum_{i\in I\setminus\{j\}} \frac{v_i}{x+k_i} + \frac{v_j}{x+k_j}\right)\left(\frac{1}{x+k_j}\right) = w$. Expanding, we obtain, $\frac{v_j}{(x+\rho_j)^2} = w - \left(\sum_{i\in I} \frac{v_i}{x+k_i}\right)\left(\frac{1}{x+k_j}\right)$. Let $z_i := \frac{v_i}{k_j - k_i}$ for $i \in I \setminus \{j\}$. Then,

$$\sum_{i\in I\setminus\{j\}} z_i\left(\frac{1}{x+k_i} - \frac{1}{x+k_j}\right) = \left(\sum_{i\in I\setminus\{j\}} \frac{v_i}{x+k_i}\right)\left(\frac{1}{x+k_j}\right)$$

Let $h_i^{(1)} = R[k_i][1]$, the first entry of the registration information of key k_i. Set $u := \frac{1}{v_j}\left(\pi - \sum_{i\in I\setminus\{j\}} z_i\left(h_i^{(1)} - h^{(2)}\right)\right)$. Observe that $u = \left[\frac{1}{(x+k_j)^2}\right]_1$.

(e) $\mathsf{VerifyMem}(\mathsf{vk}, \mathsf{com}, (k_j, v), \pi) = 1$ and $\mathsf{VerifyMem}(\mathsf{vk}, \mathsf{com}, (k_j, v'), \pi') = 1$. Parse $\pi =: \left(h^{(1)}, h^{(2)}, \Lambda\right)$, and $\pi' =: \left(h^{(1)'}, h^{(2)'}, \Lambda'\right)$. By the verification pairing equations like in (c) above, we conclude $h^{(i)} = h^{(i)'} = \left[\frac{1}{x+k_j}\right]_i$.
this implies, $e\left(\mathsf{com} - v\cdot h^{(1)}, h^{(2)}\right) = e(\Lambda, [1]_2)$, and $e\left(\mathsf{com} - v'\cdot h^{(1)'}, h^{(2)'}\right) = e\left(\mathsf{com} - v'\cdot h^{(1)}, h^{(2)}\right) = e(\Lambda', [1]_2)$. By bilinearity of e, we obtain $e\left((v'-v)\cdot h^{(1)}, h^{(2)}\right) = e(\Lambda - \Lambda', [1]_2)$. Set $u := \frac{1}{v'-v}(\Lambda - \Lambda')$. Observe that $u = \left[\frac{1}{(x+k_j)^2}\right]_1$ by the same reasoning in (c).

5. $\mathcal{B}'$ responds to $\mathsf{CH}^{\mathsf{SqPole}}$ with u.

In each case $u = \left[\frac{1}{(x+k_j)^2}\right]_1$ and thus, $\Pr[\mathcal{B}' \text{ outputs } u] = \mathsf{Adv}^{\mathsf{SqPole}}_{q',\mathcal{B}'}$. Therefore, $\mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A},1} = \mathsf{Adv}^{\mathsf{SqPole}}_{q',\mathcal{B}'} \cdot \Pr[\mathcal{A} \text{ outputs form (c) – (e)}] \le \mathsf{Adv}^{\mathsf{SqPole}}_{q',\mathcal{B}'}$.
Putting it all together, $\mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A}} = \mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A},0} \le \mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A},1} + \mathsf{Adv}^{\mathsf{PoleUF}}_{q,\mathcal{B}} \le \mathsf{Adv}^{\mathsf{PoleUF}}_{q,\mathcal{B}} + \mathsf{Adv}^{\mathsf{SqPole}}_{q',\mathcal{B}'}$. Since q-PoleUF and q-SqPole hold, we conclude $\mathsf{Adv}^{\mathsf{robust\text{-}bind}}_{\mathsf{CbKVC},\mathcal{A}} = \mathsf{negl}(\lambda)$. □

5 Dynamic Threshold Encryption

We consider Dynamic Threshold Encryption with *preprocessing* in which given the authorized set of partial decryptors, it is possible to publicly *preprocess* the public parameters and authorized public keys to form a succinct encryption key ek. We formally introduce definitions in Section 5.1, our final construction in Section 5.2, and introduce our new assumption using which we prove security in Section 5.3.

5.1 Definitions

Definition 5 (Dynamic Threshold Encryption With Preprocessing). *A dynamic threshold encryption scheme with preprocessing is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{Join}, \mathsf{Preprocess}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Verify}, \mathsf{Combine})$,

1. $\mathsf{Setup}(1^\lambda, n) \xrightarrow{\$} (\mathsf{pp}, \mathsf{msk}, \mathsf{vk}, \mathsf{dk}, \mathsf{ck})$. *On input the security parameter* 1^λ*, and an upper bound on the number of parties* n*, outputs public parameters* pp*, master secret key* msk*, public verification key* vk*, public decryption key* dk*, and public combining key* ck.
2. $\mathsf{Join}(\mathsf{msk}, \mathsf{id}) \to (\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}, \mathsf{vk}_{\mathsf{id}})$. *On input the master secret key* msk *and identity* id *of a new user, outputs a tuple of user's keys:* $\mathsf{sk}_{\mathsf{id}}$ *is the user's secret key,* $\mathsf{pk}_{\mathsf{id}}$ *is the public key, and* $\mathsf{vk}_{\mathsf{id}}$ *is the public verification key.*
3. $\mathsf{Preprocess}(\mathsf{pp}, S, K, \{\mathsf{pk}_{\mathsf{id}_i}\}_{i\in K}) \to \mathsf{ek}$. *On input public parameters* pp*, an "authorized" set of identities* $S = \{\mathsf{id}_i\}_i$*, the set of all joined identities* K*, and public keys corresponding to* K*,* $\{\mathsf{pk}_{\mathsf{id}_i}\}_{i\in K}$*, outputs an encryption key* ek.
4. $\mathsf{Encrypt}(\mathsf{ek}, m, t) \xrightarrow{\$} \mathsf{ct}$. *On input encryption key* ek *(for some authorized set* S*), message* m*, and threshold* t*, outputs a corresponding ciphertext* ct.
5. $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{id}, \mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) \to \partial_{\mathsf{id}}$. *On input the decryption key* dk*, user identity* id*, corresponding secret key* $\mathsf{sk}_{\mathsf{id}}$*, and ciphertext* ct*, outputs a partial decryption share* ∂_{id}.
6. $\mathsf{Verify}(\mathsf{vk}, \mathsf{id}, \mathsf{vk}_{\mathsf{id}}, \mathsf{ct}, \partial) \to 0/1$. *On input the verification key* vk*, user identity* id*, corresponding verification key* $\mathsf{vk}_{\mathsf{id}}$*, ciphertext* ct*, and partial decryption share* ∂*, outputs 0 or 1.*
7. $\mathsf{Combine}(\mathsf{ck}, S, t, \mathsf{ct}, T, D) \to m$. *On input the combining key* ck*, authorized set* S*, threshold* t*, ciphertext* ct*, a subset of authorized users* $T \subseteq S$ *of size* $t' \geq t$*, and set of* t' *decryption shares* $D = \{\partial_i\}_{i\in[t']}$*, outputs a plaintext message* m.

Correctness. We define correctness the usual way using two notions. First is the notion of *partial decryption correctness* which, informally, states that for a correctly generated partial decryption share, the algorithm Verify outputs 1; and second is the notion of *(robust) correctness* which, informally, states that if sufficient verified partial decryption shares are combined using $\mathsf{Combine}$, then the combiner successfully decrypts the message. We formalize these below.

Definition 6 (Partial Decryption Correctness). *For all* $\lambda \in \mathbb{N}, n \leq 2^\lambda$*, all* $\mathsf{pp}, \mathsf{msk}, \mathsf{vk}, \mathsf{dk}, \mathsf{ck}$ *in the support of* $\mathsf{Setup}(1^\lambda, n)$*, if* $(\mathsf{sk}_{\mathsf{id}_i}, \mathsf{pk}_{\mathsf{id}_i}, \mathsf{vk}_{\mathsf{id}_i}) \leftarrow \mathsf{Join}(\mathsf{msk}, \mathsf{id}_i)$*, then for any set of identities* K*, authorized set* $S \subseteq K$*, any threshold* $t \leq |S|$*, and all* $\mathsf{id}_j \in S$*, it holds that,*

$$\Pr\left[\mathsf{Verify}\big(\mathsf{vk}, \mathsf{id}_j, \mathsf{vk}_{\mathsf{id}_j}, \mathsf{ct}, \partial\big) = 1 \;\middle|\; \begin{array}{c} \mathsf{ek} \leftarrow \mathsf{Preprocess}\Big(\mathsf{pp}, S, K, \{\mathsf{pk}_{\mathsf{id}_i}\}_{i\in K}\Big), \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m, t), \\ \partial \leftarrow \mathsf{Decrypt}(\mathsf{dk}, \mathsf{id}_j, \mathsf{sk}_{\mathsf{id}_j}, \mathsf{ct}) \end{array}\right] = 1$$

Definition 7 ((Robust) Correctness). *For all* $\lambda \in \mathbb{N}, n \leq 2^\lambda$*, all* $\mathsf{pp}, \mathsf{msk}, \mathsf{vk}, \mathsf{dk}, \mathsf{ck}$ *in the support of* $\mathsf{Setup}(1^\lambda, n)$*, if* $(\mathsf{sk}_{\mathsf{id}_i}, \mathsf{pk}_{\mathsf{id}_i}, \mathsf{vk}_{\mathsf{id}_i}) \leftarrow \mathsf{Join}(\mathsf{msk}, \mathsf{id}_i)$*, then for any set of identities* K*, authorized set* $S \subseteq K$*, any threshold* $t \leq |S|$*, any subset* $T \subseteq S$ *of size at least* t*, it holds that,*

$$\Pr\left[\mathsf{Combine}(\mathsf{ck}, S, t, \mathsf{ct}, T, D) = m \;\middle|\; \begin{array}{c} \mathsf{ek} \leftarrow \mathsf{Preprocess}\Big(\mathsf{pp}, S, K, \{\mathsf{pk}_{\mathsf{id}_i}\}_{i\in K}\Big), \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m, t), \\ (\partial_{\mathsf{id}_j} \leftarrow \mathsf{Decrypt}(\mathsf{dk}, \mathsf{id}_j, \mathsf{sk}_{\mathsf{id}_j}, \mathsf{ct}) \text{ and} \\ \mathsf{Verify}\big(\mathsf{vk}, \mathsf{id}_j, \mathsf{vk}_{\mathsf{id}_j}, \mathsf{ct}, \partial\big) = 1) \text{ for } \mathsf{id}_j \in T, \\ D \coloneqq \{\partial_{\mathsf{id}_j}\}_{\mathsf{id}_j \in T} \end{array}\right] = 1$$

The above definition may be relaxed to include negligible correctness error, i.e., the probability is at least $1 - \mathsf{negl}(\lambda)$.

Efficient Preprocessing. We require the output of $\mathsf{Preprocess}$ to be *sublinear* in the size of n.

Security. We prove security against Chosen Plaintext Attack (CPA) adversaries that may *adaptively* corrupt parties. We formalize this below. In Appendix C we also extend our scheme to be (publicly-verifiable) CCA2 secure.

