# Relational Hash*

Avradip Mandal and Arnab Roy

Fujitsu Laboratories of America
Sunnyvale, CA, USA
{amandal,aroy}@us.fujitsu.com

**Abstract.** Traditional cryptographic hash functions allow one to easily check whether the original plaintexts are equal or not, given a pair of hash values. Probabilistic hash functions extend this concept where given a probabilistic hash of a value and the value itself, one can efficiently check whether the hash corresponds to the given value. However, given distinct probabilistic hashes of the same value it is not possible to check whether they correspond to the same value. In this work we introduce a new cryptographic primitive called *Relational Hash* using which, given a pair of (relational) hash values, one can determine whether the original plaintexts were related or not. We formalize various natural security notions for the Relational Hash primitive - one-wayness, twin one-wayness, unforgeability and oracle simulatibility.

We develop a Relational Hash scheme for discovering linear relations among bit-vectors (elements of $\mathbb{F}_2^n$) and $\mathbb{F}_p$-vectors. Using the linear Relational Hash schemes we develop Relational Hashes for detecting proximity in terms of hamming distance. The proximity Relational Hashing schemes can be adapted to a privacy preserving biometric identification scheme, as well as a privacy preserving biometric authentication scheme secure against passive adversaries.

**Keywords:** Probabilistic Hash Functions, Functional Encryption, Biometric Authentication.

## 1 Introduction

Traditional cryptographic hash functions, like MD-5 and SHA-3, enable checking for equality while hiding the plaintexts. Since these are deterministic functions, this just involves checking if the hashes are identical. The notion of probabilistic hash functions was developed in [Can97,CMR98]. In this setting, the computation of hashes is randomized and thus no two independently generated hashes of the same plaintext look same. However, given the plaintext and a hash, it can be checked efficiently if the hash corresponds to the plaintext. Probabilistic hashes can provably enable strong privacy guarantees in standard model, like oracle simulatability, which deterministic hash functions cannot provide. Oracle simulatability captures the notion that a hash reveals nothing about the value except enabling equality checking. This typically has come at the price of efficiency. In addition, the property of compression, which is desirable for deterministic hash functions, is no longer at the forefront.

However, probabilistic hashes suffer from the drawback that for verification of equality the plaintext has to be provided in the clear, which deterministic hashes do not require. Probabilistic hashes do not allow checking whether the plaintexts are equal, given two distinct hash values. This drawback can preclude use of probabilistic hashes in certain scenarios where it is desirable to hide the plaintext from the verifier as well. For example, consider a scenario where password equality is to be checked by a server. If the server uses deterministic hashes, then only the hash of the password could be transmitted to the server. However, with probabilistic hashes, the actual password has to be sent to the server for verification[1]. Therefore question arises whether we can build probabilistic hashes which allow verification given two distinct hashes of the plaintexts.

* This is the full version of the article "Relational Hash: Probabilistic Hash for Verifying Relations, Secure against Forgery and More", which appears in proceedings of CRYPTO 2015 [MR15], © IACR 2015.

[1] We need additional protocol steps to ensure security against replay attacks and so on. However, for now, we focus on the core property of the hashes themselves.

So suppose we had a probabilistic hash function *ph* which allows efficient checking of equality of plaintexts $x_1$ and $x_2$, given $ph(x_1, r_1)$ and $ph(x_2, r_2)$, where the $r_i$'s are randomnesses used for hashing. Now we run into a different problem. The existence of such a functionality implies that a secrecy property called 2-value perfect one-wayness (2-POW) [CMR98] would no longer hold. This property states that the distribution of two probabilistic hashes of the same value is computationally indistinguishable from the distribution of probabilistic hashes of two independent values. The property trivially breaks down if we have an efficient mechanism for checking if two hashes correspond to the same plaintext. In addition to being a strong security notion, this property also implies oracle simulatability [CMR98]. So now the question is:

> How do we develop probabilistic hashes which enable equality checking just given hashes but at the same time preserve 2-value perfect one-wayness?

*Our Contributions.* We propose a cryptographic primitive called *Relational Hash* which attempts to model the question above. One of the key ideas is to have distinct, but related, hashing systems for the individual co-ordinates, i.e., have two probabilistic hash functions $ph_1$ and $ph_2$ and enable checking of $x_1 \stackrel{?}{=} x_2$, given $ph_1(x_1, r_1)$ and $ph_2(x_2, r_2)$. Having two hashing systems leaves open the possibility that they can individually be 2-POW. Extending equality, we define *Relational Hash* with respect to a relation $R$, such that given two hashes $ph_1(x_1, r_1)$ and $ph_2(x_2, r_2)$, we can efficiently determine whether $R(x_1, x_2)$ holds. It may also be desirable to compute ternary relations $R'$ on $x_1, x_2$ and a third plaintext parameter $z$, so that given $ph_1(x_1, r_1), ph_2(x_2, r_2)$ and $z$, we can efficiently determine whether $R'(x_1, x_2, z)$ holds. For any Relational Hash primitive, we formalize a few natural and desirable security properties, namely one-wayness, unforgeability, twin one-wayness and oracle simulatability. The notion of oracle simulatability was introduced in [Can97,CMR98] for the equality relation. Here we extend this concept for arbitrary relations.

For the equality relation, there is a simple construction which extends Canetti's scheme in [Can97]. While the [Can97] probabilistic hash on a plaintext $m$ and randomness $r$ is $(\mathbf{g}^r, \mathbf{g}^{rm})$, one can consider bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and define $ph_1(x_1, r_1) := (\mathbf{g}^{r_1}, \mathbf{g}^{r_1 x_1})$ and $ph_2(x_2, r_2) := (\mathbf{h}^{r_2}, \mathbf{h}^{r_2 x_2})$ with $\mathbf{g} \in \mathbb{G}_1$ and $\mathbf{h} \in \mathbb{G}_2$. Plaintext equality of two hashes $(c_1, c_2)$ and $(d_1, d_2)$ of different types can be done as: $e(c_1, d_2) \stackrel{?}{=} e(c_2, d_1)$. We do not develop this construction formally in the body of the paper, additionally relegating some proof sketches to Appendix I[2].

For hamming proximity relations among vectors, especially low characteristic ones, the constructions turn out to be far more sophisticated and form the main thrust of our paper. Towards that end, we first develop a construction for a linear Relational Hash scheme. In our scheme, for any $x, y, z \in \mathbb{F}_2^n$, given just the hashes of $x$ and $y$ and the plaintext $z$, it is possible to verify whether $x + y \stackrel{?}{=} z$. A linear Relational Hash scheme is also trivially an equality Relational Hash scheme, by taking $z$ to be all 0's. We also extend our construction to verify linear relations over $\mathbb{F}_p^n$. We show that our linear Relational Hash constructions satisfy all four security notions: one-wayness, unforgeability, twin one-wayness and oracle simulatability. Next we show that using a linear Relational Hash and error correcting codes it is possible to build Relational Hashing schemes which can verify proximity relations and enjoy one-wayness, unforgeability and a stronger version of twin one-wayness. It remains open to build a proximity Relational Hash scheme which is oracle simulation secure.

*Application.* A motivating application of the proximity relation hash primitive is a privacy preserving biometric identification scheme. Consider a scenario where there is a database of fingerprints of known criminals. The database should not reveal the actual fingerprints, even internally. An investigative officer might want to check, whether a candidate fingerprint digest matches with the database. Using a Relational Hash scheme for proximity relation, one can build a biometric identification scheme which guarantees complete template privacy (to the server, as well as to the investigating officer). While storing the fingerprints in the database, hashes of type 1 are used. On the other hand, the officer gets access to type 2 hash of the fingerprint template. The Relational Hash scheme will guarantee that, with access to a relational secret key the server can only verify whether the original templates are close to each other or not. To construct authentication schemes, rather than identification schemes, additional protocol layers are needed to address replay attacks and so

[2] We thank Mehdi Tibouchi for observing this example.

on. Merely providing a type 2 hash of the challenge biometric template does not suffice as that can easily be replayed. We leave open the construction of such protocols building on the Relational Hash primitive. However, we show that for the case of a passive adversary attempting to recover the biometric template, a Relational Hash can be seen as a biometric authentication mechanism (Section 4).

*Relation to Fuzzy Extractor/ Secure Sketch based schemes.* Existing biometric authentication schemes, e.g. fuzzy vault [JS02], fuzzy commitment [JW99] and secure sketch [DRS04,DS05] based schemes guarantee template privacy only during the registration phase. Boyen solved this issue in [Boy04], by constructing a "Zero Storage remote biometric authentication scheme", which provides complete template privacy. Boyen's construction only assumes that the biometric template comes from a high entropy distribution. Compared to that, we only achieve a passive adversary secure biometric authentication scheme assuming uniform distribution of biometric templates. On the positive side, our biometric authentication scheme is much simpler, in particular during authentication the client generates the authentication token on its own, without requiring any intervention from the server. Moreover, for our primary application - the non-interactive biometric identification mechanism, the advantage becomes more apparent. It is not readily clear whether one can build such identification mechanism based on fuzzy extractors/ secure sketches.

*Relation to Multi-Input Functional Encryption (MIFE).* Goldwasser et al proposed the concept of MIFE in [GGG$^+$14], which is a functional encryption which enables the computation of $f(x_1, x_2, \cdots, x_n)$ given the encryptions of $x_1, x_2, \cdots, x_n$. The paper [GGG$^+$14] is a merge of two independent and concurrent works [GGJS13,GKL$^+$13]. While a Relational Hash scheme for a relation $R$ can be considered an MIFE for evaluating the relation $R$, there are several important differences between the MIFE work of [GGG$^+$14] and Relational Hash. We only consider the fully public key model where encryption keys for all the co-ordinates are given to the adversary.

We first remark that an indistinguishability based functional encryption security definition (FE-IND) for the equality relation is a rather trivial notion. The FE-IND notion asks the adversary to query two sets of $n$-tuples, and the challenger randomly selects which set to encrypt. We observe that even a standard CPA secure public-key encryption scheme satisfies this notion, where the functional key is simply the secret key for decryption. The FE-IND security notion is satisfied for equality because the restriction on the adversary's queries forces it to choose equal sets of messages to the challenger. So in the end the adversary has information theoretically no clue about which of the messages was chosen for encryption by the challenger. In a Relational Hash scheme, even when given the relational key, the encryption of the plaintexts is required to be at least one-way secure. No such guarantee is provided by the standard CPA scheme, since giving the full decryption key fully exposes the plaintext to the functional key recipient.

Thus we have to resort to the simulation based security notion (FE-SIM) for any meaningful assurance of security. The only possibility result in the fully public key setting is given by [GKL$^+$13], who give a construction of FE-SIM secure encryption scheme for a class of functionalities they call "learnable" functions. They also prove that if an FE-SIM secure scheme exists for a class of functionalities, then this class must be learnable. Briefly, a 2-ary function $f(.,.)$ is learnable if, given a description of $f$ and oracle access to $f(x,.)$, one can output the description of a function that is indistinguishable from $f_x(.)$, which is the restriction of $f$ on fixing the first input to $x$. This has to hold true with high probability even if the distinguisher is given $x$. One can immediately see that equality is not a learnable function. When $x$ comes from high min-entropy distribution, it is not possible to learn the value of $x$ efficiently by querying $f(x,.)$ on various inputs. A distinguisher can immediately thwart any such 'learnt' function by simply testing it on $x$.