Definition 8 (CPA security with adaptive corruptions). *Let* DTE *be a Dynamic Threshold Encryption scheme with preprocessing. For all* $\lambda \in \mathbb{N}$, $n \leq 2^\lambda$, *we define the following game between a challenger* CH *and adversary* $\mathcal{A}$.

$\mathsf{Game}^{\mathsf{CPA}}$:

1. CH *runs* $\mathsf{DTE.Setup}(1^\lambda, n) \to (\mathsf{pp}, \mathsf{msk}, \mathsf{vk}, \mathsf{dk}, \mathsf{ck})$ *and sends* $(\mathsf{pp}, \mathsf{vk}, \mathsf{dk}, \mathsf{ck})$ *to* $\mathcal{A}$. CH *maintains internal state initialized as* $K \leftarrow \{\}$, *the set of all joined identities;* $J \leftarrow \{\}$, *a dictionary mapping joined identities to their keys; and* $C \leftarrow \{\}$, *a set of all corrupted identities.*
2. $\mathcal{A}$ *makes queries of the following form in any order,*
   (a) $(\mathsf{Join}, \mathsf{id})$. *If* id *has already joined, return* $\perp$. *Else run* $\mathsf{DTE.Join}(\mathsf{msk}, \mathsf{id}) \to (\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}, \mathsf{vk}_{\mathsf{id}})$. *Send* $(\mathsf{pk}_{\mathsf{id}}, \mathsf{vk}_{\mathsf{id}})$ *to* $\mathcal{A}$. *Update* $K \leftarrow K \cup \{\mathsf{id}\}$, *and* $J \leftarrow J \cup \{(\mathsf{id}, (\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}, \mathsf{vk}_{\mathsf{id}}))\}$.
   (b) $(\mathsf{Corrupt}, \mathsf{id})$. *If* $\mathsf{id} \notin K$, *return* $\perp$. *Else return* $\mathsf{sk}_{\mathsf{id}}$ *from* $J[\mathsf{id}]$ *and update* $C \leftarrow C \cup \{\mathsf{id}\}$.
3. $\mathcal{A}$ *responds with a set* S. *If* $S \not\subseteq K$, *return* $\perp$.
4. CH *runs* $\mathsf{DTE.Preprocess}(\mathsf{pp}, S, K, \{\mathsf{pk}_{\mathsf{id}_i}\}_{\mathsf{id}_i \in K}) \to \mathsf{ek}$ *where* $\mathsf{pk}_{\mathsf{id}_i}$ *is taken from* J. *Send* ek *to* $\mathcal{A}$.
5. $\mathcal{A}$ *continues to corrupt parties by making queries* $(\mathsf{Corrupt}, \mathsf{id})$ *as above.*
6. $\mathcal{A}$ *picks a threshold* $t \leq |S|$, *messages* $m_0, m_1 \in \mathbb{G}_T$ *and sends* (t, m_0, m_1) *to* CH.
7. CH *samples* $b \xleftarrow{\$} \{0,1\}$ *and encrypts* $\mathsf{DTE.Encrypt}(\mathsf{ek}, m_b, t) \xrightarrow{\$} \mathsf{ct}$. *Send* ct *to* $\mathcal{A}$.
8. $\mathcal{A}$ *continues to corrupt parties by making queries* $(\mathsf{Corrupt}, \mathsf{id})$ *as above.*
9. $\mathcal{A}$ *responds with bit* b'.
10. CH *outputs* fail *if* $b = b'$ *and* $|C| < t$. *Else output* $\mathsf{success}$.

A dynamic threshold encryption scheme DTE *is CPA-secure with adaptive corruptions if for every adversary* $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{DTE},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathsf{CH} \text{ outputs } \mathsf{fail} \text{ in } \mathsf{Game}^{\mathsf{CPA}}\right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda)$$

5.2 Construction

First, we reintroduce notation as used in the technical overview. Then, we proceed to elaborate on optimizations to make the construction concretely efficient. The final construction is given in Figure 2. For simplicity, we assume the identities are values in $[n]$. This can easily extend to arbitrary identities using a collision-resistant hash function to map the identities to (an exponentially large) subset of $\mathbb{Z}_p$. We reserve a linear-sized portion of $\mathbb{Z}_p$ for the dummies as they must be distinct from real user identities. (This restriction may also be relaxed for negligible correctness error.)

Partial Fractions. Consistent with the technical overview (Section 2), we define the following partial fractions,

$$p_i(X) = \frac{1}{X+i} \text{ for } i \in [n], \qquad d_i(X) = \frac{1}{X+n+i} \text{ for } i \in [2n],$$
$$f_i(X) = \frac{1}{X+3n+i} \text{ for } i \in [n], \quad g(x) = \frac{1}{X-1}$$

The p_i correspond to the user keys, d_i correspond to the dummy keys used for encryption, f_i correspond to the dummy keys used for decryption, and g corresponds to the "target" key used to mask the message during encryption.

Decryption matrix. We assume that the set of partial decryptions used to combine and recover the message is exactly t. Let K be the set of all joined identities and let $k := |K|$. For authorized set $S \subseteq K$ of size s, let $\bar{S} := K \setminus S$ and so, $\left|\bar{S}\right| = k - s$. We write $\bar{S} = \left(s'_1, \ldots, s'_{k-s}\right)$. Given a subset of authorized identities $T \subseteq S$ (and $|T| = t$), let $\boldsymbol{a} := -(T_1 \ldots, T_t, 3n+1, \ldots, 4n)$ and $\boldsymbol{b} := -(s'_1, \ldots, s'_{k-s}, n+1, \ldots, 2n-k+s+t, -1)$. Define,

$$H_{n,K,S,T} := \begin{bmatrix} \mathbf{1}_{1\times(n+t+1)} \\ \mathcal{C}(\boldsymbol{a}, \boldsymbol{b}) \end{bmatrix}$$

The combiner computes the first row of the inverse of $H_{n,K,S,T}$ denoted by $\boldsymbol{h} = (h_1, \ldots, h_{n+t+1}) \in \mathbb{F}^{1\times(n+t+1)}$. Recall that $H_{n,K,S,T}$ is invertible by Lemma 5.

Linear Combinations. We define the following linear combinations or simply sums that are relevant for decryption. For $i \in \mathbb{Z}$ and threshold t, define,

$$l_i^{K,S,t} := \left(\sum_{j \in K \setminus S} \frac{1}{j-i} + \sum_{j=1}^{n-k+s+t} \frac{1}{n+j-i} + \frac{1}{-1-i} \right)$$

We are now ready to elaborate on the optimizations.

Removing the dependence on S in partial decryption. In our high-level construction in Section 2, an authorized party i must know S to be able to generate their decryption share ∂_i. We show how to remove this requirement. Essentially, we outsource computing the linear combination to the combiner. So instead, party i computes and sends,

$$\partial'_i := (e(\mathsf{ct}_2, \mathsf{sk}_i), e(\mathsf{ct}_1, \mathsf{sk}_i))$$

By Lemma 2, the combiner can construct ∂_i as before using the knowledge of S as, $\partial_i = \partial'_{i,1} + l_i^{K,S,t} \partial'_{i,2}$ Therefore, as desired,

$$\partial_i = r \left(\sum_{j \in [n] \setminus S} \frac{1}{j-i} [p_j(x)]_T + \sum_{j=1}^{n-k+s+t} \frac{1}{n+j-i} [d_j(x)]_T + \frac{1}{-1-i} [g(x)]_T \right)$$

Adding share verification. We make one more modification to partial decryption to be able to support the "verification" of the partial decryption share. Verification becomes necessary to obtain *robust* correctness, which stated informally, says that if all shares are verified, then the combiner decrypts the correct message. (See Definition 7.) In contrast, standard correctness would assume the shares are honestly/correctly generated. Most prior work treat robust correctness as the standard definition of correctness and we follow the same.

To add share verification, we modify the scheme such that the partial decryptors don't construct the linear combinations themselves. Instead, party i samples a random mask r_i to mask their secret key, and use the inverse r_i^{-1} to mask the ciphertext. The new partial decryption share for party i is defined as,

$$\partial_i'' \coloneqq \left(\frac{1}{r_i}\mathsf{ct}_1, \frac{1}{r_i}\mathsf{ct}_2, [r_i]_2, r_i \cdot \mathsf{sk}_i\right)$$

First, we show that the combiner can still recover the necessary linear combination by performing the pairings itself. Indeed, $e\left(\partial_{i,1}'', \partial_{i,4}''\right) = e\left(\frac{1}{r_i}\mathsf{ct}_1, r_i \cdot \mathsf{sk}_i\right) = e(\mathsf{ct}_1, \mathsf{sk}_i) = \partial_{i,1}'$ (and similarly reconstruct $\partial_{i,2}'$). In fact, the combiner can save another pairing and directly compute, $\partial_i = e\left(\partial_{i,2}'' + l_i^{S,t}\partial_{i,1}'', \partial_{i,4}''\right) = e(\mathsf{ct}_2 + l_i^{S,t}\mathsf{ct}_1, \mathsf{sk}_i)$.

Moreover, this allows us to perform share verification by checking the following pairing equations,$e(\mathsf{pk}_i, \partial_{i,3}'') \stackrel{?}{=} e([1]_1, \partial_{i,4}'')$, $e(\partial_{i,1}'', \partial_{i,3}'') \stackrel{?}{=} e(\mathsf{ct}_1, [1]_2)$, and $e(\partial_{i,2}'', \partial_{i,3}'') \stackrel{?}{=} e(\mathsf{ct}_2, [1]_2)$. Here, pk_i acts as the verification key vk_i for party i. Not only do we get share verification, we save on both computation and bandwidth for the partial decrypting party. In particular, it no longer computes any pairings and instead of sending $2|\mathbb{G}_T|$ bits, now sends only $2(|\mathbb{G}_1| + |\mathbb{G}_2|)$ bits which is in practice an order of magnitude smaller. In the full construction, we batch the pairings using random linear combinations. As a result, the total number of pairings required reduces to only 3 at the cost of negligible correctness error.

Reducing combiner's work. Observe that to derive n more independent relations, the combiner must currently do n pairings. This can easily be reduced to only two pairings. First, we define F_i' similarly to ∂_i' above, i.e., $F_i' \coloneqq (e(\mathsf{ct}_2, \mathsf{dsk}_i), e(\mathsf{ct}_1, \mathsf{dsk}_i))$. (So far, we have increased the number of pairings to $2n$...) Then just like ∂_i, we obtain, $F_i = F_{i,1}' + l_{3n+i}^{K,S,t}F_{i,2}'$. Notice that for decryption, the combiner is only interested in the first row of $H_{n,S,t}^{-1}$. Therefore, we are interested in deriving,

$$R = h_1 \cdot e(\mathsf{ct}_2, [1]_2) + \sum_{i\in[t]} h_{i+1} \cdot \partial_{T_i} + \sum_{i\in[n]} h_{i+t+1}F_i$$

We can rewrite the last term as, $F \coloneqq \sum_{j\in[n]} h_{i+t+1}F_i = \sum_{j\in[n]} h_{i+t+1}F_{i,1}' + \sum_{i\in[n]} h_{i+t+1}l_{3n+i}F_{i,2}'$. By bilinearity of e, we can compute each sum using a single pairing $\sum_{i\in[n]} h_{1,i+t+1}F_{i,1}' = e\left(\mathsf{ct}_2, \sum_{i\in[n]} h_{i+t+1} \cdot \mathsf{dsk}_i\right)$ and, $\sum_{i\in[n]} h_{1,i+t+1}l_{3n+i}^{K,S,t}F_{i,2}' = e\left(\mathsf{ct}_1, \sum_{i\in[n]} h_{i+t+1} \cdot l_{3n+i}^{K,S,t}\mathsf{dsk}_i\right)$. Therefore the message mask is, $R = h_1 \cdot e(\mathsf{ct}_2, [1]_2) + \sum_{i\in[t]} h_{i+1} \cdot \partial_{T_i} + e\left(\mathsf{ct}_2, \sum_{i\in[n]} h_{i+t+1} \cdot \mathsf{dsk}_i\right) + e\left(\mathsf{ct}_1, \sum_{i\in[n]} h_{i+t+1} \cdot l_{3n+i}^{K,S,t}\mathsf{dsk}_i\right)$.