Thus these work(s) effectively show that there is no FE-SIM secure functional encryption scheme for the function testing equality. How does our construction get around this impossibility? The reason is that the security properties that we consider: one-wayness and unforgeability do not imply FE-SIM. The property closest to FE-SIM is oracle simulatability, but it differs from FE-SIM in that the adversary does not choose the messages to be encrypted, rather they are sampled from a distribution and only their encryption is given to the adversary.

*Relation to Property Preserving (Tagged) Encryption (PPE).* PPE [PR12] is a special case of MIFE in the symmetric key setting. PPE offers IND based security guarantees, where attacker queries are constrained such that the preserved property values cannot be trivially used for distinguishing purposes. Moreover, PPE involves a secret key, whereas for Relational Hashes all the keys are public. For our public key case, the trivial construction which makes the functional key the same as the decryption key, is IND secure and does not provide any meaningful security guarantee. On the other hand, for the symmetric key PPE schemes, chosen message security is non-trivial.

*Relation to Perceptual Image Hashing (PIH).* PIH [KVM04] is a related technique which aims to construct hash of images invariant under geometric transformations which preserve perceptual similarity. There are several differences, most importantly: (1) the primary objective of PIH is the detection of similar inputs, however privacy of the inputs may not be preserved, (2) generating hashes requires a secret key, and (3) while for PIH the hashes are required to be equal for similar images, we require that the hashes are randomized and a verification algorithm is given which uses a key to perform the relation check.

*Organization of the paper.* In Section 2, we formally define the notion of Relational Hash and its desired security properties. In Section 3, we construct a Relational Hash for linearity over $\mathbb{F}_2^n$, with extension to $\mathbb{F}_p^n$. In Section 4, we show how to construct a proximity (in terms of hamming distance) Relational Hash using a linear Relational Hash and a linear error correcting code. In Section 5, we describe relations among notions of security for constructing Relational Hashes for various relations. Standard hardness assumptions are summarized in Appendix A.

*Notations.* We denote a sequence $x_j, \cdots, x_k$ as $\langle x_i \rangle_{i=j}^k$. We treat $\mathbb{F}_p^n$ as an $\mathbb{F}_p$ vector space and write $x \in \mathbb{F}_p^n$ also as $\langle x_i \rangle_{i=1}^n$. Group elements are written in bold font: $\mathbf{g}$, $\mathbf{f}$. The security parameter is denoted as $\lambda$.

## 2 Relational Hash

The concept of *Relational Hash* is an extension of regular probabilistic hash functions. In this work, we only consider 3-tuple relations. Suppose $R \subseteq X \times Y \times Z$ be a 3-tuple relation, that we are interested in. We abuse the notation a bit, and often use the equivalent functional notation $R : X \times Y \times Z \to \{0, 1\}$. The Relational Hash for the relation $R$, will specify two hash algorithms $\textsc{Hash}_1$ and $\textsc{Hash}_2$ which will output the hash values $\textsc{Hash}_1(x)$ and $\textsc{Hash}_2(y)$ for any $x \in X$ and $y \in Y$. Any Relational Hash must also specify a verification algorithm $\textsc{Verify}$, which will take $\textsc{Hash}_1(x)$, $\textsc{Hash}_2(y)$ and any $z \in Z$ as input and output $R(x, y, z)$. Formally, we define the notion of Relational Hash as follows.

**Definition 1 (Relational Hash).** *Let $\{R_\lambda\}_{\lambda \in \mathbb{N}}$ be a relation ensemble defined over set ensembles $\{X_\lambda\}_{\lambda \in \mathbb{N}}$, $\{Y_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{Z_\lambda\}_{\lambda \in \mathbb{N}}$ such that $R_\lambda \subseteq X_\lambda \times Y_\lambda \times Z_\lambda$. A Relational Hash for $\{R_\lambda\}_{\lambda \in \mathbb{N}}$ consists of four efficient algorithms:*

- *A randomized key generation algorithm:* $\textsc{KeyGen}(1^\lambda)$ *outputs key pk from key space $K_\lambda$.*
- *The hash algorithm of first type (possibly randomized):* $\textsc{Hash}_1 : K_\lambda \times X_\lambda \to \textsc{RangeX}_\lambda$, *here* $\textsc{RangeX}_\lambda$ *denotes the range of* $\textsc{Hash}_1$ *for security parameter $\lambda$.*
- *The hash algorithm of second type (possibly randomized):* $\textsc{Hash}_2 : K_\lambda \times Y_\lambda \to \textsc{RangeY}_\lambda$, *here* $\textsc{RangeY}_\lambda$ *denotes the range of* $\textsc{Hash}_2$ *for security parameter $\lambda$.*
- *The deterministic verification algorithm:*
  $\textsc{Verify} : K_\lambda \times \textsc{RangeX}_\lambda \times \textsc{RangeY}_\lambda \times Z_\lambda \to \{0, 1\}$.

Treating the third parameter $z$ differently from the first two might strike as odd. Our reason behind the choice of this asymmetric definition is to convey the intention that we are not trying to hide $z$ and that the verifier or attacker can choose the value of $z$ to test relations.

In the rest of the paper we will drop the subscript $\lambda$ for simplicity and it will be implicitly assumed in the algorithm descriptions. Often, we will also denote the 1 output of $\textsc{Verify}$ as $\textsc{Accept}$, and the 0 output as $\textsc{Reject}$. The definition of Relational Hashing consists of two requirements: *Correctness* and *Security* (or *Secrecy*).

*Correctness:* Informally speaking, the correctness condition is, if an honest party evaluates $\textsc{Verify}(\textsc{Hash}_1(pk, x), \textsc{Hash}_2(pk, y), z)$ for some key $pk$ which is the output of $\textsc{KeyGen}$ and any $(x, y, z) \in X \times Y \times Z$, the output can differ from $R(x, y, z)$ only with negligible probability (the probability is calculated over the internal randomness of $\textsc{KeyGen}$, $\textsc{Hash}_1$ and $\textsc{Hash}_2$). Formally,

**Definition 2 (Relational Hash - Correctness).** *A* Relational Hash *scheme* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *for a relation $R \subseteq X \times Y \times Z$ satisfies correctness if the following holds for all $(x, y, z) \subseteq X \times Y \times Z$:*

$$\Pr\begin{bmatrix} pk \leftarrow \textsc{KeyGen}(1^\lambda) & \\ hx \leftarrow \textsc{Hash}_1(pk, x) : & \textsc{Verify}(pk, hx, hy, z) \equiv R(x, y, z) \\ hy \leftarrow \textsc{Hash}_2(pk, y) & \end{bmatrix} \approx 1$$

*Security:* The notion of security for a Relational Hash will depend on the context where the Relational Hash is going to be used and also on the a priori information available to the adversary. Recall that for a regular hash function one of the weakest form of security is one-wayness. We will consider Probabilistic Polynomial Time (PPT) adversaries for our security definitions.

**Definition 3 (Security of Relational Hash - One-way).** *Let $\mathcal{X}$ and $\mathcal{Y}$ be (independent) probability distributions over $X$ and $Y$. We define a Relational Hash scheme* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *to be* one-way *secure for the probability distributions $\mathcal{X}$ and $\mathcal{Y}$, if the following hold:*

- *$pk \leftarrow \textsc{KeyGen}(1^\lambda)$, $x \leftarrow \mathcal{X}$, $y \leftarrow \mathcal{Y}$, $hx \leftarrow \textsc{Hash}_1(pk, x)$, $hy \leftarrow \textsc{Hash}_2(pk, y)$*
- *For any PPT adversary $A_1$, there exists a negligible function $\texttt{negl}()$, such that $\Pr[A_1(pk, hx) = x] < \texttt{negl}(\lambda)$.*
- *For any PPT adversary $A_2$, there exists a negligible function $\texttt{negl}()$, such that $\Pr[A_2(pk, hy) = y] < \texttt{negl}(\lambda)$.*

*Here the probabilities are calculated over the internal randomness of* $\textsc{KeyGen}$, $\textsc{Hash}_1$ *and* $\textsc{Hash}_2$, *internal randomness of the adversarial algorithms $A_1$ and $A_2$ as well as the probability distributions $\mathcal{X}$ and $\mathcal{Y}$.*

The above definition captures the security notion in case the adversary has access to either type 1 or type 2 hash values. We observe that if the distributions $\mathcal{X}$ and $\mathcal{Y}$ remain independent, Relational Hash still remains one-way secure, even if the adversary has access to both type of hash values. However for correlated $x$ and $y$, sampled from a joint probability distribution $\Psi$ over $X \times Y$, the previous security notion does not provide sufficient security guarantee when the attacker has access to both types of hash values. For these kind of distributions we define a stronger security notion called *twin one-wayness* as follows.

**Definition 4 (Security of Relational Hash - Twin One-way).** *Let $\Psi$ be a probability distribution over $X \times Y$. We define a Relational Hash scheme* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *to be* twin one-way *secure for the probability distribution $\Psi$, if the following hold:*

- *$pk \leftarrow \textsc{KeyGen}(1^\lambda)$, $(x, y) \leftarrow \Psi$, $hx \leftarrow \textsc{Hash}_1(pk, x)$, $hy \leftarrow \textsc{Hash}_2(pk, y)$*
- *For any PPT adversary $A_1$, there exists a negligible function $\texttt{negl}()$, such that $\Pr[A_1(pk, hx, hy) = x] < \texttt{negl}(\lambda)$.*
- *For any PPT adversary $A_2$, there exists a negligible function $\texttt{negl}()$, such that $\Pr[A_2(pk, hx, hy) = y] < \texttt{negl}(\lambda)$.*

*Here the probabilities are calculated over the internal randomness of* $\textsc{KeyGen}$, $\textsc{Hash}_1$ *and* $\textsc{Hash}_2$, *internal randomness of the adversarial algorithms $A_1$ and $A_2$ as well as the probability distribution $\Psi$.*

Note that the twin one-wayness property is actually a stronger version of correlated input security due to Rosen and Segev [RS09]. We require each coordinate to be one-way, whereas correlated input security requires the input involving all coordinates should be one-way.

*Remark 1.* For our application scenarios: biometric identification and authentication, the twin one-wayness property plays a key role. Intuitively, this guarantees that even if the server has access to both type of hashes coming from biometric templates (possibly generated at different times) of the same person, the template still remains one-way to the server[3].

In this work, we are mostly interested in *sparse* relations (Definition 7). Informally speaking, for a sparse relation $R \subseteq X \times Y \times Z$ and unknown $x$ it is hard to output $y$ and $z$ such that $(x, y, z) \in R$. A Relational Hash scheme is called *unforgeable* if given $hx = \textsc{Hash}_1(pk, x)$ and $pk$ it is hard to output $hy, z$, such that $\textsc{Verify}(pk, hx, hy, z)$ outputs 1. Formally,

**Definition 5 (Security of Relational Hash - Unforgeable).** *Let $\mathcal{X}$ and $\mathcal{Y}$ be (independent) probability distributions over $X$ and $Y$. A Relational Hash scheme* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *is* unforgeable *for the probability distributions $\mathcal{X}$ and $\mathcal{Y}$, if the following holds:*

- $pk \leftarrow \textsc{KeyGen}(1^\lambda)$, $x \leftarrow \mathcal{X}$, $y \leftarrow \mathcal{Y}$, $hx \leftarrow \textsc{Hash}_1(pk, x)$, $hy \leftarrow \textsc{Hash}_2(pk, y)$
- *For any PPT adversary $A_1$, there exists a negligible function* $\mathtt{negl}()$*, such that:*

$$\Pr[(hy', z) \leftarrow A_1(pk, hx) \wedge \textsc{Verify}(pk, hx, hy', z) = 1] < \mathtt{negl}(\lambda)$$

- *For any PPT adversary $A_2$, there exists a negligible function* $\mathtt{negl}()$*, such that:*

$$\Pr[(hx', z) \leftarrow A_2(pk, hy) \wedge \textsc{Verify}(pk, hx', hy, z) = 1] < \mathtt{negl}(\lambda)$$

For Relational Hash functions, the strongest form of security notion is based on oracle simulations. The concept of oracle simulation was introduced in [Can97]. However, over there the author was interested in regular probabilistic hash functions. In case of Relational Hash functions, we want to say that: having $hx = \textsc{Hash}_1(pk, x)$ gives no information on $x$, besides the ability to evaluate the value of $R(x, y, z)$ for any $y$, $z$ chosen from their respective domains. Similarly, $hy = \textsc{Hash}_1(pk, y)$ should not provide any extra information other than the ability to evaluate the value of $R(x, y, z)$ for any $x \in X$ and $z \in Z$. Also, having access to both $hx$ and $hy$, one should be able to only evaluate $R(x, y, z)$ for any $z \in Z$.

For any relation $R \subseteq X \times Y \times Z$ and $x \in X$, $y \in Y$, let $R_x(\cdot, \cdot) : Y \times Z \to \{0, 1\}$, $R_y(\cdot, \cdot) : X \times Z \to \{0, 1\}$ and $R_{x,y}(\cdot) : Z \to \{0, 1\}$ be the oracles defined as follows:

- For any $y' \in Y, z' \in Z, R_x(y', z') = 1$ if and only if $(x, y', z') \in R$.
- For any $x' \in X, z' \in Z, R_y(x', z') = 1$ if and only if $(x', y, z') \in R$.
- For any $z' \in Z, R_{x,y}(z') = 1$ if and only if $(x, y, z') \in R$.

We note that giving oracle access to $R_{x,y}$ on top of $R_x$ and $R_y$ is not superfluous as both $x$ and $y$ are generated and kept unknown from the adversary.

**Definition 6 (Security of Relational Hash - Oracle Simulation).** *Let $\Psi$ be a probability distribution over $X \times Y$. A Relational Hash scheme* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *is said to be oracle simulation secure with respect to the distribution $\Psi$ if for any PPT adversary $C$, there exists a PPT simulator $S$ such that for any predicate $P(\cdot, \cdot, \cdot) : K \times X \times Y \to \{0, 1\}$ (where $K$ is the range of* KeyGen*), there exists a negligible function* $\mathtt{negl}()$*, such that*

$$\left| \begin{array}{c} \Pr[C(pk, \textsc{Hash}_1(pk, x), \textsc{Hash}_2(pk, y)) = P(pk, x, y)] \\ - \Pr[S^{R_x, R_y, R_{x,y}}(pk) = P(pk, x, y)] \end{array} \right| < \mathtt{negl}(\lambda),$$

*where $(x, y) \leftarrow \Psi$ and $pk \leftarrow \textsc{Keygen}(1^\lambda)$.*

---

[3] Strictly speaking, we need a stronger a security criterion, i.e. not only the server should be able to recover exact $x$ or $y$, it should not be able to recover any nearby $x'$ from $x$ or $y$. Theorem 4 in Section 4, in fact guarantees this stronger security notion.

## 3 Relational Hash for Linearity in $\mathbb{F}_2^n$

We now construct a Relational Hash scheme for the domains $X, Y, Z = \mathbb{F}_2^n$ and the relation $R = \{(x, y, z) \mid x + y = z \wedge x, y, z \in \mathbb{F}_2^n\}$.

KeyGen*:* Given the security parameter, bilinear groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are generated of prime order $q$, exponential in the security parameter, and with a bilinear pairing operator $e$. Now we sample generators $\mathbf{g}_0 \leftarrow \mathbb{G}_1$ and $\mathbf{h}_0 \leftarrow \mathbb{G}_2$. Next we sample $\langle a_i \rangle_{i=1}^{n+1}$ and $\langle b_i \rangle_{i=1}^{n+1}$, all randomly from $\mathbb{Z}_q^*$. Define $\mathbf{g}_i = \mathbf{g}_0^{a_i}$ and $\mathbf{h}_i = \mathbf{h}_0^{b_i}$. Now we define the output of KeyGen as $pk := (pk_1, pk_2, pk_R)$, defined as follows:

$$pk_1 := \langle \mathbf{g}_i \rangle_{i=0}^{n+1}, \qquad pk_2 := \langle \mathbf{h}_i \rangle_{i=0}^{n+1}, \qquad pk_R := \sum_{i=1}^{n+1} a_i b_i$$

Hash$_1$*:* Given plaintext $x = \langle x_i \rangle_{i=1}^n \in \mathbb{F}_2^n$ and $pk_1 = \langle \mathbf{g}_i \rangle_{i=0}^{n+1}$, the hash is constructed as follows: Sample a random $r \in \mathbb{Z}_q^*$ and then compute the following:

$$hx := \left( \mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^n, \mathbf{g}_{n+1}^r \right)$$

Hash$_2$*:* Given plaintext $y = \langle y_i \rangle_{i=1}^n \in \mathbb{F}_2^n$ and $pk_2 = \langle \mathbf{h}_i \rangle_{i=0}^{n+1}$, the hash is constructed as follows: Sample a random $s \in \mathbb{Z}_q^*$ and then compute the following:

$$hy := \left( \mathbf{h}_0^s, \left\langle \mathbf{h}_i^{(-1)^{y_i} s} \right\rangle_{i=1}^n, \mathbf{h}_{n+1}^s \right)$$

Verify*:* Given hashes $hx = \langle hx_i \rangle_{i=0}^{n+1}$ and $hy = \langle hy_i \rangle_{i=0}^{n+1}$, the quantity $z = \langle z_i \rangle_{i=1}^n \in \mathbb{F}_2^n$ and $pk_R$, the algorithm Verify checks the following equality:

$$e(hx_0, hy_0)^{pk_R} \stackrel{?}{=} e(hx_{n+1}, hy_{n+1}) \prod_{i=1}^n e(hx_i, hy_i)^{(-1)^{z_i}}$$

*Correctness.* Correctness of the scheme follows from standard algebraic manipulation of pairing operations. Details are given in Appendix B.

*One-wayness.* This Relational Hash can be shown to be one-way secure based on the SXDH assumption, and a new hardness assumption we call Binary Mix DLP. The assumption says if we choose a random $x$ from $\mathbb{F}_2^n$ (for sufficiently large $n$), $n$ random elements $\mathbf{g}_1, \cdots, \mathbf{g}_n$ from group $\mathbb{G}$ then given the product $\prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}}$ it is hard to find any candidate $x$.

**Assumption 1. (Binary Mix DLP) :** *Assuming a generation algorithm $\mathcal{G}$ that outputs a tuple $(n, q, \mathbb{G})$ such that $\mathbb{G}$ is a group of prime order $q$, the Binary Mix DLP assumption asserts that given random elements $\langle \mathbf{g}_i \rangle_{i=1}^n$ from the group $\mathbb{G}$ and $\prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}}$, for a random $x \leftarrow \mathbb{F}_2^n$, it is computationally infeasible to output $y \in \mathbb{F}_2^n$ such that*

$$\prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}} = \prod_{i=1}^n \mathbf{g}_i^{(-1)^{y_i}}.$$

There is an interesting parallel between the Binary Mix DLP assumption and the Discrete Log hardness assumption which may appeal to the appreciation of its hardness at an intuitive level. The Discrete Log problem asks to find $w \in \mathbb{Z}_q^*$ given a random element $\mathbf{g} \in \mathbb{G}$ and $\mathbf{g}^w$. Consider the sequence of elements $\mathbf{g}_1 = \mathbf{g}, \mathbf{g}_2 = \mathbf{g}^2, \cdots, \mathbf{g}_\lambda = \mathbf{g}^{2^\lambda}$, where $\lambda = \lg q$. When we think of the binary expansion of $w = \overline{w_\lambda \cdots w_0}$ and interpret the vector $W = w_\lambda \cdots w_0$ in $\mathbb{F}_2^{\lambda+1}$, then equivalently we are asking for computing $W$, given the product $\prod_{i=0}^\lambda \mathbf{g}_i^{w_i}$.

In the $\mathsf{Binary\ Mix\ DLP}$ problem, the difference is that the $\mathbf{g}_i$'s are independently random and that instead of raising the $\mathbf{g}_i$'s to the powers 0 or 1, we raise them to the powers $\pm 1$. This is, of course, not a formal proof of its hardness. In Appendix E.1, we show that the Binary Mix DLP assumption can actually be reduced to the more standard Random Modular Subset Sum assumption [Lyu05]. As an added assurance, in Appendix E.2, we show that the Binary Mix DLP assumption is also secure in the Generic Group Model [Sho97].

The Binary Mix DLP assumption is similar to [BGG95], where Bellare et al define a hash function to be a subset product of publicly given random group elements based on the bits of the plaintext. In our case, we either use a random group element or its inverse depending on the bit. They achieve reduction from DLP to collision resistance. In contrast, this does not work for one-wayness, as for certain admissible values of $(q, n)$ our function (as also [BGG95]) may turn out to be collision-free.

**Theorem 1.** *The above algorithms* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *constitute a Relational Hash scheme for the relation* $R = \{(x, y, z) \mid x + y = z \wedge x, y, z \in \mathbb{F}_2^n\}$*. The scheme is one-way secure under the SXDH and Binary Mix DLP(Assumption 1) assumptions, when* $x$ *and* $y$ *are sampled uniformly from* $\mathbb{F}_2^n$*.*

The proof is given in Appendix C.

*Twin one-wayness.* Until now, we have shown this Relational Hash is one-way when the adversary has access to only one type of hash values. However, an important scenario to consider is the case when adversary has access to both type of hash values for any $x$ uniformly drawn from $\mathbb{F}_2^n$. The following theorem claims our scheme is indeed twin one-way secure in this case and is proved in Appendix E.3.

**Theorem 2.** *The above algorithms* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *constitute a Relational Hash scheme for the relation* $R = \{(x, y, z) \mid x + y = z \wedge x, y, z \in \mathbb{F}_2^n\}$*. The scheme is twin one-way secure in the generic group model, when* $x$ *is sampled uniformly from* $\mathbb{F}_2^n$ *and* $y = x$*.*

*Unforgeability and Oracle Simulation Security.* In Section 5, we show this Relational Hash is in fact a 2-value perfectly one-way function, albeit under a stronger hardness assumption. By Theorem 8 from Section 5, that will imply this Relational Hash construction is also unforgeable and oracle simulation secure.