Shrinking public parameters and encryption key. To preprocess the encryption key ek, we require $2n$ dummy public keys which are part of the public parameters. We observe

that the identities of these public keys are irrelevant—we only need sufficiently many of them to correctly define ek. Therefore, we can use the optimization from the recent work by Waters and Wu [31] to obtain logarithmically-sized pp and ek. In particular, instead of publishing $2n$ of dpk_i, we can publish sums of dpk_i for increasing powers of 2. More formally, for $j \in [\lceil \log n \rceil]$, define intervals $I_j := \{2^{j-1}, \ldots, 2^j - 1\}$, $\mathsf{dpk}'_j := \sum_{i \in I_j} \mathsf{dpk}_i$, and $\mathsf{dpk}''_j = \sum_{i \in I_j} \mathsf{dpk}_{n+i}$. Publish dpk'_j and dpk''_j as part of pp instead of $2n$ dpk_i. Then, use dpk'_j for the first component of ek and dpk''_j for the second in $\mathsf{Preprocess}$ in the construction below.
For simplicity, we present our scheme with $O(n)$ public parameters as the optimization is generic in this case. We remark that this optimization does not readily work for [12] giving us a unique advantage.

Putting it all together. We present the final construction with all the optimizations above in Figure 1.

$\mathsf{Setup}(1^\lambda, n)$*:*
- Sample $x \xleftarrow{\$} \mathbb{Z}_p$.
- Set $\mathsf{msk} := x$.
- Let $\mathsf{dpk}_i := [d_i(x)]_1$ for $i \in [2n]$, $\mathsf{dsk}_i := [f_i(x)]_2$ for $i \in [n]$, $\mathsf{tk} := [g(x)]_1$.
- Set $\mathsf{pp} := (\mathsf{dpk}_1, \ldots, \mathsf{dpk}_{2n}, \mathsf{tk})$, $\mathsf{vk} := \bot$, $\mathsf{dk} := \bot$, $\mathsf{ck} := (\mathsf{dsk}_1, \ldots, \mathsf{dsk}_n)$.
- Output $(\mathsf{msk}, \mathsf{pp}, \mathsf{vk}, \mathsf{dk}, \mathsf{ck})$.

$\mathsf{Join}(\mathsf{msk}, i)$*:*
- Output $\left(\mathsf{vk}_i := \left[\frac{1}{\mathsf{msk}+i}\right]_1 = [p_i(\mathsf{msk})]_1, \mathsf{pk}_i := [p_i(\mathsf{msk})]_1, \mathsf{sk}_i = [p_i(\mathsf{msk})]_2\right)$. (Note that $\mathsf{vk}_i = \mathsf{pk}_i$.)

$\mathsf{Preprocess}(\mathsf{pp}, S, K, \{\mathsf{pk}_i\}_{i \in K})$*:*
- If $S \not\subseteq K$ (where K is the set of all joined identities), abort. Let $s = |S|, k' = |K \setminus S|$.
- Parse $\mathsf{pp} =: (\mathsf{dpk}_1, \ldots, \mathsf{dpk}_{2n}, \mathsf{tk})$. Let $s = |S|$.
- Output $\mathsf{ek} := \left(\sum_{j \in K \setminus S} \mathsf{pk}_j + \sum_{j=1}^{n-k'} \mathsf{dpk}_j + \mathsf{tk}, \left(\mathsf{dpk}_j\right)_{j=n-k'+1}^{n-k'+s}, e(\mathsf{tk}, [1]_2)\right)$

$\mathsf{Encrypt}(\mathsf{ek}, m, t)$*:*
- Parse $\mathsf{ek} =: \left(\mathsf{ek}_S, \left(\mathsf{dpk}_j\right)_{j=n-k'+1}^{n-k'+s}, \mathsf{tk}_T\right)$.
- Sample $r \xleftarrow{\$} \mathbb{Z}_p$.
- Output $\mathsf{ct} := \left([r]_1, r \cdot \left(\mathsf{ek}_S + \sum_{i=n-k'+1}^{n-k'+t} \mathsf{dpk}_j\right), r \cdot \mathsf{tk}_T + m\right)$

$\mathsf{Decrypt}(\mathsf{dk}, i, \mathsf{sk}_i, \mathsf{ct})$*:*
- Parse $\mathsf{ct} =: (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3)$.
- Sample $r_i \xleftarrow{\$} \mathbb{F}^*$ and output $\partial''_i := \left(\frac{1}{r_i}\mathsf{ct}_1, \frac{1}{r_i}\mathsf{ct}_2, [r_i]_2, r_i \cdot \mathsf{sk}_i\right)$.

$\mathsf{Verify}(\mathsf{vk}, i, \mathsf{vk}_i, \mathsf{ct}, \partial)$*:*
- Parse $\partial =: (\partial_1, \partial_2, \partial_3, \partial_4)$.
- Sample $r_1, r_2, r_3 \xleftarrow{\$} \mathbb{Z}_p$.
- Output $e(r_1\mathsf{vk}_i + r_2\partial_1 + r_3\partial_2, \partial_3) \stackrel{?}{=} e(r_1[1]_1, \partial_4) + e(r_2\mathsf{ct}_1 + r_3\mathsf{ct}_2, [1]_2)$.

$\mathsf{Combine}(\mathsf{ck}, S, t, \mathsf{ct}, T, D)$:
- Parse $\mathsf{ck} =: (\mathsf{dsk}_i)_{i\in[n]}$, $D =: \{\partial''_{T_i}\}_{i\in[t]}$.
- For each $i \in [t]$, compute, $\partial_{T_i} := e\Big(\partial_{T_i,2} + l_i^{K,S,t}\partial''_{T_i,1}, \partial''_{T_{i,4}}\Big)$
- Compute $R = h_1 \cdot e(\mathsf{ct}_2, [1]_2) + \sum_{i\in[t]} h_{i+1} \cdot \partial_{T_i} + e\Big(\mathsf{ct}_2, \sum_{i\in[n]} h_{i+t} \cdot \mathsf{dsk}_i\Big) + e\Big(\mathsf{ct}_1, \sum_{i\in[n]} h_{i+t+1} \cdot l_{3n+i}^{K,S,t}\mathsf{dsk}_i\Big)$
- Output $\mathsf{ct}_3 - R$.

Fig. 2: DTE: Dynamic Threshold Encryption with Preprocessing construction

Correctness. We show that our construction satisfies Partial Decryption and robust correctness. Indeed for any party i, honestly generated ciphertext ct and honestly generated partial decryption share $\partial = \left(\frac{1}{r_i} \cdot \mathsf{ct}_1, \frac{1}{r_i} \cdot \mathsf{ct}_2, [r_i]_2, r_i \cdot \mathsf{sk}_i\right)$ for $r_i \xleftarrow{\$} \mathbb{F}^*$, we have,

$$e(\mathsf{vk}_i, \partial_3) = e\left(\left[\frac{1}{x+i}\right]_1, [r_i]_2\right) = \left[\frac{r_i}{x+i}\right]_T = e([1]_1, r_i \cdot \mathsf{sk}_i) = e([1]_1, \partial_3),$$
$$e(\partial_1, \partial_3) = e\left(\frac{1}{r_i} \cdot \mathsf{ct}_1, [r_i]_2\right) = e(\mathsf{ct}_1, [1]_2), \text{ and}$$
$$e(\partial_2, \partial_3) = e\left(\frac{1}{r_i} \cdot \mathsf{ct}_2, [r_i]_2\right) = e(\mathsf{ct}_2, [1]_2)$$

Therefore the batched pairing equation is also satisfied and Verify outputs 1. Observe that if the partial decryption shares are correct, then the combiner solves a linear system with the correct equations, recovers $e(\mathsf{tk}, [1]_2)$ and thus, successfully unmasks the message. Robust correctness follows.

Efficiency. (n is an upper bound on the number of parties in the system.) The public parameters pp are $O(\log n)$ (after the optimization in [31]). pp is only used to preprocess ek which is also $O(\log n)$. Each party has an $O(1)$ public and secret key—a single group element in $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The public key also acts as the party's verification key. The combining key is exactly n $\mathbb{G}_2$ elements. The ciphertext is $O(1)$. In particular, it is only 2 $\mathbb{G}_1$ elements and 1 $\mathbb{G}_T$ element. It requires no pairings. Partial decryption shares also require no pairings; they are $O(1)$ and are concretely small with only 2 $\mathbb{G}_1$ and 2 $\mathbb{G}_2$ elements. The combiner must compute $t + 3$ pairings.

5.3 Security of DTE

We prove our construction secure in the standard model using a new q-type assumption we call *q-Rank-Deficient-Diffie-Hellman* (Assumption 3). Informally, the assumption says that given partial fractions in $\mathbb{G}_1$, a *deficit* amount in $\mathbb{G}_2$, and a linear relation over partial fractions in $\mathbb{G}_1$, one cannot recover a designated *target* partial fraction in $\mathbb{G}_T$. We formally present the assumption below and establish its generic hardness in Appendix A.

Assumption 3 (Decisional Bilinear Rank-Deficient Diffie-Hellman)**.** Let $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ be a bilinear group. We define the q-Decisional-Bilinear-Rank-Deficient-Diffie-Hellman (or q-RankDef for short) problem using a game between the challenger CH and adversary $\mathcal{A}$,

1. CH samples $x \xleftarrow{\$} \mathbb{Z}_p$. CH maintains internal state initialized as $S_1 \leftarrow \{\}, S_2 \leftarrow \{\}$.
2. $\mathcal{A}$ sends query $(\mathsf{target}, \rho_t \in \mathbb{Z}_p)$. CH responds with $\left[\frac{1}{x+\rho_t}\right]_1$ and updates $S_1 \leftarrow S_1 \cup \{\rho_t\}$
3. $\mathcal{A}$ sends queries (adaptively) of the form,
   (a) $(1, \rho \in \mathbb{Z}_p)$. CH responds with $\left[\frac{1}{x+\rho}\right]_1$ and updates $S_1 \leftarrow S_1 \cup \{\rho\}$.
   (b) $(2, \rho \in \mathbb{Z}_p)$. CH responds with $\left[\frac{1}{x+\rho}\right]_2$ and updates $S_2 \leftarrow S_2 \cup \{\rho\}$.
   (c) $(\mathsf{Combine}, S)$. (This query may only be sent once.) CH samples uniformly random $r, r' \xleftarrow{\$} \mathbb{Z}_p$ and a bit $b \xleftarrow{\$} \{0,1\}$. CH responds with a triple,
$$\left([r]_1, r\left(\sum_{\rho \in S}\left[\frac{1}{x+\rho}\right]_1 + \left[\frac{1}{x+\rho_t}\right]_1\right), [u]_T\right)$$
   where $u = \frac{r}{x+\rho_t}$ if $b = 0$ or $u = r'$ if $b = 1$.
4. $\mathcal{A}$ responds with bit $b' \in \{0,1\}$.