*Remark 2.* This linear Relational Hash construction is weakly homomorphic, in the sense that, given

$$\textsc{Hash}_2(y) = (hy_0, \langle hy_i \rangle_{i=1}^n, hy_{n+1}) = \left(\mathbf{h}_0^s, \left\langle \mathbf{h}_i^{(-1)^{y_i} s} \right\rangle_{i=1}^n, \mathbf{h}_{n+1}^s\right),$$

it is easy to construct

$$\textsc{Hash}_2(y + t) = \left(hy_0, \left\langle hy_i^{(-1)^{t_i}} \right\rangle_{i=1}^n, hy_{n+1}\right) = \left(\mathbf{h}_0^s, \left\langle \mathbf{h}_i^{(-1)^{y_i + t_i} s} \right\rangle_{i=1}^n, \mathbf{h}_{n+1}^s\right)$$

for any $t \in \mathbb{F}_2^n$. $\textsc{Hash}_1$ is also homomorphic in a similar manner. However, this does not really refute any of our security claims. In fact, in next section we will see this linear homomorphism gives us strong security guarantee for relation hash construction for hamming proximity (Theorem 4).

*Remark 3.* Theorem 2 and Remark 2 imply that given $\textsc{Hash}_1(x)$, $\textsc{Hash}_2(y)$ and $x + y$ it is hard to output either of $x$ or $y$, for uniformly sampled $x$ and $y$ from $\mathbb{F}_2^n$.

**Relational Hash for Linearity in $\mathbb{F}_p^n$:** For any prime $p$, we can choose the order $q$ of the bilinear groups to be exponential in the security parameters as well as equal to 1 (mod $p$). This means the group $\mathbb{Z}_q^*$ has a subgroup $\mathbb{J}_p$ of prime order $p$. Let $\omega$ be an arbitrary generator of $\mathbb{J}_p$. We can publish this arbitrary generator as part of the public key. For $\textsc{Hash}_1$ evaluation (similarly in $\textsc{Hash}_2$), we can simply calculate $hx_i$ as $\mathbf{g}_i^{\omega^{x_i} r}$ (instead of $\mathbf{g}_i^{(-1)^{x_i} r}$). Similarly during verification, instead of checking $e(hx_0, hy_0)^{pk_R} \stackrel{?}{=} e(hx_{n+1}, hy_{n+1}) \prod_{i=1}^n e(hx_i, hy_i)^{(-1)^{z_i}}$, we can just check $e(hx_0, hy_0)^{pk_R} \stackrel{?}{=} e(hx_{n+1}, hy_{n+1}) \prod_{i=1}^n e(hx_i, hy_i)^{\omega^{-z_i}}$. We provide the details in Appendix D.

## 4 Relational Hash for Hamming Proximity

In this section we construct a Relational Hash for the domains $X, Y = \mathbb{F}_2^n$ and the relation[4] $R_\delta = \{(x, y) \mid \texttt{dist}(x, y) \leq \delta \wedge x, y \in \mathbb{F}_2^n\}$, where dist is the hamming distance and $\delta$ is a positive integer less than $n$. Specifically, we construct a Relational Hash for proximity from a family of binary $(n, k, d)$ linear error correcting codes (ECC) $\mathcal{C}$ and a Relational Hash for linearity in $\mathbb{F}_2^k$: (KeyGenLinear, HashLinear$_1$, HashLinear$_2$, VerifyLinear).

For any $C \in \mathcal{C}$, Encode and Decode are the encoding and decoding algorithms of the $(n, k, d)$ error correcting code $C$. For any $x \in \mathbb{F}_2^n$, $\texttt{weight}(x)$ is the usual *hamming* weight of $x$, denoting the number of one's in the binary representation of $x$. For any error vector $e \in \mathbb{F}_2^n$, with $\texttt{weight}(e) \leq d/2$ and $m \in \mathbb{F}_2^k$ we have,

$$\textsc{Decode}(\textsc{Encode}(m) + e) = m.$$

If $\texttt{weight}(e) > d/2$, the decoding algorithm Decode is allowed to return $\perp$.

KeyGen*:* Given the security parameter, choose a binary $(n, k, 2\delta + 1)$ linear error correcting code $C$, where $k$ is of the order of the security parameter. Run KeyGenLinear and let $pk_{lin}$ be its output. Publish,

$$pk := (\textsc{Encode}, \textsc{Decode}, pk_{lin})$$

Hash$_1$*:* Given plaintext $x \in \mathbb{F}_2^n$ and $pk = (\textsc{Encode}, \textsc{Decode}, pk_{lin})$, the hash value is constructed as follows: Sample a random $r \leftarrow \mathbb{F}_2^k$ and then compute the following:

$$hx_1 := x + \textsc{Encode}(r)$$
$$hx_2 := \textsc{HashLinear}_1(pk_{lin}, r)$$

Publish the final hash value $hx := (hx_1, hx_2)$.

Hash$_2$ is defined similarly.

Verify*:* Given the hash values $hx = (hx_1, hx_2)$, $hy = (hy_1, hy_2)$ and $pk = (\textsc{Encode}, \textsc{Decode}, pk_{lin})$ verification is done as follows.

- Recover $z$ as $z := \textsc{Decode}(hx_1 + hy_1)$.
- Output Reject if Decode returns $\perp$ or $\texttt{dist}(\textsc{Encode}(z), hx_1 + hy_1) > \delta$
- Output $\textsc{VerifyLinear}(pk_{lin}, hx_2, hy_2, z)$.

**Theorem 3.** *The above algorithms* (KeyGen, Hash$_1$, Hash$_2$, Verify) *constitute a Relational Hash for the relation* $R_\delta = \{(x, y) \mid \texttt{dist}(x, y) \leq \delta \wedge x, y \in \mathbb{F}_2^n\}$*. The scheme is one-way secure with respect to the uniform distributions on* $\mathbb{F}_2^n$ *if the linear Relational Hash is a one-way secure with respect to the uniform distributions on* $\mathbb{F}_2^k$*. The scheme is unforgeable for the uniform distributions on* $\mathbb{F}_2^n$ *if the linear Relational Hash is unforgeable with respect to the uniform distributions on* $\mathbb{F}_2^k$*.*

The proof is given in Appendix F.

[4] Note that Relational Hash is defined over 3-tuple relations (Definition 2). However, here proximity encryption is defined over 2-tuple relations. 2-tuple relations can be regarded as special cases of 3-tuple relations, where the third entry does not matter. E.g. the relation $R'_\delta \subseteq \mathbb{F}_2^n \times \mathbb{F}_2^n \times Z$ (where $Z$ is any non empty domain) and $(x, y, z) \in R'_\delta$ if and only if $(x, y) \in R_\delta$.

*Twin one-wayness.* For our target application scenarios (biometric identification / authentication), we need a slightly stronger security property compared to the Twin one-wayness as defined in Definition 4. We only consider a passive adversary looking at the communication transcripts between the entities. Consideration of active adversaries would require an additional challenge-response mechanism which we do not develop in this paper. In particular, we should show that if an attacker has access to $\textsc{Hash}_1(x)$ and a number of samples of $\textsc{Hash}_2(y_i)$ (where $x$ and the $y_i$'s are biometric templates generated by same individual), the attacker cannot output any other biometric template $z$ near to $x$. If we assume that every individual's biometric template has full entropy we can model the scenario as follows:

$$x \leftarrow \mathbb{F}_2^n,\ y_i = x + e_i,$$

where the $e_i$'s are sampled from some known noise distribution $\Xi$, such that with high probability we have $\texttt{weight}(e_i) \le \delta$. We now show that, given $\textsc{Hash}_1(x)$ and any number of samples[5] $\textsc{Hash}_2(y_i)$, the attacker cannot output $z$, such that $\texttt{dist}(x, z) \le \delta$. The proof, which is a reduction to twin one-wayness of the linear Relational Hash is given in Appendix F.

**Theorem 4.** *If the above Relational Hash for $R_\delta = \{(x, y) \mid \texttt{dist}(x, y) \le \delta \wedge x, y \in \mathbb{F}_2^n\}$, is instantiated by the* twin one-way *secure linear Relational Hash in Section 3, then for a random $x \leftarrow \mathbb{F}_2^n$ and for any polynomially bounded number of error samples $e_1, \cdots, e_t \leftarrow \Xi$, given $(\textsc{Hash}_1(x), \textsc{Hash}_2(x + e_1), \cdots, \textsc{Hash}_2(x + e_t))$ it is hard to output $x' \in \mathbb{F}_2^n$ such that $\texttt{dist}(x', x) \le \delta$.*

**Privacy Preserving Biometric Authentication Scheme.** Suppose we have a biometric authentication scheme, where during registration phase a particular user generates a biometric template $x \in \{0, 1\}^n$ and sends it to the server. During authentication phase the user generates a new biometric template $y \in \{0, 1\}^n$ and sends $y$ to server. The server authenticates the user if $\texttt{dist}(x, y) \le \delta$. The drawback of this scheme is the lack of template privacy. However, if we have a Relational Hash $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ for the relation $R_\delta = \{(x, y) \mid \texttt{dist}(x, y) \le \delta \wedge x, y \in \mathbb{F}_2^n\}$, we readily get a privacy preserving biometric authentication scheme as follows: 1. A trusted third party runs $\textsc{KeyGen}$ and publishes $pk \leftarrow \textsc{KeyGen}$. 2. During Registration, the client generates biometric template $x \in \{0, 1\}^n$ and sends $hx = \textsc{Hash}_1(pk, x)$ to the server. 3. During Authentication, the client generates biometric template $y \in \{0, 1\}^n$ and sends $hy = \textsc{Hash}_2(pk, y)$ to the server. 4. The server authenticates the client iff $\textsc{Verify}(pk, hx, hy)$ returns $\textsc{Accept}$.

If we assume that the biometric templates of individuals follow uniform distribution over $\{0, 1\}^n$, then Theorem 3 would imply that the server can never recover the original biometric template $x$. Moreover, the unforgeability property guarantees that even if the server's database gets leaked to an attacker then also the attacker cannot come up with a forged $hy'$, which would authenticate the attacker. Theorem 4 will guarantee that even with access to the registered hash and several authentication transcripts from the same individual, the biometric template will remain private to the server.

In spite of these strong guarantees there is a significant drawback of our privacy preserving authentication scheme. One basic premise of this scheme is that the biometric template $x$ comes from a uniform distribution over $\{0, 1\}^n$. From a practical point of view this is really a strong assumption. One interesting open problem in this direction is whether we can build a privacy preserving biometric authentication scheme when $x$ comes from a distribution with high min-entropy which is not necessarily uniform.

## 5 Relation among Notions of Security for Relational Hashes

In Section 2 we introduced three natural definitions of security for Relational Hash functions: one-wayness, unforgeability and oracle simulation security. In this section we define the notion of *sparse* and *biased* relations. We show, if a Relational Hash function is unforgeable, that implies the relation must be sparse. Following [CMR98], we extend the notion of *2-value Perfectly One-Way* (2-POW) function. We show if a Relational Hash function is 2-POW, then the relation must be biased. We also show that the 2-POW

[5] Limited only by the time complexity of the attacker.

property is actually a sufficient condition for oracle simulation security, as well as unforgeability (when the relation is sparse).