Let $q := |S_1|$ and let,

$$\mathsf{Adv}^{\mathsf{RankDef}}_{q,\mathcal{A}} := \left|\Pr\left[b' = b \text{ and } \rho_t \notin S_2 \cup S \text{ and } |S_2| < |S| \text{ and } S \subseteq S_1 \;\middle|\; \begin{array}{c}\mathcal{G} \xleftarrow{\$} \mathsf{BGrpGen}(1^\lambda), \\ x \xleftarrow{\$} \mathbb{Z}_p\end{array}\right] - \frac{1}{2}\right|$$

We say that the q-$\mathsf{RankDef}$ assumption holds in $\mathcal{G}$ if for every PPT algorithm $\mathcal{A}$, and for every polynomially bounded $q\colon \mathbb{Z} \to \mathbb{Z}$,

$$\mathsf{Adv}^{\mathsf{RankDef}}_{q(\lambda),\mathcal{A}(\lambda)} = \mathsf{negl}(\lambda)$$

Using Assumption 3, we first prove our construction DTE is CPA-secure with adaptive corruptions. Then, we lift it to be CCA2-secure using one-time simulation sound NIZKs (see Appendix C), while still maintaining succinctness of the ciphertext.

Lemma 7. *The* DTE *construction in Figure 2 is CPA-secure with adaptive corruptions assuming* q-$\mathsf{RankDef}$ *(Assumption 3) holds.*

Proof. Let $\mathcal{A}$ be a PPT adversary for q-$\mathsf{RankDef}$. We construct a PPT adversary $\mathcal{B}$ such that $\frac{1}{2}\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{DTE},\mathcal{A}} = \mathsf{Adv}^{\mathsf{RankDef}}_{q,\mathcal{B}}$. Let CH be the challenger for q-$\mathsf{RankDef}$.
$\mathcal{B}$ proceeds as follows.

1. Maintain internal state $K \leftarrow \{\}$. Furthermore, $\mathcal{B}$ stores everything it receives from CH.
2. Send query $(\mathsf{target}, -1)$ to CH. CH responds with tk.
3. For $i \in [2n]$, send query $(1, n+i)$ to CH. CH responds with dpk_i.
4. For $i \in [n]$, send query $(2, 3n+i)$ to CH. CH responds with dsk_i.
5. Send to $\mathcal{A}$ $\left(\mathsf{pp} := (\mathsf{dpk}_1, \ldots, \mathsf{dpk}_{2n}, \mathsf{tk}), \mathsf{vk} := \bot, \mathsf{dk} := \bot, \mathsf{ck} = (\mathsf{dsk}_i)_{i\in[n]}\right)$.
6. $\mathcal{A}$ makes queries of the following form,
   (a) (Join, i). $\mathcal{B}$ sends $(1, i)$ to CH and receives pk_i. Send pk_i to $\mathcal{A}$. Update $K \leftarrow K \cup \{i\}$.
   (b) $(\mathsf{Corrupt}, i)$. $\mathcal{B}$ sends $(2, i)$ to CH and receives sk_i. Send sk_i to $\mathcal{A}$.
7. $\mathcal{A}$ responds with a set S. Let $s := |S|$.
8. Let $k := |K|$. $\mathcal{B}$ constructs ek as below and sends to $\mathcal{A}$,

$$\mathsf{ek} := \left(\sum_{j \in K \setminus S} \mathsf{pk}_j + \sum_{j=1}^{n-k+s} \mathsf{dpk}_j + \mathsf{tk}, \left(\mathsf{dpk}_j\right)_{n-k+s+1}^{n-k+2s}, e(\mathsf{tk}, [1]_2)\right)$$

9. $\mathcal{A}$ continues to send $(\mathsf{Corrupt}, i)$ queries and $\mathcal{B}$ does as above.
10. $\mathcal{A}$ responds with threshold $t \leq s$ and two messages m_0, m_1.
11. $\mathcal{B}$ sends CH query $(\mathsf{Combine}, (K \setminus S) \cup \{n+1, \ldots, 2n-k+s+t\})$. $\mathcal{B}$ receives (h_1, h_2, h_3) from CH.
12. $\mathcal{A}$ continues to send $(\mathsf{Corrupt}, i)$ queries and $\mathcal{B}$ does as above.
13. $\mathcal{B}$ samples a bit $b \xleftarrow{\$} \{0,1\}$ and sends $(h_1, h_2, m_b + h_3)$ to $\mathcal{A}$.
14. $\mathcal{A}$ responds with bit b'.
    (a) If $b' = b$, then $\mathcal{B}$ responds to CH with 0.
    (b) Else respond with 0 or 1 with equal probability.

Observe that $(\mathsf{pp}, \mathsf{vk}, \mathsf{dk}, \mathsf{ck})$ is identically distributed to the output of $\mathsf{DTE.Setup}$, $\mathsf{pk}_i, \mathsf{sk}_i$ are identically distributed to the output of $\mathsf{DTE.Join}$, and ek is identically distributed to the output of $\mathsf{DTE.Preprocess}$. Therefore $\mathcal{B}$ up till step 11 behaves identically to $\mathsf{Game}^{\mathsf{CPA}}$ up till step 6. Denote by b_{CH}, the bit CH samples. Now if $b_{\mathsf{CH}} = 0$, then the output of $\mathcal{B}$ is identically distributed to $\mathsf{DTE.Encrypt}$. If the number of $\mathsf{Corrupt}$ queries that $\mathcal{A}$ sends is less than t, then the number of $(2, \cdot)$ queries to CH is less than $n + t + 1$. Observe that $|S| = n + t + 1$ above. So $|S_2| < |S|$ in the game in Assumption 3). Therefore, we have,

$$\begin{aligned}
\Pr[\mathsf{CH} \text{ outputs } \mathsf{fail}] &= \Pr[b_{\mathsf{CH}} = 0] \cdot \Pr[b' = b \mid b_{\mathsf{CH}} = 0] \\
&\quad + \Pr[b_{\mathsf{CH}} = 1] \cdot \Pr[b' = b \mid b_{\mathsf{CH}} = 1] \\
&= \frac{1}{2}\left(\frac{1}{2} \pm \mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{DTE},\mathcal{B}}\right) + \frac{1}{2} \cdot \frac{1}{2} = \pm\frac{1}{2}\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{DTE},\mathcal{B}} + \frac{1}{2} \\
\implies \mathsf{Adv}^{\mathsf{RankDef}}_{q,\mathcal{A}} &= \left|\Pr[\mathsf{CH} \text{ outputs } \mathsf{fail}] - \frac{1}{2}\right| = \frac{1}{2}\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{DTE},\mathcal{B}}
\end{aligned}$$

□

References

1. Agrawal, S., Raghuraman, S.: KVaC: Key-Value Commitments for blockchains and beyond. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020, Part III. Lecture Notes in Computer Science, vol. 12493, pp. 839–869. Springer, Cham, Switzerland, Daejeon, South Korea (Dec 7–11, 2020). https://doi.org/10.1007/978-3-030-64840-4_28
2. Berrut, J.P., Trefethen, L.N.: Barycentric lagrange interpolation. SIAM review **46**(3), 501–517 (2004)
3. Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 223–238. Springer Berlin Heidelberg, Germany, Interlaken, Switzerland (May 2–6, 2004). https://doi.org/10.1007/978-3-540-24676-3_14
4. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 56–73. Springer Berlin Heidelberg, Germany, Interlaken, Switzerland (May 2–6, 2004). https://doi.org/10.1007/978-3-540-24676-3_4
5. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. Lecture Notes in Computer Science, vol. 3494, pp. 440–456. Springer Berlin Heidelberg, Germany, Aarhus, Denmark (May 22–26, 2005). https://doi.org/10.1007/11426639_26
6. Boneh, D., Bünz, B., Fisch, B.: Batching techniques for accumulators with applications to IOPs and stateless blockchains. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology – CRYPTO 2019, Part I. Lecture Notes in Computer Science, vol. 11692, pp. 561–586. Springer,

Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26948-7_20
7. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) Advances in Cryptology – CRYPTO 2001. Lecture Notes in Computer Science, vol. 2139, pp. 213–229. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001). https://doi.org/10.1007/3-540-44647-8_13
8. Boros, T., Kailath, T., Olshevsky, V.: Pivoting and backward stability of fast algorithms for solving cauchy linear equations. Linear Algebra and its Applications **343-344**, 63–99 (03 2002). https://doi.org/10.1016/S0024-3795(01)00519-5
9. Boyen, X.: The uber-assumption family. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing-Based Cryptography – Pairing 2008. pp. 39–56. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
10. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 315–334 (2018). https://doi.org/10.1109/SP.2018.00020
11. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) Advances in Cryptology – EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 45–64. Springer Berlin Heidelberg, Germany, Amsterdam, The Netherlands (Apr 28 – May 2, 2002). https://doi.org/10.1007/3-540-46035-7_4
12. Delerablée, C., Pointcheval, D.: Dynamic threshold public-key encryption. In: Wagner, D. (ed.) Advances in Cryptology – CRYPTO 2008. Lecture Notes in Computer Science, vol. 5157, pp. 317–334. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2008). https://doi.org/10.1007/978-3-540-85174-5_18
13. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: Vaudenay, S. (ed.) PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography. Lecture Notes in Computer Science, vol. 3386, pp. 416–431. Springer Berlin Heidelberg, Germany, Les Diablerets, Switzerland (Jan 23–26, 2005). https://doi.org/10.1007/978-3-540-30580-4_28
14. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part II. Lecture Notes in Computer Science, vol. 8043, pp. 129–147. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013). https://doi.org/10.1007/978-3-642-40084-1_8
15. Fiore, D., Kolonelos, D., de Perthuis, P.: Cuckoo commitments: Registration-based encryption and key-value map commitments for large spaces. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023, Part V. Lecture Notes in Computer Science, vol. 14442, pp. 166–200. Springer, Singapore, Singapore, Guangzhou, China (Dec 4–8, 2023). https://doi.org/10.1007/978-981-99-8733-7_6
16. Garg, S., Hajiabadi, M., Mahmoody, M., Rahimi, A.: Registration-based encryption: Removing private-key generator from IBE. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018: 16th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 11239, pp. 689–718. Springer, Cham, Switzerland, Panaji, India (Nov 11–14, 2018). https://doi.org/10.1007/978-3-030-03807-6_25
17. Garg, S., Kolonelos, D., Policharla, G.V., Wang, M.: Threshold encryption with silent setup. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology – CRYPTO 2024, Part VII. Lecture Notes in Computer Science, vol. 14926, pp. 352–386. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2024). https://doi.org/10.1007/978-3-031-68394-7_12
18. Glaeser, N., Kolonelos, D., Malavolta, G., Rahimi, A.: Efficient registration-based encryption. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023: 30th Conference on Computer and Communications Security. pp. 1065–1079. ACM Press, Copenhagen, Denmark (Nov 26–30, 2023). https://doi.org/10.1145/3576915.3616596

19. Goyal, R., Vaikuntanathan, V.: Locally verifiable signature and key aggregation. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022, Part II. Lecture Notes in Computer Science, vol. 13508, pp. 761–791. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15979-4_26
20. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 305–326. Springer Berlin Heidelberg, Germany, Vienna, Austria (May 8–12, 2016). https://doi.org/10.1007/978-3-662-49896-5_11
21. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) Advances in Cryptology – EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 415–432. Springer Berlin Heidelberg, Germany, Istanbul, Turkey (Apr 13–17, 2008). https://doi.org/10.1007/978-3-540-78967-3_24
22. Haböck, U.: Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Paper 2022/1530 (2022), https://eprint.iacr.org/2022/1530
23. Jarecki, S., Liu, X.: Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In: Reingold, O. (ed.) TCC 2009: 6th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 5444, pp. 577–594. Springer Berlin Heidelberg, Germany (Mar 15–17, 2009). https://doi.org/10.1007/978-3-642-00457-5_34
24. Jutla, C.S., Roy, A.: Relatively-sound NIZKs and password-based key-exchange. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography. Lecture Notes in Computer Science, vol. 7293, pp. 485–503. Springer Berlin Heidelberg, Germany, Darmstadt, Germany (May 21–23, 2012). https://doi.org/10.1007/978-3-642-30057-8_29
25. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology – ASIACRYPT 2013, Part I. Lecture Notes in Computer Science, vol. 8269, pp. 1–20. Springer Berlin Heidelberg, Germany, Bengalore, India (Dec 1–5, 2013). https://doi.org/10.1007/978-3-642-42033-7_1
26. Kiltz, E., Wee, H.: Quasi-adaptive NIZK for linear subspaces revisited. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 101–128. Springer Berlin Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46803-6_4
27. Knuth, D.E.: The Art of Computer Programming, Volume 1: Fundamental Algorithms. Addison-Wesley, 3rd edn. (1997)
28. Mousavi, M., Esmaeili, M., Gulliver, T.A.: Circulant, circulant-like and orthogonal MDS generalized cauchy matrices. Adv. Math. Commun. **19**(2), 716–735 (2025)
29. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th Annual Symposium on Foundations of Computer Science. pp. 543–553. IEEE Computer Society Press, New York, NY, USA (Oct 17–19, 1999). https://doi.org/10.1109/SFFCS.1999.814628
30. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) Advances in Cryptology – EUROCRYPT'97. Lecture Notes in Computer Science, vol. 1233, pp. 256–266. Springer Berlin Heidelberg, Germany, Konstanz, Germany (May 11–15, 1997). https://doi.org/10.1007/3-540-69053-0_18
31. Waters, B., Wu, D.J.: Silent threshold cryptography from pairings: Expressive policies in the plain model. Cryptology ePrint Archive, Paper 2025/1547 (2025), https://eprint.iacr.org/2025/1547
32. Wee, H.: Déjà q: Encore! un petit ibe. In: Kushilevitz, E., Malkin, T. (eds.) Theory of Cryptography. pp. 237–258. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
33. Zhang, Y., Katz, J., Papamanthou, C.: An expressive (zero-knowledge) set accumulator. In: 2017 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 158–173 (2017). https://doi.org/10.1109/EuroSP.2017.35

A Generic Hardness of New Assumptions

We formally prove the generic hardness of the assumptions introduced in this work.

A.1 Generic Hardness of q-SqPole

Lemma 8. q-SqPole *(Assumption 1) holds in the GBGM.*

Proof. We assume $\mathcal{A}$ is a *generic* adversary, i.e., it may only perform generic operations over labels of group elements to obtain labels of the resultant group operation.
We use the standard technique of using the symbolic representation of group elements [30]. In particular, each group element is labeled using a rational function over a formal indeterminate. Let $[\cdot]_{\mathbb{G}_*}$ represent labels for elements in $\mathbb{G}_*$.
Let X be the formal variable for $[x]_{\mathbb{G}_i}$, the label of $[x]_i$ sent by the challenger CH to the generic adversary $\mathcal{A}$. For any query ρ_i for $i \in [q]$, CH responds with the correct labels, $[P_i(x)]_{\mathbb{G}_1}, [P_i(x)]_{\mathbb{G}_2}$ where $P_i(X) \in \mathbb{F}(X), P_i(X) \coloneqq \frac{1}{X+\rho_i}$.
Since $\mathcal{A}$ is generic, at any query i, it can construct, $[F^{(i)}_{a,b,\boldsymbol{c}}(x)]_{\mathbb{G}_i}$ for,

$$F^{(i)}_{a,b,\boldsymbol{c}}(X) \coloneqq aX + b + \sum_{j \le i} \frac{c_j}{X+\rho_j} \qquad \text{for any } a, b \in \mathbb{F}, \boldsymbol{c} \in \mathbb{F}^q$$

using generic group operations. This follows immediately from the fact that $\mathcal{A}$ is generic and thus, the only obtainable labels are the $\mathbb{F}$-linear span of X and $P_i(X)$. Furthermore, there is no efficient isomorphism from $\mathbb{G}_2$ or $\mathbb{G}_T$ to $\mathbb{G}_1$.
In particular after q queries, it can construct $[F_{a,b,\boldsymbol{c}}(x)]_{\mathbb{G}_1}$ for,

$$F_{a,b,\boldsymbol{c}}(X) \coloneqq aX + b + \sum_{j \in [q]} \frac{c_j}{X+\rho_j} \qquad \text{for any } a, b \in \mathbb{F}, \boldsymbol{c} \in \mathbb{F}^q$$

We suppress $a, b, \boldsymbol{c}$ and write $F \coloneqq F_{a,b,\boldsymbol{c}}$ when it's clear. Let $H_j(X) = \frac{1}{(X+\rho_j)^2}$ for $j \in [q]$.
Let $Z(X) = \prod_{i \in [q]}(X + \rho_i)$. Observe that $F(X) \cdot Z(X)$ is a polynomial in $\mathbb{F}[X]$ while $H(X) \cdot Z(X) = \frac{\prod_{i \neq j}(X+\rho_i)}{(X+\rho_j)}$ is a rational function in $\mathbb{F}(X)$. Therefore, $H_j(X) \neq F(X)$.
$\mathcal{A}$ wins if and only if $[F(x)]_{\mathbb{G}_1} = [H_j(x)]_{\mathbb{G}_1}$ for any $j \in [q]$. We show that this happens with negligible probability for randomly sampled $x \xleftarrow{\$} \mathbb{F}$.
Fix any $j \in [q]$. Define $Z^j(X) = Z(X) \cdot (X + \rho_j)$. Then, $F(X) \cdot Z^j(X) \in \mathbb{F}[X]$ is a polynomial of degree $q+2$. Moreover, $H_j(X) \cdot Z^j(X) \in \mathbb{F}[X]$ and has degree $q+1$. Thus, $(F(X) - H_j(X))Z^j(X)$ is a non-zero polynomial (since $F(X) \neq H_j(X)$) of degree $q+2$. Observe that $F(x) = H_j(x)$ if and only if $(F(x) - H_j(X))Z^j(x)$ (except when $x \in \{-\rho_1, \ldots, -\rho_q\}$ which happens with probability at most q/p).
Since $(F(X) - H_j(X))Z^j(X) \neq 0$ and $x \xleftarrow{\$} \mathbb{F}$ uniformly at random, we obtain,

$$\Pr\big[(F(x) - H(x))Z^j(x) = 0\big] \le \frac{q+2}{p}$$

by the Schwartz-Zippel lemma. Therefore, $\Pr[F(x) = H_j(x)] \le O(q/p) = \mathsf{negl}(\lambda)$ for any polynomial $q = O(\lambda)$ and $p = \Omega(2^\lambda)$.
Therefore, the probability of generic $\mathcal{A}$ to output a label, $[H_j(x)]_{\mathbb{G}_1}$ for any $j \in [q]$ is negligible. □

A.2 Generic Hardness of q-DistPole

Lemma 9. q-DistPole *(Assumption 2) holds in the GBGM.*

Proof. We proceed as above. Let X be the formal variable for $[x]_{\mathbb{G}_i}$, the label of $[x]_i$ sent by the challenger CH to the generic adversary $\mathcal{A}$. For any query ρ_i for $i \in [q]$, CH responds with the correct labels, $[P_i(x)]_{\mathbb{G}_1}, [P_i(x)]_{\mathbb{G}_2}$ where $P_i(X) \in \mathbb{F}(X), P_i(X) \coloneqq \frac{1}{X+\rho_i}$.
Since $\mathcal{A}$ is generic, after q queries, it can construct $[F_{a,b,\boldsymbol{c}}(x)]_{\mathbb{G}_1}$ for,

$$F_{a,b,\boldsymbol{c}}(X) \coloneqq aX + b + \sum_{j\in[q]} \frac{c_j}{X+\rho_j} \quad \text{for any } a, b \in \mathbb{F}, \boldsymbol{c} \in \mathbb{F}^q$$

We suppress $a, b, \boldsymbol{c}$ and write $F \coloneqq F_{a,b,\boldsymbol{c}}$ when it's clear.
Fix $\rho \notin \{\rho_1, \ldots, \rho_q\}$ Let $H(X) = \frac{1}{(X+\rho)}$. Let $Z(X) = \prod_{i\in[q]}(X+\rho_j)$. Then, $F(X)\cdot Z(X) \in \mathbb{F}[X]$ while $H(X)\cdot Z(X) \in \mathbb{F}(X)$. Therefore, $F(X)\cdot Z(X) \neq H(X)\cdot Z(X)$.
$\mathcal{A}$ wins if and only if $[F(x)]_{\mathbb{G}_1} = [H(x)]_{\mathbb{G}_1}$. We show that this happens with negligible probability for randomly sampled $x \xleftarrow{\$} \mathbb{F}$.
Let $Z(X) = \prod_{i\in[q]}(X+\rho_i)$. Observe that $F(X)\cdot Z(X)$ is a polynomial in $\mathbb{F}[X]$ while $H(X)\cdot Z(X) = \frac{\prod_{i\neq j}(X+\rho_i)}{(X+\rho_j)}$ is a rational function in $\mathbb{F}(X)$. Therefore, $H_j(X) \neq F(X)$.
$\mathcal{A}$ wins if and only if $[F(x)]_{\mathbb{G}_2} = [H(x)]_{\mathbb{G}_2}$. We show that this happens with negligible probability for randomly sampled $x \xleftarrow{\$} \mathbb{F}$.
Define $Z'(X) = Z(X)\cdot(X+\rho)$. Then, $F(X)\cdot Z'(X) \in \mathbb{F}[X]$ is a polynomial of degree $q+2$. Moreover, $H(X)\cdot Z'(X) \in \mathbb{F}[X]$ and has degree $q+1$. The remainder of the proof is identical to the proof of Lemma 8 above. □