We begin by asking the question: What kind of relations can support the existence of an unforgeable Relational Hash? It is easy to see that certain relations cannot support unforgeability. Take, for example, the relation $R(x, y, z)$, where $x, y \in \mathbb{F}_2^n$ and $z \in \mathbb{F}_2$ which returns 1 iff the parity of $x + y$ is equal to the bit $z$. One cannot construct an unforgeable hash for this relation because given the type 1 hash of a random $x$, it is easy to construct a type 2 hash of a $y$ such that the relational verification outputs 1, without knowing $x$: We just pick an arbitrary $y$, compute a type 2 hash of the arbitrary $y$ and verify with the relational key with the type 1 hash of $x$ for both $z$ values 0 and 1.

So the intuitive property of relations supporting unforgeability is that without knowing $x$, it should be hard to come up with $(y, z)$, such that $R(x, y, z)$ holds. We formalize this intuition below in defining *sparse* relations.

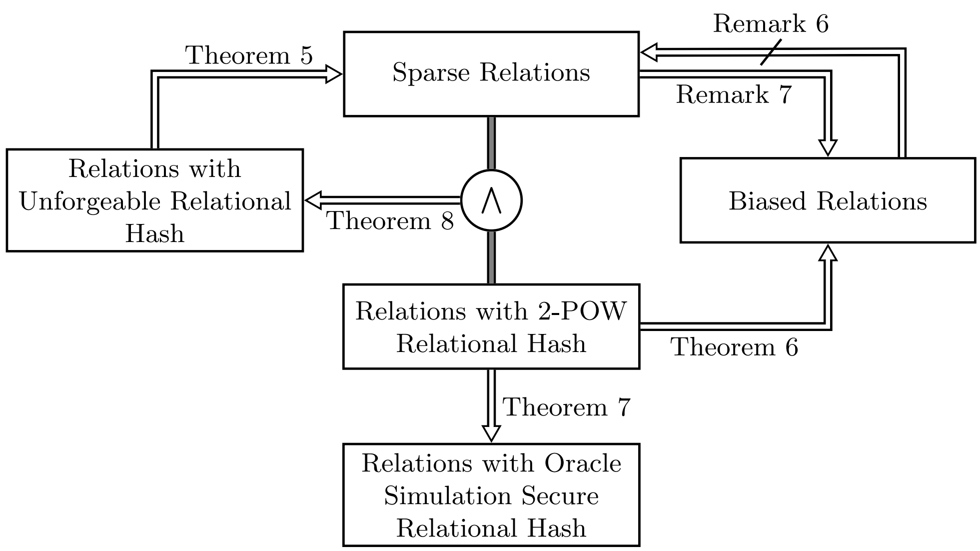


**Fig. 1.** Relationship among Types of Relations. Arrowhead indicates direction of implication. Strike on an arrow indicates the existence of a counter-example.

**Definition 7.** *A relation* $R \subseteq X \times Y \times Z$ *is called a* sparse relation *in the first co-ordinate with respect to a probability distribution* $\mathcal{X}$ *over* $X$*, if for all PPTs A:*

$$\Pr[x \leftarrow \mathcal{X}, (y, z) \leftarrow \mathrm{A}(\lambda) : (x, y, z) \in R] < \mathtt{negl}(\lambda)$$

Similarly, we can define a *sparse relation* in the second co-ordinate with respect to a probability distribution $\mathcal{Y}$ over $Y$. A relation $R \subseteq X \times Y \times Z$ is called a *sparse relation* with respect to probability distributions $\mathcal{X}$ over $X$ and $\mathcal{Y}$ over $Y$, if it is a sparse relation in first coordinate with respect to $\mathcal{X}$, as well as a sparse relation in second coordinate with respect to $\mathcal{Y}$.

*Remark 4.* Similar to Section 2, the definitions given in this sections are actually defined with respect to ensemble of probability distributions $\mathcal{X}_\lambda, \mathcal{Y}_\lambda, \mathcal{K}_\lambda$, ensemble of sets $X_\lambda, Y_\lambda, Z_\lambda, K_\lambda$ and ensemble of relation $R_\lambda$. However, for simplicity we drop the subscript $\lambda$.

Now, we show if a Relational Hash function is unforgeable, that implies the relation must be sparse.

**Theorem 5.** *If a Relational Hash scheme* (KeyGen, Hash$_1$, Hash$_2$, Verify) *for a relation* $R$ *is unforgeable for probability distributions* $\mathcal{X}$ *over* $X$ *and* $\mathcal{Y}$ *over* $Y$*, then the relation* $R$ *is sparse with respect to* $\mathcal{X}$ *and* $\mathcal{Y}$*.*

*Proof.* Suppose, the relation $R$ is not sparse over first coordinate, and there exists an PPT attacker A such that $\Pr[x \leftarrow \mathcal{X}, (y,z) \leftarrow \mathrm{A}(\lambda) : (x,y,z) \in R]$ is non-negligible. Now, given an unforgeability challenge $(pk, cx)$, such that $pk \leftarrow \textsc{KeyGen}(1^\lambda)$ and $cx \leftarrow \textsc{Hash}_1(pk, x)$ for some $x \leftarrow \mathcal{X}$; we can just get $(y,z) \leftarrow \mathrm{A}(\lambda)$ and output $(\textsc{Hash}_2(pk, y), z)$. From the correctness of the Relational Hash function, it follows that this output is a valid forgery with non-negligible probability. □

Following [CMR98], we recall the definition of 2-value perfectly one-way (POW) functions. Intuitively, this property states that the distribution of a two probabilistic hashes of the same value is computationally indistinguishable from the distribution of probabilistic hashes of two independent values. This is a useful property, because if we can show a Relational Hash function is 2-POW, we show that it would immediately imply the Relational Hash function is oracle simulation secure, as well as unforgeable (if the relation is sparse).

**Definition 8 (2-value Perfectly One-Way function).** *Let $\mathcal{X}$ be a probability distribution over $X$. Let $H = \{h_k\}_{k \in K}$ be a keyed probabilistic function family with domain $X$ and randomness space $U$, where the key $k$ gets sampled from a probability distribution $\mathcal{K}$ over $K$. $H$ is 2-value perfectly one-way (POW) with respect to $\mathcal{X}$ and $\mathcal{K}$ if for any PPT distinguisher $D$,*

$$\left| \begin{array}{c} \Pr[D(k, h_k(x, r_1), h_k(x, r_2)) = 1] \\ - \Pr[D(k, h_k(x_1, r_1), h_k(x_2, r_2)) = 1] \end{array} \right| < \mathtt{negl}(\lambda),$$

*where $x, x_1, x_2$ are drawn independently from $\mathcal{X}$, $k$ is drawn from $\mathcal{K}$ and $r_1, r_2$ are generated uniformly at random from the randomness space $U$.*

*Remark 5.* In [CMR98], the key $k$ was universally quantified, and the function family $H$ was called 2-POW if the inequality was true for all $k \in K$. However, for our purpose it is sufficient if we consider random $k$ coming from the distribution $\mathcal{K}$ (or $\textsc{KeyGen}$).

Now we ask what kind of relations can support the existence of 2-POW Relational Hashes? Intuitively, we require that it should be hard to distinguish two distinct samples $x$ and $w$ from the distribution $\mathcal{X}$ by testing relations with a $(y,z)$ tuple which is efficiently computable without knowing the samples. That is we should have $R(x,y,z)$ and $R(w,y,z)$ come out equal most of the time. This intuition is formalized in the following definition of *biased* relations.

**Definition 9.** *A relation $R \subseteq X \times Y \times Z$ is called a* biased relation *in the first co-ordinate with respect to a probability distribution $\mathcal{X}$ over $X$, if for all PPTs A:*

$$\Pr[x, w \leftarrow \mathcal{X}, (y,z) \leftarrow \mathrm{A}(\lambda) : R(x,y,z) \neq R(w,y,z)] < \mathtt{negl}(\lambda)$$

Similarly, we can define a *biased relation* in the second co-ordinate with respect to a probability distribution $\mathcal{Y}$ over $Y$. A relation $R \subseteq X \times Y \times Z$ is called a *biased relation* with respect to independent probability distributions $\mathcal{X}$ over $X$ and $\mathcal{Y}$ over $Y$, if it is a biased relation in first coordinate with respect to $\mathcal{X}$, as well as a biased relation in second coordinate with respect to $\mathcal{Y}$.

*Remark 6.* We observe that if a relation $R$ is biased, then its complement $\bar{R}$ is also biased. Now one might begin to think that maybe for a biased relation $R$, either $R$ or $\bar{R}$ is sparse. However, the following counterexample shows that this is not the case. Consider the relation $R(x,y,z)$ which outputs the first bit of $y$. This is a biased relation, but neither $R$, nor its complement $\bar{R}$ is sparse.

*Remark 7.* The other direction is actually an implication, that is, if a relation $R$ is sparse then it is also biased. The proof intuition is as follows: Given an algorithm A breaking the biased-ness of $R$, we construct an algorithm breaking the sparse-ness of $R$. Let A output $(y,z)$, such that with probability $p$ over the choice of $x \leftarrow \mathcal{X}$, $R(x,y,z) = 1$ and therefore with probability $1-p$, $R(x,y,z) = 0$. The probability of breaking the biased-ness of $R$ is thus $2p(1-p)$ which should be non-negligible. Hence $p$ should be non-negligible. Now observe that $p$ is the probability of breaking the sparse-ness of $R$.

Now, we show if a Relational Hash is 2-POW, then the relation must be biased.

**Theorem 6.** *For a Relational Hash scheme* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *for a relation R, if* $\textsc{Hash}_1$ *is 2-value Perfectly One-Way with respect to* $\mathcal{X}$ *and* $\textsc{KeyGen}$*, then R is a biased relation in the 1st co-ordinate with respect to* $\mathcal{X}$*.*

*Proof.* We are given that,

$$\forall\ PPT\ D : \left| \begin{array}{c} \Pr[D(k, \textsc{Hash}_1(k, x, r_1), \textsc{Hash}_1(k, x, r_2)) = 1] \\ -\Pr[D(k, \textsc{Hash}_1(k, x_1, r_1), \textsc{Hash}_1(k, x_2, r_2)) = 1] \end{array} \right| < \texttt{negl}(\lambda)$$

Suppose $R$ is not a biased relation in the 1st co-ordinate. Then, there exists an efficient algorithm A, which outputs $(y, z) \in Y \times Z$, such that $\Pr[x \leftarrow X, (y, z) \leftarrow \mathrm{A}(\lambda) : R(x, y, z) \neq R(w, y, z)]$ is non-negligible in the security parameter. So now given $(k, \textsc{Hash}_1(k, x, r_1), \textsc{Hash}_1(k, w, r_2))$, we generate $(y, z) \leftarrow \mathrm{A}(\lambda)$, compute $\textsc{Hash}_2(k, y, r')$ and then compute $\textsc{Verify}(k, \textsc{Hash}_1(k, x, r_1), \textsc{Hash}_2(k, y, r'), z)$ and $\textsc{Verify}(k, \textsc{Hash}_1(k, w, r_2), \textsc{Hash}_2(k, y, r'), z)$. By the correctness of the Relational Hash scheme, these boolean results are $R(x, y, z)$ and $R(w, y, z)$ respectively. In the case $R(x, y, z) = R(w, y, z)$, the distinguisher $D$ outputs 1, else 0. By the non-sparseness of $R$, $D$ will have a non-negligible chance of distinguishing the distributions. Hence we get a contradiction. □

Theorem 7, stated below, shows that if a Relational Hash is 2-POW, then it is also oracle simulation secure. The proof is given in Appendix G.

**Theorem 7.** *For a Relational Hash scheme* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$*, if the algorithms* $\textsc{Hash}_1$ *and* $\textsc{Hash}_2$ *are individually 2-value Perfectly One-Way for distributions* $(\mathcal{X}, \textsc{KeyGen})$ *and* $(\mathcal{Y}, \textsc{KeyGen})$ *respectively, then the Relational Hash scheme is Oracle Simulation Secure for the distribution* $\mathcal{X} \times \mathcal{Y}$*. Formally, for all PPT C, there exists a PPT S, such that:*

$$\left| \begin{array}{c} \Pr[C(pk, \textsc{Hash}_1(pk, x), \textsc{Hash}_2(pk, y)) = P(pk, x, y)] \\ -\Pr[S^{R_x, R_y, R_{x,y}}(pk) = P(pk, x, y)] \end{array} \right| < \texttt{negl}(\lambda),$$

*where* $pk \leftarrow \textsc{KeyGen}$*,* $x \leftarrow \mathcal{X}$*,* $y \leftarrow \mathcal{Y}$*.*

Finally, we show that if a Relational Hash is 2-POW as well as sparse, then it must be unforgeable.

**Theorem 8.** *If* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *is a Relational Hash scheme for a sparse relation R with respect to independent probability distributions* $\mathcal{X}$ *and* $\mathcal{Y}$ *and* $\textsc{Hash}_1$ *(*$\textsc{Hash}_2$*) is 2-value Perfectly One-Way for distribution* $\mathcal{X}$ *(*$\mathcal{Y}$*) and* $\textsc{KeyGen}$*, then the Relational Hash scheme is unforgeable for the distribution* $\mathcal{X}$ *(*$\mathcal{Y}$*).*

*Proof.* Assume that the scheme is not unforgeable. This means that given $(pk, \textsc{Hash}_1(pk, x, r))$ for $x \leftarrow \mathcal{X}$, there is an attacker A, which outputs $\textsc{Hash}_2(pk, y, s)$ and $z$, such that $R(x, y, z) = 1$, with non-negligible probability. Using A, we now build an attacker B which distinguishes the distributions $(pk, \textsc{Hash}_1(pk, x, r_1), \textsc{Hash}_1(pk, x, r_2))$ and $(pk, \textsc{Hash}_1(pk, x, r_1), \textsc{Hash}_1(pk, x', r_2))$ with non-negligible probability. Given $(pk, \textsc{Hash}_1(pk, x, r_1), \textsc{Hash}_1(pk, w, r_2))$, B sends $\textsc{Hash}_1(pk, x, r_1)$ to A. With non-negligible probability A outputs $\textsc{Hash}_2(pk, y, s)$ and $z$, such that $R(x, y, z) = 1$. Now since $R$ is a sparse relation, if $w \neq x$, then with non-negligible probability $R(w, y, z) = 0$, whereas if $w = x$, then $R(w, y, z) = 1$. Now $R(w, y, z)$ can be efficiently evaluated by computing $\textsc{Verify}(pk, \textsc{Hash}_1(pk, w, r_2), \textsc{Hash}_2(pk, y, s), z)$. Thus, B will have a non-negligible probability of breaking the 2-value POW security of $\textsc{Hash}_1$. □

**Stronger Security Properties for the Relational Hash Constructions.** In Theorem 9, we show that the Relational Hash construction for linearity over $\mathbb{F}_2^n$ from Section 3 is actually a 2-value perfectly one-way function. This property is based on a stronger hardness assumption called Decisional Binary Mix(Assumption 2). In Appendix E.2(Theorem 13) we show that this assumption holds in the Generic Group Model [Sho97]. One can easily verify that the linearity relation over $\mathbb{F}_2^n$, $R = \{(r, s, z) \mid r + s = z \wedge r, s, z \in \mathbb{F}_2^n\}$ is actually a sparse relation with respect to uniform distributions over $\mathbb{F}_2^n$. Hence, by Theorem 7 and Theorem 8 we get that the Relational Hash construction from Section 3 is actually oracle simulation secure as well as unforgeable with respect to the independent uniform distributions over $\mathbb{F}_2^n$.

**Assumption 2 (Decisional Binary Mix).** *Assuming a generation algorithm $\mathcal{G}$ that outputs a tuple $(n, q, \mathbb{G})$ such that $\mathbb{G}$ is a group of prime order $q$, the* Decisional Binary Mix *assumption asserts that for random $x, y \leftarrow \mathbb{F}_2^n$, given random elements $\langle \mathbf{g}_i \rangle_{i=1}^n$, $\langle \mathbf{f}_i \rangle_{i=1}^n$ from the group $\mathbb{G}$ it is hard to distinguish the following distributions:*

$$\left( \prod_{i=1}^{n} \mathbf{g}_i^{(-1)^{x_i}}, \prod_{i=1}^{n} \mathbf{f}_i^{(-1)^{x_i}} \right) \text{ and } \left( \prod_{i=1}^{n} \mathbf{g}_i^{(-1)^{x_i}}, \prod_{i=1}^{n} \mathbf{f}_i^{(-1)^{y_i}} \right).$$

**Theorem 9.** *The algorithms* (KeyGen, $\text{Hash}_1$, Verify) *in Section 3 constitute a 2-value Perfectly One Way Function for the uniform distribution on $\mathbb{F}_2^n$, under the* Decisional Binary Mix *(Assumption 2) and* DDH *assumptions.*

The proof is given in Appendix H.

**On Stronger Security Properties for the Proximity Hash Constructions.** We observe that our proximity hash construction is not 2-POW secure. This is readily seen by considering the first component of the proximity hash, which is $x+c$, where $x$ is the plaintext and $c$ is a codeword. Two independent hashes of $x$ will have first components $x+c$ and $x+c'$, and therefore adding them will lead to $c+c'$, which is a codeword. However for the hash of an independently randomly generated $y$, the first component will be $y+c''$. If we add the first components we get $x+y+c+c''$, which is unlikely to be a codeword. Therefore there is an efficient distinguisher for the 2-POW distributions. Our construction is also not Oracle Simulation secure, because it reveals the syndrome of the plaintext with respect to the ECC used - this is more information than what the simulation world can provide. We leave it as an open problem to construct 2-POW and Oracle Simulation secure Relational Hashes for proximity.

## References


BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, August 2004.

BGG95. Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography and application to virus protection. In *27th ACM STOC*, pages 45–56. ACM Press, May / June 1995.

Boy04. Xavier Boyen. Reusable cryptographic fuzzy extractors. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04*, pages 82–91. ACM Press, October 2004.

Can97. Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 455–469. Springer, August 1997.

CMR98. Ran Canetti, Daniele Micciancio, and Omer Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In *30th ACM STOC*, pages 131–140. ACM Press, May 1998.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

DRS04. Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, May 2004.

DS05. Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 654–663. ACM Press, May 2005.

GGG+14. Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, May 2014.

GGJS13. Shafi Goldwasser, Vipul Goyal, Abhishek Jain, and Amit Sahai. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/727, 2013. `http://eprint.iacr.org/2013/727`.

GKL+13. S. Dov Gordon, Jonathan Katz, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774, 2013. `http://eprint.iacr.org/2013/774`.

JS02. Ari Juels and Madhu Sudan. A fuzzy vault scheme. Cryptology ePrint Archive, Report 2002/093, 2002. `http://eprint.iacr.org/2002/093`.

JW99. Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *ACM CCS 99*, pages 28–36. ACM Press, November 1999.

KVM04. Suleyman Serdar Kozat, Ramarathnam Venkatesan, and Mehmet Kivanç Mihçak. Robust perceptual image hashing via matrix invariants. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 5, pages 3443–3446. IEEE, 2004.

Lyu05. Vadim Lyubashevsky. On random high density subset sums. *Electronic Colloquium on Computational Complexity (ECCC)*, 12(007), 2005.

MR15. Avradip Mandal and Arnab Roy. Relational Hash: Probabilistic Hash for Verifying Relations, Secure against Forgery and More. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015*, volume xxxx of *LNCS*, pages xxx–xxx. Springer, August 2015.

PR12. Omkant Pandey and Yannis Rouselakis. Property preserving symmetric encryption. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 375–391. Springer, April 2012.

RS09. Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 419–436. Springer, March 2009.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, May 1997.

# A Hardness Assumptions

We summarize the standard hardness assumptions used in this paper.

**Assumption 3 (DLP).** *Assuming a generation algorithm $\mathcal{G}$ that outputs a tuple $(q, \mathbb{G}, \mathbf{g})$ such that $\mathbb{G}$ is of prime order $q$ and has generator $g$, the DLP assumption asserts that given $(\mathbf{g}, \mathbf{g}^x)$it is computationally infeasible to output $x$ for random $x \leftarrow \mathbb{Z}_q$. More formally, for all PPT adversaries $A$ there exists a negligible function* $\mathtt{negl}()$ *such that*

$$Pr[(q, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda); x \leftarrow \mathbb{Z}_q : A(\mathbf{g}, \mathbf{g}^x) = x] < \mathtt{negl}(\lambda)$$

**Assumption 4 (DDH [DH76]).** *Assuming a generation algorithm $\mathcal{G}$ that outputs a tuple $(q, \mathbb{G}, \mathbf{g})$ such that $\mathbb{G}$ is of prime order $q$ and has generator $g$, the DDH assumption asserts that it is computationally infeasible to distinguish between $(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^c)$ and $(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^{ab})$ for $a, b, c \leftarrow \mathbb{Z}_q^*$. More formally, for all PPT adversaries $A$ there exists a negligible function* $\mathtt{negl}()$ *such that*

$$\left| \begin{matrix} Pr[(q, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda); a, b, c \leftarrow \mathbb{Z}_q^* : A(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^c) = 1] - \\ Pr[(q, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda); a, b \leftarrow \mathbb{Z}_q^* : A(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^{ab}) = 1] \end{matrix} \right| < \mathtt{negl}(\lambda)$$

**Assumption 5 (SXDH [BBS04]).** *Consider a generation algorithm $\mathcal{G}$ taking the security parameter as input, that outputs a tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}_1, \mathbf{g}_2)$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order $q$ with generators $\mathbf{g}_1, \mathbf{g}_2$ and $e(\mathbf{g}_1, \mathbf{g}_2)$ respectively and which allow an efficiently computable $\mathbb{Z}_q^*$-bilinear pairing map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The Symmetric eXternal decisional Diffie-Hellman (SXDH) assumption asserts that the Decisional Diffie-Hellman (DDH) problem is hard in* both *the groups $\mathbb{G}_1$ and $\mathbb{G}_2$.*