A.3 Generic Hardness of q-RankDef

Lemma 10. q-RankDef *(Assumption 3) holds in the GBGM.*

Proof. Let X be the formal variable for $[x]_{\mathbb{G}_i}$, the label of $[x]_i$ sampled by the challenger CH. For any $y \in \mathbb{Z}_p$, define $P_y(X) \in \mathbb{F}(X)$ as $P_y(X) \coloneqq \frac{1}{X+y}$. CH maintains sets S_1 and S_2 as in the definition.
For query (target, t), CH responds with the correct label, $[P_t(x)]_{\mathbb{G}_1}$ Similarly, for any query (i, j), CH responds with the correct labels, $[P_j(x)]_{\mathbb{G}_i}$. For query $(\mathsf{Combine}, S)$, CH samples $b \xleftarrow{\$} \{0,1\}, r, r' \xleftarrow{\$} \mathbb{Z}_p$ and responds with the correct labels,

$$\left([r]_{\mathbb{G}_1}, \left[r\left(\left(\sum_{i\in S} P_i(x)\right) + P_t(x)\right)\right]_{\mathbb{G}_1}, [u_b(x)]_{\mathbb{G}_T}\right)$$

where $u_0(x) = r\cdot P_t(x)$ and $u_1(x) = r'$.
Finally, $\mathcal{A}$ responds with a bit $b' \in \{0,1\}$.
Suppose $|S_2| < |S|$, $t \notin S_2 \cup S$ and $S \subseteq S_1$. (Because $\mathcal{A}$ loses in both cases otherwise.) Then $\mathcal{A}$ wins if and only if $b' = b$. We show this happens with negligible probability by showing that for uniformly random $x \xleftarrow{\$} \mathbb{Z}_p$, the labels $[u_0(x)]_{\mathbb{G}_T}$ and $[u_1(x)]_{\mathbb{G}_T}$ are computationally indistinguishable. In particular, we show given all the labels $\mathcal{A}$ obtains, $[u_0(x)]_{\mathbb{G}_T}$ is uniformly random.
Since $\mathcal{A}$ is generic, it is sufficient to show that $u_0(X)$ cannot be written as a linear combination of (symbolic representation of) labels in $\mathbb{G}_T$ [5, 9]. $\mathcal{A}$ can compute the labels in $\mathbb{G}_T$ using the pairing oracle for the following representations evaluated at x,

– 1, r,
– $P_j(X)$ for any $j \in S_1 \cup S_2$,
– $rP_j(X)$ for any $j \in S_2$,
– $P_i(X) \cdot P_j(X)$ For any $i \in S_1, j \in S_2$,
– $r\big(\big(\sum_{i\in S} P_i(X)\big) + P_t(X)\big)P_j(X)$ for any $j \in S_2$, and
– $r\big(\big(\sum_{i\in S} P_i(X)\big) + P_t(X)\big)$

Suppose for a contradiction, there exists $a, b, v_i, d_j, f_{i,j}, g_i, m \in \mathbb{F}$ such that,

$$u_0(X) = rP_t(X) = a + rb + \sum_{j\in S_1\cup S_2} v_j P_j(X) + r\sum_{j\in S_2} d_j P_j(X)$$
$$+ \sum_{i\in S_1, j\in S_2} f_{i,j} P_i(X)P_j(X) + r\sum_{j\in S_2} g_j P_j(X)\left(\sum_{i\in S} P_i(X) + P_t(X)\right) + rm\left(\left(\sum_{i\in S} P_i(X)\right) + P_t(X)\right)$$

(That is, u_0 is a linear combination of all computable labels in $\mathbb{G}_T$.)
Isolating $rP_t(X)$ in the equation,

$$rP_t(X)\left(-\sum_{j\in S_2} g_j P_j(X) - m\right) = a + rb + \sum_{j\in S_1\cup S_2} v_j P_j(X) + r\sum_{j\in S_2} d_j P_j(X)$$
$$+ \sum_{i\in S_1, j\in S_2} f_{i,j} P_i(X)P_j(X) + r\sum_{j\in S_2} g_j P_j(X)\left(\sum_{i\in S} P_i(X)\right) + rm\left(\sum_{i\in S} P_i(X)\right)$$

We partition S_2 into $S_2 \setminus S$ and $S_2 \cap S$ on the RHS for terms that include r. Rewriting,

$$rP_t(X)\left(-\sum_{j\in S_2} g_j P_j(X) - m\right) = a + rb + \sum_{j\in S_1\cup S_2} v_j P_j(X) + r\sum_{j\in S_2} d_j P_j(X)$$
$$+ \sum_{i\in S_1, j\in S_2} f_{i,j} P_i(X)P_j(X) + r\sum_{j\in S_2\cap S} g_j P_j^2(X) + r\sum_{j\in S_2\cap S} g_j P_j(X)\left(\sum_{i\in S\setminus\{j\}} P_i(X)\right)$$
$$+ r\sum_{j\in S_2\setminus S} g_j P_j(X)\left(\sum_{i\in S} P_i(X)\right) + rm\left(\sum_{i\in S} P_i(X)\right)$$

Observe that $P_j^2(X)$ for each $j \in S_2 \cap S$ are the only *square poles* in the equation. By the $\mathbb{F}$-independence of rational functions with distinct poles, we must have that $rg_j = 0$ for all $j \in S_2 \cap S$. Since r is uniformly random, $g_j = 0$ (except with negligible probablity). Simplifying, we obtain,

$$rP_t(X)\left(-\sum_{j\in S_2\setminus S} g_j P_j(X) - m\right) = a + rb + \sum_{j\in S_1\cup S_2} v_j P_j(X) + r\sum_{j\in S_2} d_j P_j(X)$$
$$+ \sum_{i\in S_1, j\in S_2} f_{i,j} P_i(X)P_j(X) + r\sum_{j\in S_2\setminus S} g_j P_j(X)\left(\sum_{i\in S} P_i(X)\right) + rm\left(\sum_{i\in S} P_i(X)\right)$$

Since S and $S_2 \setminus S$ are disjoint, and $t \notin S_2$, we apply Lemma 2 to decompose the products $P_t(X)P_j(X)$ and $P_i(X)P_j(X)$ as follows,

$$
\begin{aligned}
&- r\left(\sum_{j\in S_2\setminus S} \frac{g_j}{t-j}(P_j(X) - P_t(X))\right) - rmP_t(X) \\
&= a + rb + \sum_{j\in S_1\cup S_2} v_j P_j(X) + r\sum_{j\in S_2} d_j P_j(X) + \sum_{i\in S_1, j\in S_2} f_{i,j} P_i(X)P_j(X) \\
&+ r\sum_{i\in S}\sum_{j\in S_2\setminus S} \frac{g_j}{i-j}(P_j(X) - P_i(X)) + rm\left(\sum_{i\in S} P_i(X)\right)
\end{aligned}
$$

which implies,

$$
\begin{aligned}
&- r\sum_{j\in S_2\setminus S} \frac{g_j}{t-j} P_j(X) + rP_t(X)\left(\sum_{j\in S_2\setminus S} \frac{g_j}{t-j}\right) - rmP_t(X) \\
&= a + rb + \sum_{j\in S_1\cup S_2} v_j P_j(X) + r\sum_{j\in S_2} d_j P_j(X) + \sum_{i\in S_1, j\in S_2} f_{i,j} P_i(X)P_j(X) \\
&+ r\sum_{j\in S_2\setminus S}\sum_{i\in S} \frac{g_j}{i-j} P_j(X) - r\sum_{i\in S} P_i(X) \sum_{j\in S_2\setminus S} \frac{g_j}{i-j} + rm\left(\sum_{i\in S} P_i(X)\right)
\end{aligned}
$$

Combining $rP_i(X)$ terms on the right and $rP_t(X)$ terms on the left,

$$
\begin{aligned}
&- r\sum_{j\in S_2\setminus S} \frac{g_j}{t-j} P_j(X) + rP_t(X)\left(\sum_{j\in S_2\setminus S} \frac{g_j}{t-j} - m\right) \\
&= a + rb + \sum_{j\in S_1\cup S_2} v_j P_j(X) + r\sum_{j\in S_2} d_j P_j(X) + \sum_{i\in S_1, j\in S_2} f_{i,j} P_i(X)P_j(X) \\
&+ r\sum_{j\in S_2\setminus S}\sum_{i\in S} \frac{g_j}{i-j} P_j(X) - r\sum_{i\in S} P_i(X)\left(\sum_{j\in S_2\setminus S} \frac{g_j}{i-j} + m\right)
\end{aligned}
$$

Finally, we decompose the last remaining sum over S_2 and combine $rP_i(X)$ terms in $S \setminus S_2$ to obtain,

$$
\begin{aligned}
&- r\sum_{j\in S_2\setminus S}\left(\frac{g_j}{t-j} + \sum_{i\in S}\frac{g_j}{i-j} + d_j\right) P_j(X) + rP_t(X)\left(\sum_{j\in S_2\setminus S} \frac{g_j}{t-j} - m\right) \\
&= a + rb + \sum_{j\in S_1\cup S_2} v_j P_j(X) + \sum_{i\in S_1, j\in S_2} f_{i,j} P_i(X)P_j(X) \\
&- r\sum_{i\in S\cap S_2} P_i(X)\left(\sum_{j\in S_2\setminus S} \frac{g_j}{i-j} + m - d_j\right) - r\sum_{i\in S\setminus S_2} P_i(X)\left(\sum_{j\in S_2\setminus S} \frac{g_j}{i-j} + m\right)
\end{aligned}
$$

Observe that for $i \in S\setminus S_2$, $P_i(X)$ are only on the RHS. Again, by $\mathbb{F}$-independence of rational functions with distinct poles, we must have the coefficient of $P_i(X)$ be 0. Since r is uniformly

random, we must furthermore have $\left(\sum_{j\in S_2\setminus S}\frac{g_j}{i-j}+m\right)=0$ for all $i\in S\setminus S_2$. This defines an $\mathbb{F}$-linear system on variables g_j for $j\in S_2\setminus S$ with $|S\setminus S_2|$ equations, with more equations than unknowns. In fact, these equations over unknowns g_i form an overdetermined Cauchy matrix with an all $-m$ vector on the RHS, which cannot have a solution.
To see this consider the rational function $f(X)=\sum_{j\in S_2\setminus S}\frac{g_j}{X-j}$. We have $f(x)=-m$ for $x\in S\setminus S_2$. Now write $f(X)=\frac{N(X)}{\prod_{j\in S_2\setminus S}(X-j)}$. Then we have:

$$N(X)=\sum_{j\in S_2\setminus S} g_j\prod_{i\neq j}(X-i)$$

Therefore $N(X)$ is a degree $|S_2\setminus S|-1$ polynomial with:

$$N(x)=-m\prod_{j\in S_2\setminus S}(x-j),\ \text{for all } x\in S\setminus S_2.$$

That is, the degree $|S_2\setminus S|-1$ degree polynomial $N(X)+m\prod_{j\in S_2\setminus S}(X-j)$ has $|S\setminus S_2|$ zeroes. Hence it must be identically 0. That is $f(X)$ is identically $-m$, which is a contradiction. This concludes the proof. □

B Special Cauchy Matrix

Lemma 5 (repeated) The (i,j)-th entry c^*_{ij} $(i,j\in[m])$ of inverse of matrix $H(\boldsymbol{a},\boldsymbol{b})$ is given by

$$c^*_{ij}=\begin{cases}\frac{\prod_{k\in[m-1]}(a_k-b_i)}{\prod_{\substack{k\in[m-1]\\ k\neq j-1}}(a_{j-1}-a_k)}\prod_{\substack{k\in[m]\\ k\neq i}}\frac{(a_{j-1}-b_k)}{(b_k-b_i)} & \text{for } j\neq 1\\ \frac{\prod_{k\in[m-1]}(a_k-b_i)}{\prod_{k\in[m],\ k\neq i}(b_k-b_i)} & \text{for } j=1\end{cases}$$

Proof. We first note that the formula for inverse elements in lemma 3 holds for any integer domain and not just finite fields. Next, any finite field $\mathbb{F}$ is isomorphic to $\mathbb{Z}[X]/(p,f(X))$ for some irreducible (over $\mathbb{Z}_p$) polynomial $f(X)\in\mathbb{Z}_p[X]$, where p is a prime integer. This implies $f(X)$ is also irreducible over $\mathbb{Z}[X]$ and also over $\mathbb{Q}[X]$. Thus, any non-zero element in $g(X)\in\mathbb{Z}[X]/(f(X))$ has an inverse in $\mathbb{Q}[X]$, because the gcd of $g(X)\neq 0 \bmod f(X)$ and $f(X)$ will be an integer z. Thus, the formula for c'_{ij} in lemma 3 can be evaluated to be in $\mathbb{Q}[X]/(p,f(X))$. In fact, this polynomial in $\mathbb{Q}[X]$ is actually of the form $\frac{1}{z}\cdot h(X)$, where $h(X)\in\mathbb{Z}[X]$. Having thus simplified c'_{ij} as in lemma 3 we can invert $z \bmod p$, as long as $z\neq 0 \bmod p$, to obtain c'_{ij} in $\mathbb{F}=\mathbb{Z}[X]/(p,f(X))$.
We now define a Cauchy-sequence of $m\times m$ matrices $H_N(\boldsymbol{a},b)$ (over $\mathbb{Q}[X]$)

$$H_N(\boldsymbol{a},\boldsymbol{b})=\begin{bmatrix}\langle\frac{N}{N-\boldsymbol{b}_j}\rangle_{j\in[m]}\\ \mathcal{C}(\boldsymbol{a},\boldsymbol{b})\end{bmatrix}$$

Note that the Cauchy limit of this matrix is just the original matrix $H(\boldsymbol{a},\boldsymbol{b})$. More importantly, $H_N(\boldsymbol{a},\boldsymbol{b})=\mathsf{diag}(N,1,...,1)\cdot\mathcal{C}(\boldsymbol{a}^*_N,\boldsymbol{b})$, where $\boldsymbol{a}^*_N$ is an m-vector that is N appended with original $\boldsymbol{a}$. Thus, $H_N^{-1}=\mathcal{C}(\boldsymbol{a}^*_N,\boldsymbol{b})^{-1}\cdot\mathsf{diag}(1/N,1,...,1)$. We now evaluate c'_{ij} using the above formula to compute H_N^{-1} as a fraction of polynomials in $\mathbb{Z}[X]$. We will denote

$c^*_{ij,N} = c'_{ij}/M_j$, where $M_j = N$ if $j = 1$, and $M_j = 1$ otherwise. The $c^*_{ij,N}$ then give the entries of H_N^{-1}. It is easy to check that (for $j \neq 1$)

$$\begin{aligned} c^*_{ij,N} &= \frac{1}{M_j} \cdot \frac{\prod_{k\in[m]}(a^*_k - b_i)}{\prod_{\substack{k\in[m]\\k\neq j}}(a^*_j - a^*_k)} \prod_{\substack{k\in[m]\\k\neq i}} \frac{(a^*_j - b_k)}{(b_k - b_i)} \\ &= \frac{1}{1} \cdot \frac{(N - b_i)\prod_{k\in[m-1]}(a_k - b_i)}{(a_{j-1} - N)\prod_{\substack{k\in[m-1]\\k\neq j-1}}(a_{j-1} - a_k)} \prod_{\substack{k\in[m]\\k\neq i}} \frac{(a_{j-1} - b_k)}{(b_k - b_i)} \\ &\underset{N\to\infty}{=} \frac{\prod_{k\in[m-1]}(a_k - b_i)}{\prod_{\substack{k\in[m-1]\\k\neq j-1}}(a_{j-1} - a_k)} \prod_{\substack{k\in[m]\\k\neq i}} \frac{(a_{j-1} - b_k)}{(b_k - b_i)} \end{aligned}$$

and (for $j = 1$)

$$c^*_{ij,N} = \frac{1}{N} \cdot \frac{(N - b_i)\prod_{k\in[m-1]}(a_k - b_i)}{\prod_{k\in[m-1]}(N - a_k)} \prod_{\substack{k\in[m]\\k\neq i}} \frac{(N - b_k)}{(b_k - b_i)} \underset{N\to\infty}{=} \frac{\prod_{k\in[m-1]}(a_k - b_i)}{\prod_{\substack{k\in[m]\\k\neq i}}(b_k - b_i)}$$

Noting that in both cases, as N tends to infinite, the terms involving N cancel out. It now suffices to show that if for all $j \neq k$, $a_j \neq a_k$ and similarly $b_j \neq b_k$ (in $Z[X]/(p, f(X))$), the denominators treated as polynomials in $\mathbb{Z}[X]$ have a resultant with $f(X)$ that is non-zero modulo p. However, if this was not the case, then that would mean the denominator is not invertible modulo $(p, f(X))$, which is a contradiction. Finally, we remark that since the matrices are of finite dimension, an easy Cauchy-limit type argument shows that the inverse of H is indeed the limit of the inverse of H_N (as $N \to \infty$). □

C CCA-2 Security of DTE

We outline an extension of the Dynamic Threshold Encryption (DTE) described in Section 5.2 and proven CPA-secure to be CCA2-secure. Further, we will show that this scheme is *publicly-verifiable* CCA2-secure. In a publicly-verifiable CCA2-secure scheme there is a public verify function (possibly using a CRS and/or public-key) such that it outputs $\perp$ on a ciphertext iff the decryptor (with secret keys) output $\perp$ on the same ciphertext. This is extremely useful in combating network attacks such as denial of service attacks, where even the routers can remove junk ciphertexts. In general CCA2 is useful as it makes the scheme non-malleable [29].

We first extend the definition of DTE as in Section 5.2 to include another function, as described next. We also define a new composite function comboDecrypt.

- $\mathsf{Validate}(\mathsf{ek}, S, t, \mathsf{ct}) \to 0/1$. On input encryption key ek for authorized set S, a threshold t, and ciphertext ct, outputs 0 if ct is invalid and 1 otherwise.
- $\mathsf{comboDecrypt}(\mathsf{ck}, S, t, \mathsf{ct}, T, D, \mathsf{ek}) \to m/\perp$. On input the combining key ck, authorized set S, threshold t, ciphertext ct, a subset of authorized users $T \subseteq S$ of size $t' \geq t$, and set of t' secret keys $\mathsf{sk}_{\mathsf{id}}$ corresponding to S, and ek corresponding to S, outputs a plaintext message, m or $\perp$. It employs validate on ct using ek. If validate fails it outputs $\perp$. Otherwise it does t partial decryptions and then combines to output m.

The definition of CPA-security of DTE as described in Definition 8 is extended as follows to get CCA2-security. Essentially, now the adversary has access to oracle $\mathsf{comboDecrypt}$, which can be called with any message except the ciphertext returned by the challenge encryption in the CPA game. The decryption oracle can even be called after the adversary receives the challenge ciphertext.
We describe two schemes. One using Groth-Sahai proofs [21], which have a short CRS, and another using quasi-adpative NIZKs [25], which have a shorter proof (in terms of group elements) but a larger CRS.

$\mathsf{Validate}(\mathsf{ek}, S, t, \mathsf{ct})$*:* To validate the ciphertext and enable CCA2-security we extend the ciphertext by several elements as described. To start with, an element $\mathsf{ct}_4 = r \cdot e(\mathsf{altk}, 1) + m$ is included, where altk is chosen randomly in $\mathbb{G}_1$ during setup and is part of ek. Next, the ciphertext must include a one-time simulation-sound NIZK proof of the following relation:
$\exists r \in \mathbb{Z}_p, \exists Y \in \mathbb{G}_1$:
– $\mathsf{ct}_1 = r \cdot [1]_1$,
– $\mathsf{ct}_4 - \mathsf{ct}_3 = e(Y, 1)$,
– $Y = r \cdot (\mathsf{altk} - \mathsf{tk})$.
A Groth-Sahai NIZK proof of this can be made one-time simulation-sound by including a 2-universal hash-proof [11] and additionally proving that the hash proof is correct (see e.g. [25]). The hash proof requires two more elements in ct and the net Groth-Sahai proof has 9 elements in $\mathbb{G}_1$ and 6 in $\mathbb{G}_2$. The Groth-Sahai CRS is made part of the above CRS generated during setup.

Here is an alternate using QA-NIZK [25]. Now, ct contains only 4 more total elements out of which 2 are for a 1-ss qa-nizk proof. Let $\mathsf{ct}_4\ (\in \mathbb{G}_T) = r \cdot (\mathsf{altk}, 1) + m$, and $\mathsf{ct}_5\ (\in \mathbb{G}_1) = r \cdot (\mathsf{tk} - \mathsf{altk})$. Here, altk is a new element in CRS in $\mathbb{G}_1$ (chosen randomly during setup). Note that $\mathsf{ct}_3 - \mathsf{ct}_4 = e(\mathsf{ct}_5, 1)$, so this is checked first during validation. In addition, the ciphertext includes a two group element one-time simulation-sound (qa)-nizk proof[15] $\pi = (\mathsf{ct}_6, \mathsf{ct}_7)$ [26] for the following relation which is checked for validation. The CRS also has qa-nizk CRS (that has 8 group elements) for this relation. Since S is not pre-determined at setup, the qa-nizk CRS ends up having $4(n+1)$ group elements [26, Fig 6].
– $\exists r : \mathsf{ct}_1 = r \cdot [1]_1$ *and* $\mathsf{ct}_2 = r \cdot \left(\mathsf{ek}_S + \sum_{i=s+1}^{s+t} \mathsf{dpk}_j\right)$, *and* $\mathsf{ct}_5 = r \cdot (\mathsf{tk} - \mathsf{altk})$

D Other Applications

In this section, we discuss other applications of our construction. These applications demonstrate more construction patterns that partial fraction decomposition can be used to achieve. However, we do not develop full formal treatment for these applications, leaving it to future work.

D.1 Hadamard Product Proofs

The Hadamard (or entrywise) product of vectors is a fundamental operation in many cryptographic protocols, particularly in zero-knowledge proof systems for arithmetic circuits

[15] This requires a collision-resistant hash function employed on rest of the ciphertext, which is used as a tag in proof generation.

targeted for inner products [10], or R1CS [20]. We show how partial fraction decomposition provides a natural framework for proving Hadamard product relations using bilinear pairings.