**Assumption 6 (Random Modular Subset Sum [Lyu05]).** *Assuming a generation algorithm $\mathcal{G}$ that outputs a tuple $(n, q)$,where $q$ is prime,the Random Modular Subset Sum assumption asserts that given random elements $\langle a_i \rangle_{i=1}^n$ from the group $\mathbb{Z}_q$ and $c = \sum_{i=1}^n \epsilon_i a_i$ for a random $\epsilon \leftarrow \{0, 1\}^n$, it is hard to output $\eta \in \{0, 1\}^n$ such that*

$$\sum_{i=1}^n \eta_i a_i = c \pmod q.$$

*More formally, for all PPT* A*, there exists a negligible function* $\mathtt{negl}()$ *such that*

$$\Pr \left[ \begin{matrix} (n, q) \leftarrow \mathcal{G}(1^\lambda),\ \langle a_i \rangle_{i=1}^n \leftarrow \mathbb{Z}_q \\ \epsilon \leftarrow \{0, 1\}^n, c = \sum_{i=1}^n \epsilon_i a_i \\ \eta \leftarrow \mathrm{A}(\langle a_i \rangle_{i=1}^n, c) \end{matrix} : \sum_{i=1}^n \eta_i a_i = c \pmod q \right] < \mathtt{negl}(\lambda).$$

## B Correctness of the $\mathbb{F}_2^n$ Linear Relational Hash

For any, $x, y \in \mathbb{F}_2^n$ we have

$$\text{HASH}_1(x) = (hx_0, \langle hx_i \rangle_{i=1}^n, hx_{n+1}) = \left(\mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^n, \mathbf{g}_{n+1}^r\right)$$
$$\text{HASH}_2(y) = (hy_0, \langle hy_i \rangle_{i=1}^n, hy_{n+1}) = \left(\mathbf{h}_0^s, \left\langle \mathbf{h}_i^{(-1)^{y_i} s} \right\rangle_{i=1}^n, \mathbf{h}_{n+1}^s\right)$$

Hence,

$$e(hx_{n+1}, hy_{n+1}) \prod_{i=1}^n e(hx_i, hy_i)^{(-1)^{(x_i+y_i)}} = e(\mathbf{g}_{n+1}^r, \mathbf{h}_{n+1}^s) \prod_{i=1}^n e\left(\mathbf{g}_i^{(-1)^{x_i} r}, \mathbf{h}_i^{(-1)^{y_i} s}\right)^{(-1)^{(x_i+y_i)}}$$
$$= \prod_{i=1}^{n+1} e\left(\mathbf{g}_i^r, \mathbf{h}_i^s\right) = \prod_{i=1}^{n+1} e\left(\mathbf{g}_0^{a_i r}, \mathbf{h}_0^{b_i s}\right) = e\left(\mathbf{g}_0^r, \mathbf{h}_0^s\right)^{\sum_{i=1}^{n+1} a_i b_i} = e(hx_0, hy_0)^{pk_R}$$

This shows that our relational hash scheme correctly verifies tuples of the form $(x, y, x+y)$ for any $x, y \in \mathbb{F}_2^n$.

On the other hand, if the verification equation gets satisfied for some $z \in \mathbb{F}_2^n$, we must have

$$e\left(\mathbf{g}_0^r, \mathbf{h}_0^s\right)^{\sum_{i=1}^{n+1} a_i b_i} = e(\mathbf{g}_{n+1}^r, \mathbf{h}_{n+1}^s) \prod_{i=1}^n e\left(\mathbf{g}_i^{(-1)^{x_i} r}, \mathbf{h}_i^{(-1)^{y_i} s}\right)^{(-1)^{z_i}}$$
$$\implies \sum_{i=1}^n a_i b_i = \sum_{i=1}^n (-1)^{x_i+y_i+z_i} a_i b_i$$

Let $Q \subseteq \{1, \cdots, n\}$ be the set of indices, such that $i \in Q$ if and only if $x_i + y_i \neq z_i$. Now, the above equation reduces to

$$\sum_{i \in Q} a_i b_i = 0.$$

If $x + y \neq z$, then $Q$ is non empty and we can consider a fixed $i^* \in Q$ and we have

$$a_{i^*} b_{i^*} = -\sum_{i \in Q \setminus \{i^*\}} a_i b_i.$$

Now, if we fix $a_i$'s and $b_i$'s for $i \in Q \setminus \{i^*\}$ and consider only the randomness of $a_{i^*}$ and $b_{i^*}$, the above equation holds with probability at most $1/q$ when $\sum_{i \in Q \setminus \{i^*\}} a_i b_i \neq 0$ and with probability at most $2/q$ when $\sum_{i \in Q \setminus \{i^*\}} a_i b_i = 0$. This implies for any tuple $(x, y, z)$ with $z \neq x + y$, the verification equation gets satisfied with probability at most $2/q$. Hence the above algorithms in fact constitute a correct relational hash for linearity over $\mathbb{F}_2^n$. □

## C One-wayness of $\mathbb{F}_2^n$ Linear Relational Hash: Proof of Theorem 1

In this section we go through a sequence of lemmas, leading to proof of Theorem 1.

**Lemma 1. (DDH) :** *For random $\mathbf{g}, \mathbf{h} \leftarrow \mathbb{G}$ and $r \leftarrow \mathbb{Z}_q^*$, the following tuples are computationally indistinguishable under the DDH assumption:*

$$(\mathbf{g}, \mathbf{h}, \mathbf{g}^r, \mathbf{h}^r) \approx_{DDH} (\mathbf{g}, \mathbf{h}, \mathbf{g}^r, \mathbf{h}^{-r}).$$

*Proof.* We have,

$$(\mathbf{g}, \mathbf{h}, \mathbf{g}^r, \mathbf{h}^r) \approx_{DDH} (\mathbf{g}, \mathbf{h}, \mathbf{g}^r, \mathbf{h}^{r'}) \approx_{statistical} (\mathbf{g}, \mathbf{h}, \mathbf{g}^r, \mathbf{h}^{-r'}) \approx_{DDH} (\mathbf{g}, \mathbf{h}, \mathbf{g}^r, \mathbf{h}^{-r}),$$

where $r, r'$ are generated independently randomly from $\mathbb{Z}_q^*$. □

**Lemma 2.** *Under the Binary Mix DLP assumption(Assumption 1), given random elements* $\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^n$ *from a group* $\mathbb{G}$ *and* $\mathbf{g}^r, \mathbf{v} = \prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i} r}, \langle \mathbf{z}_i \rangle_{i=1}^n = \langle \mathbf{g}_i^r \rangle_{i=1}^n$ *for random* $r \leftarrow \mathbb{Z}_q^*$ *and random* $x \leftarrow \mathbb{F}_2^n$ *it is hard to output* $y \in \mathbb{F}_2^n$ *such that*

$$\mathbf{v} = \prod_{i=1}^n \mathbf{z}_i^{(-1)^{y_i}}.$$

*Formally, for all PPT adversaries* $A$ *there exists a negligible function* $\texttt{negl}()$ *such that*

$$\Pr\left[\begin{array}{c} \mathbb{G} \leftarrow \mathcal{G}(1^\lambda); \mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^n \leftarrow \mathbb{G}; r \leftarrow \mathbb{Z}_q^*; x \leftarrow \mathbb{F}_2^n; \\ y \leftarrow A\left(\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^n, \mathbf{g}^r, \mathbf{v} = \prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i} r}, \langle \mathbf{z}_i \rangle_{i=1}^n = \langle \mathbf{g}_i^r \rangle_{i=1}^n\right) \end{array} : \mathbf{v} = \prod_{i=1}^n \mathbf{z}_i^{(-1)^{y_i}}\right] < \texttt{negl}(\lambda)$$

*Proof.* Suppose there exists an adversary $A^*$ for which the above probability is non negligible. We will show, using the adversary $A^*$ we can break Lemma 2 challenge. Let $(\langle \mathbf{g}_i \rangle_{i=1}^n, \mathbf{w} = \prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}})$ be the Binary-Mix-DLP (Assumption 1) challenge. We choose random $\mathbf{g} \leftarrow \mathbb{G}$ and random $r \leftarrow \mathbb{Z}_q^*$. We send $\left(\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^n, \mathbf{g}^r, \mathbf{w}^r, \langle \mathbf{g}_i^r \rangle_{i=1}^n\right)$ to the adversary $A^*$. We publish the output of $A^*$ to the Binary-Mix-DLP challenger. Clearly, the success probability of breaking the Binary-Mix-DLP assumption is same as success probability of the adversary $A^*$. Hence, under Binary-Mix-DLP assumption the success probability of $A^*$ can not be non-negligible. □

**Lemma 3.** *Under the DDH assumption, given random elements* $\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^n$ *from a group* $\mathbb{G}$*, random* $r \leftarrow \mathbb{Z}_q^*$ *and any* $x \in \mathbb{F}_2^n$ *the following tuples are computationally indistinguishable.*

$$\left(\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^n, \mathbf{g}^r, \prod_{i=1}^n \mathbf{g}_i^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^n\right) \approx_{DDH} \left(\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^n, \mathbf{g}^r, \prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i} r}, \langle \mathbf{g}_i^r \rangle_{i=1}^n\right)$$

*Proof.* We define a sequence of games $\langle \mathbf{Game}_j \rangle_{j=1}^n$. We argue, in $\mathbf{Game}_j$, for random $\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^n$ from a group $\mathbb{G}$, random $r \leftarrow \mathbb{Z}_q^*$ and any $x \in \mathbb{F}_2^n$ the following tuples are computationally indistinguishable under DDH assumption

$$\left(\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^n, \mathbf{g}^r, \prod_{i=1}^{j-1} \mathbf{g}_i^{(-1)^{x_i} r} \prod_{i=j}^n \mathbf{g}_i^r, \langle \mathbf{g}_i^r \rangle_{i=1}^{j-1}, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=j}^n\right)$$
$$\approx_{DDH}$$
$$\left(\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^n, \mathbf{g}^r, \prod_{i=1}^{j} \mathbf{g}_i^{(-1)^{x_i} r} \prod_{i=j+1}^n \mathbf{g}_i^r, \langle \mathbf{g}_i^r \rangle_{i=1}^{j}, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=j+1}^n\right).$$

If $x_j = 0$, then the above two distributions are in fact identical. So, we can only consider the case $x_j = 1$. In this case we need to show the following distributions are computationally indistinguishable under DDH.

$$\left(\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^n, \mathbf{g}^r, \mathbf{g}_j^r \prod_{i=1}^{j-1} \mathbf{g}_i^{(-1)^{x_i} r} \prod_{i=j+1}^n \mathbf{g}_i^r, \langle \mathbf{g}_i^r \rangle_{i=1}^{j-1}, \mathbf{g}_j^{-r}, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=j+1}^n\right) \tag{1}$$
$$\approx_{DDH}$$
$$\left(\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^n, \mathbf{g}^r, \mathbf{g}_j^{-r} \prod_{i=1}^{j-1} \mathbf{g}_i^{(-1)^{x_i} r} \prod_{i=j+1}^n \mathbf{g}_i^r, \langle \mathbf{g}_i^r \rangle_{i=1}^{j-1}, \mathbf{g}_j^{r}, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=j+1}^n\right). \tag{2}$$

Suppose there exists an adversary $A^*$, which can distinguish between the above two distributions within non-negligible advantage. We will show using the adversary $A^*$, we can break a DDH challenge $(\mathbf{g}, \mathbf{h}, \mathbf{g}^r, \chi)$