The Problem. Let $\boldsymbol{u} = (u_1, \ldots, u_n) \in \mathbb{F}^n$, $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{F}^n$, and $\boldsymbol{w} = (w_1, \ldots, w_n) \in \mathbb{F}^n$ be three vectors. We want to prove that $\boldsymbol{u} \circ \boldsymbol{v} = \boldsymbol{w}$, i.e., that $u_i \cdot v_i = w_i$ for all $i \in [n]$, where $\circ$ denotes the Hadamard (entrywise) product.
Suppose we have commitments to these vectors using the partial fraction representation:

$$\mathsf{com}_u = \left[\sum_{i=1}^{n} \frac{u_i}{x+i}\right]_1, \quad \mathsf{com}_v = \left[\sum_{i=1}^{n} \frac{v_i}{x+i}\right]_2, \quad \mathsf{com}_w = \left[\sum_{i=1}^{n} \frac{w_i}{x+i}\right]_1,$$

where $x \in \mathbb{F}$ is a secret trapdoor, and $[\cdot]_1, [\cdot]_2$ denote group elements in $\mathbb{G}_1, \mathbb{G}_2$ respectively in an asymmetric bilinear group.

Key Algebraic Observation. The key insight is that the product of two partial fraction sums naturally encodes the Hadamard product through the pairing operation. Consider the rational function product:

$$\left(\sum_{i=1}^{n} \frac{u_i}{X+i}\right) \cdot \left(\sum_{i=1}^{n} \frac{v_i}{X+i}\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{u_i v_j}{(X+i)(X+j)}.$$

We can decompose this product into partial fractions. For $i = j$, the terms $\frac{u_i v_i}{(X+i)^2}$ do not directly correspond to our linear partial fraction basis. However, we can take these terms out by considering a modified expression:

$$H(X) \coloneqq \left(\sum_{i=1}^{n} \frac{u_i}{X+i}\right) \cdot \left(\sum_{i=1}^{n} \frac{v_i}{X+i}\right) - \left(\sum_{i=1}^{n} \frac{w_i}{X+i}\right) \cdot \left(\sum_{i=1}^{n} \frac{1}{X+i}\right).$$

Thus, $H(X)$ can be written as:

$$H(X) = \sum_{i=1}^{n} \frac{h_i}{X+i},$$

where the coefficients h_i are computable from $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$. The Hadamard product relation holds if and only if we can express $H(X)$ in this form.

Cryptographic Verification. In the cryptographic setting, the prover demonstrates that $H(X)$ has the correct form by computing:

$$\mathsf{com}_h = \left[\sum_{i=1}^{n} \frac{h_i}{x+i}\right]_1.$$

The verifier checks the pairing equation:

$$e(\mathsf{com}_u, \mathsf{com}_v) \stackrel{?}{=} e(\mathsf{com}_w, \mathsf{com}_{\mathbf{1}}) \cdot e(\mathsf{com}_h, [1]_2),$$

where $\mathsf{com}_{\mathbf{1}} = \left[\sum_{i=1}^{n} \frac{1}{x+i}\right]_2$ is a public parameter that can be precomputed.

D.2 Registration-based Encryption

Registration-Based Encryption (RBE) [16] is a public-key encryption paradigm that enables identity-based encryption without trusted key generation. In an RBE scheme, users independently generate their own public-private key pairs and register their public keys with an untrusted Key Curator (KC). The KC maintains public parameters that enable anyone to encrypt messages to registered identities, but the KC cannot decrypt messages—privacy is guaranteed even if the KC is malicious.

The RBE Model. Unlike Identity-Based Encryption (IBE) [7], where a trusted authority extracts secret keys for users, RBE separates the registration and key generation processes:
- Each user samples their own secret key independently, maintaining full control over their private information.
- The KC only maintains public parameters and user-specific auxiliary decryption information that is necessary but not sufficient for decryption.
- Encryption uses only the recipient's identity and the public parameters—no per-user public keys are needed.
- Decryption requires both the user's secret key and the public auxiliary information from the KC.

This model is particularly suitable for blockchain and decentralized applications where a proof-of-identity or registration authority exists but should not be trusted with decryption capabilities.

Our Construction via Partial Fractions. We construct an RBE scheme using partial fractions and bilinear groups, adapting [18]. Our approach builds upon the KVC framework by representing registered identities as partial fractions $\frac{1}{X+\mathsf{H}(\mathsf{id})}$. The key insight is that the public parameters can be viewed as a commitment to all registered public keys, and the auxiliary decryption information can be computed using the linear independence property of partial fraction products. We remark that the construction given below is in *insufficient*. In particular the CRS is linear in the number of identities N rendering it no better than the trivial approach of publishing and storing N public keys. We leave compressing the CRS to sublinear size (e.g. as done in [18]) to future work.
Let $\mathsf{H} : \{0,1\}^* \to \mathbb{F}$ be a collision-resistant hash function. For a set of registered identities $\mathsf{IDSet} = \{\mathsf{id}_1, \ldots, \mathsf{id}_N\}$, we define:

Construction. An RBE scheme consists of algorithms $\mathsf{RBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Register}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

1. $\mathsf{Setup}\left(1^\lambda, \mathsf{IDSet} := \{\mathsf{id}_i\}_{i\in[N]}\right) \to \mathsf{crs}$:
   - Generate a bilinear group, $\mathcal{G} = (p, \mathbb{G}, \mathbb{G}_T, g, e)$.
   - Sample $z \xleftarrow{\$} \mathbb{Z}_p$.
   - Output $\mathsf{crs} := \left(\mathcal{G}, \left(\left[\frac{1}{z+\mathsf{H}(\mathsf{id}_i)}\right]\right)_{i\in[N]}\right)$.
2. $\mathsf{KeyGen}(\mathsf{crs}, \mathsf{id} \in \mathsf{IDSet}) \to (\mathsf{pk}, \mathsf{sk}, \boldsymbol{\xi})$:
   - Sample $x_{\mathsf{id}} \xleftarrow{\$} \mathbb{Z}_p$.

– Output $\left(\mathsf{pk} := x_{\mathsf{id}} \cdot \left[\frac{1}{z+\mathsf{H}(\mathsf{id})}\right], \mathsf{sk} := x_{\mathsf{id}}, \boldsymbol{\xi} := \left(x_{\mathsf{id}} \cdot \left[\frac{1}{z+\mathsf{H}(\mathsf{id}')}\right]\right)_{\mathsf{id}' \neq \mathsf{id}}\right)$. Let $\xi_{\mathsf{id}'} = x_{\mathsf{id}} \cdot \left[\frac{1}{z+\mathsf{H}(\mathsf{id}')}\right]$.

3. $\mathsf{Register}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id} \in \mathsf{IDSet}, \mathsf{pk}, \boldsymbol{\xi}) \to \mathsf{pp}'$:
   – Check correctness of $\boldsymbol{\xi}$: for all $i \in [N]$, verify $e\left(\xi_{\mathsf{id}_i}, \left[\frac{1}{z+\mathsf{H}(\mathsf{id})}\right]\right) \stackrel{?}{=} e\left(\mathsf{pk}, \left[\frac{1}{z+\mathsf{H}(\mathsf{id}_i)}\right]\right)$. (This can be batched into a total of just two pairings using a random linear combination.)
   – Set $\mathsf{pp}' := \mathsf{pp} + \mathsf{pk}$.
   – Parse aux as $\left(\Lambda_{\mathsf{id}_j}\right)_{\mathsf{id}_j \in \mathsf{IDSet}}$ and set $\Lambda_{\mathsf{id}'} \leftarrow \Lambda_{\mathsf{id}'} + \frac{1}{\mathsf{H}(\mathsf{id})-\mathsf{H}(\mathsf{id}')}(\xi_{\mathsf{id}'} - \mathsf{pk})$ for $\mathsf{id}' \neq \mathsf{id}$.
   – Output pp'.
4. $\mathsf{Update}^{[\mathsf{aux}]}(\mathsf{pp}, \mathsf{id}) \to \mathsf{u}$:
   – parse aux as $\left(\Lambda_{\mathsf{id}_j}\right)_{\mathsf{id}_j \in \mathsf{IDSet}}$ and output Λ_{id}.
5. $\mathsf{Encrypt}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, m) \to \mathsf{ct}$:
   – Sample $r \xleftarrow{\$} \mathbb{Z}_p$.
   – Output $\mathsf{ct} := \left(r \cdot e\left(\mathsf{pp}, \left[\frac{1}{z+\mathsf{H}(\mathsf{id})}\right]\right), [r], r \cdot e\left(\left[\frac{1}{z+\mathsf{H}(\mathsf{id})}\right], \left[\frac{1}{z+\mathsf{H}(\mathsf{id})}\right]\right) + m\right)$
6. $\mathsf{Decrypt}(sk, \mathsf{u}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3)) \to m$:
   – Check $e\left(\mathsf{pp} - \left[\frac{x_{\mathsf{id}}}{z+\mathsf{H}(\mathsf{id})}\right], \left[\frac{1}{z+\mathsf{H}(\mathsf{id})}\right]\right) \stackrel{?}{=} e(\mathsf{u}, [1])$.
   – Output $\mathsf{ct}_3 - \frac{1}{\mathsf{sk}}(\mathsf{ct}_1 - e(\mathsf{u}, \mathsf{ct}_2))$.

Intuition and Correctness. The public parameters pp represent the sum $\sum_{j \in \text{Registered}} \frac{x_{\mathsf{id}_j}}{z+\mathsf{H}(\mathsf{id}_j)}$. When encrypting to identity id with index i, the term $e(\mathsf{pp}, [\frac{1}{z+i}]_2)$ computes a product of rational fractions. The auxiliary information Λ_i captures the decomposition of this product excluding the target user's own contribution, enabling them to "cancel out" everyone else's keys and recover the message using their secret key x_{id}.
The correctness follows from the partial fraction product decomposition property:

$$\left(\sum_{j \neq i} \frac{x_{\mathsf{id}_j}}{X + \mathsf{H}(\mathsf{id}_j)}\right) \cdot \frac{1}{X + \mathsf{H}(id_i)} = \sum_{j=1}^{N} \frac{h_{ij}}{X + \mathsf{H}(\mathsf{id}_j)},$$

where the coefficients h_{ij} define $\Lambda_i = [\sum_{j=1}^{N} \frac{h_{ij}}{z+\mathsf{H}(\mathsf{id}_j)}]_1$.

Advantages and Limitations. Our RBE construction demonstrates several advantages:
– **Compact ciphertexts:** Only $|\mathbb{G}_1| + 2|\mathbb{G}_T|$ in size.
– **Efficient encryption:** Requires only 2 pairings plus group operations.
– **Public verifiability:** Registration proofs can be verified by anyone.
– **Algebraic simplicity:** Built entirely on partial fractions without requiring SNARKs or complex polynomial commitments.

However, the current construction requires that the identity space $\{\mathsf{id}_i\}$ be fixed at setup time. Extending to fully dynamic identity spaces would require techniques such as distributed key generation for the trapdoor z or multiple non-colluding key curators—we leave this as an interesting direction for future work. Also, like we mentioned above, we have not explored compressing the CRS to sublinear size.