(where $\chi$ is either $\mathbf{h}^r$ or $\mathbf{h}^{-r}$ with probability 1/2 each). Given a DDH challenge $(\mathbf{g}, \mathbf{h}, \mathbf{g}^r, \chi)$, we take random $\langle u_i \rangle_{i=1}^{j-1} \leftarrow \mathbb{Z}_q^*$ and random $\langle u_i \rangle_{i=j+1}^{n} \leftarrow \mathbb{Z}_q^*$ and invoke adversary $A^*$ with input

$$\left(\mathbf{g}, \langle \mathbf{g}^{u_i} \rangle_{i=1}^{j-1}, \mathbf{h}, \langle \mathbf{g}^{u_i} \rangle_{i=j+1}^{n}, \mathbf{g}^r, \chi \prod_{i=1}^{j-1} \mathbf{g}^{(-1)^{x_i} u_i r} \prod_{i=j+1}^{n} \mathbf{g}^{u_i r}, \langle \mathbf{g}^{u_i r} \rangle_{i=1}^{j-1}, \chi^{-1}, \left\langle \mathbf{g}^{(-1)^{x_i} u_i r} \right\rangle_{i=j+1}^{n} \right).$$

Depending on whether $\chi$ takes the value $\mathbf{h}^r$ or $\mathbf{h}^{-r}$, the above expression is identically distributed as expression (1) or expression (2). So, using the output of $A^*$, we can break the DDH challenge. In other words, there is no such adversary $A^*$, which breaks $\mathbf{Game}_j$ with non-negligible probability, for $x_j = 1$. Now, transitioning through the sequence of games $\langle \mathbf{Game}_j \rangle_{j=1}^{n}$ we can argue the validity of this lemma. □

**Lemma 4.** *Under the Binary Mix DLP(Assumption 1) and DDH Assumptions, given random elements $\mathbf{g}, \langle \hat{\mathbf{g}}_i \rangle_{i=1}^{n}$ from a group $\mathbb{G}$ and $\mathbf{g}^r, \hat{\mathbf{v}} = \prod_{i=1}^{n} \hat{\mathbf{g}}_i^r, \langle \hat{\mathbf{z}}_i \rangle_{i=1}^{n} = \left\langle \hat{\mathbf{g}}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^{n}$ for a random $r \leftarrow \mathbb{Z}_q^*$, and random $r \leftarrow \mathbb{F}_2^n$ it is hard to output $y \in \mathbb{F}_2^n$, such that*

$$\hat{\mathbf{v}} = \prod_{i=1}^{n} \hat{\mathbf{z}}_i^{(-1)^{y_i}}.$$

*Formally, for all PPT adversaries $A$ there exists a negligible function* $\mathtt{negl}()$ *such that*

$$\Pr\left[\begin{array}{c} \mathbb{G} \leftarrow \mathcal{G}(1^\lambda); \mathbf{g}, \langle \hat{\mathbf{g}}_i \rangle_{i=1}^{n} \leftarrow \mathbb{G}; r \leftarrow \mathbb{Z}_q^*; r \leftarrow \mathbb{F}_2^n; \\ y \leftarrow A\left(\mathbf{g}, \langle \hat{\mathbf{g}}_i \rangle_{i=1}^{n}, \mathbf{g}^r, \hat{\mathbf{v}} = \prod_{i=1}^{n} \hat{\mathbf{g}}_i^r, \langle \hat{\mathbf{z}}_i \rangle_{i=1}^{n} = \left\langle \hat{\mathbf{g}}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^{n}\right) \end{array} : \hat{\mathbf{v}} = \prod_{i=1}^{n} \hat{\mathbf{z}}_i^{(-1)^{y_i}} \right] < \mathtt{negl}(\lambda)$$

*Proof.* Suppose we are given a Lemma 2 challenge

$$\left(\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^{n}, \mathbf{g}^r, \mathbf{v} = \prod_{i=1}^{n} \mathbf{g}_i^{(-1)^{x_i} r}, \langle \mathbf{z}_i \rangle_{i=1}^{n} = \langle \mathbf{g}_i^r \rangle_{i=1}^{n} \right),$$

for some random $\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^{n} \leftarrow \mathbb{G}$, random $r \leftarrow \mathbb{Z}_q^*$ and random $x \leftarrow \mathbb{F}_2^n$. Lemma 2 says, it is hard to output $y \in \mathbb{F}_2^n$ such that

$$\mathbf{v} = \prod_{i=1}^{n} \mathbf{z}_i^{(-1)^{y_i}}.$$

We will show if there exists an adversary $A^*$ which breaks Lemma 4 with non-negligible probability; we can use the adversary $A^*$ to break Lemma 2 challenge with non-negligible probability. $A^*$ takes input

$$\left(\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^{n}, \mathbf{g}^r, \hat{\mathbf{v}} = \prod_{i=1}^{n} \mathbf{g}_i^r, \langle \hat{\mathbf{z}}_i \rangle_{i=1}^{n} = \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^{n} \right)$$

and outputs $y$. Whenever, $A^*$ succeeds we have

$$\hat{\mathbf{v}} = \prod_{i=1}^{n} \hat{\mathbf{z}}_i^{(-1)^{y_i}}.$$

Lemma 3 says, Lemma 4 and Lemma 2 challenges are indistinguishable for all $x \in \mathbb{F}_2^n$ (hence, for random $x \leftarrow \mathbb{F}_2^n$ as well) under DDH assumption. Hence we can give the Lemma 2 challenge

$$\left(\mathbf{g}, \langle \mathbf{g}_i \rangle_{i=1}^{n}, \mathbf{g}^r, \mathbf{v} = \prod_{i=1}^{n} \mathbf{g}_i^{(-1)^{x_i} r}, \langle \mathbf{z}_i \rangle_{i=1}^{n} = \langle \mathbf{g}_i^r \rangle_{i=1}^{n} \right)$$

to $A^*$ and with non-negligible probability $A^*$'s output $y$ will satisfy the relation

$$\mathbf{v} = \prod_{i=1}^{n} \mathbf{z}_i^{(-1)^{y_i}},$$

which is a contradiction to Lemma 2. □

**Proof of Theorem 1** Now we show that if the relational hash from Section 3 is not one-way secure (and we have a one-wayness adversary $B$), then we can construct an adversary $A$ breaking Lemma 4. To achieve that, consider that the adversary $A$ is given a Lemma 4 challenge $\left(\mathbf{g}, \langle \hat{\mathbf{g}}_i \rangle_{i=1}^n, \mathbf{g}^r, \prod_{i=1}^n \hat{\mathbf{g}}_i^r, \left\langle \hat{\mathbf{g}}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^n \right)$. We now construct the one-wayness challenge as follows: We sample $u, s$ and $\langle u_i \rangle_{i=1}^n$, all randomly from $\mathbb{Z}_q^*$. Sample $\mathbf{h}_0$ randomly from $\mathbb{G}_2$. Now we define the output of KeyGen to be $pk := (pk_1, pk_2, pk_R)$ defined as follows:

$$pk_1 := \left(\mathbf{g}, \left\langle \hat{\mathbf{g}}_i^{u_i^{-1}} \right\rangle_{i=1}^n, \mathbf{g}^u \prod_{i=1}^n \hat{\mathbf{g}}_i^{-1} \right), \qquad pk_2 := \left(\mathbf{h}_0, \langle \mathbf{h}_0^{u_i s} \rangle_{i=1}^n, \mathbf{h}_0^s \right), \qquad pk_R := us$$

Observe that $\mathbf{g}, \left\langle \hat{\mathbf{g}}_i^{u_i^{-1}} \right\rangle_{i=1}^n, \mathbf{h}_0, \langle \mathbf{h}_0^{u_i s} \rangle_{i=1}^n, \mathbf{h}_0^s$ and $us$ are all uniformly random and independent elements of their respective domains [6]. The group element $\mathbf{g}^u \prod_{i=1}^n \hat{\mathbf{g}}_i^{-1}$ is determined given the other elements. Hence $(pk_1, pk_2, pk_R)$ has identical distribution as the output of the original KeyGen.

$A$ publishes $(pk_1, pk_2, pk_R)$ to the adversary $B$ and then also gives a challenge hash:

$$hx := \left(\mathbf{g}^r, \left\langle \hat{\mathbf{g}}_i^{(-1)^{x_i} r \cdot u_i^{-1}} \right\rangle_{i=1}^n, \mathbf{g}^{r \cdot u} \left( \prod_{i=1}^n \hat{\mathbf{g}}_i^r \right)^{-1} \right).$$

Once $B$ outputs an element $y \in \mathbb{F}_2^n$, $A$ just relays that to the Lemma 4 challenger. Now, observe that $hx$ is identically distributed as $\text{Hash}_1(x)$ for a random $x \leftarrow \mathbb{F}_2^n$. Therefore, the probability that $y = x$ is same as the advantage of $B$ against the security of the relational hash scheme. Therefore the scheme is secure given Lemma 4. □

## D Relational Hash for Linearity in $\mathbb{F}_p^n$

For any prime $p$, we now construct a Relational Hash scheme for the domains $X, Y, Z = \mathbb{F}_p^n$ and the relation $R = \{(x, y, z) \mid x + y = z \wedge x, y, z \in \mathbb{F}_p^n\}$.

KeyGen*:* Given the security parameter, bilinear groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are generated of prime order $q$, exponential in the security parameter and equal to 1 (mod $p$). This means the group $\mathbb{Z}_q^*$ has a subgroup $\mathbb{J}_p$ of prime order $p$. Let $\omega$ be an arbitrary generator of $\mathbb{J}_p$. Now we sample generators $\mathbf{g}_0 \leftarrow \mathbb{G}_1$ and $\mathbf{h}_0 \leftarrow \mathbb{G}_2$. Next we sample $\langle a_i \rangle_{i=1}^{n+1}$ and $\langle b_i \rangle_{i=1}^{n+1}$, all randomly from $\mathbb{Z}_q$. Define $\mathbf{g}_i = \mathbf{g}_0^{a_i}$ and $\mathbf{h}_i = \mathbf{h}_0^{b_i}$. Now we define the output of KeyGen as $pk := (\omega, pk_1, pk_2, pk_R)$ defined as follows:

$$pk_1 := \langle \mathbf{g}_i \rangle_{i=0}^{n+1}, \qquad pk_2 := \langle \mathbf{h}_i \rangle_{i=0}^{n+1}, \qquad pk_R := \sum_{i=1}^{n+1} a_i b_i$$

Hash$_1$*:* Given plaintext $x = \langle x_i \rangle_{i=1}^n \in \mathbb{F}_p^n, \omega$ and $pk_1 = \mathbf{g}_0, \langle \mathbf{g}_i \rangle_{i=1}^n$, the hash is constructed as follows: Sample a random $r \in \mathbb{Z}_q^*$ and then compute the following:

$$hx := \left(\mathbf{g}_0^r, \left\langle \mathbf{g}_i^{\omega^{x_i} r} \right\rangle_{i=1}^n, \mathbf{g}_{n+1}^r \right)$$

Hash$_2$ is analogously defined in the group $\mathbb{G}_2$.

Verify*:* Given hashes $hx = \langle hx_i \rangle_{i=0}^{n+1}$ and $hy = \langle hy_i \rangle_{i=0}^{n+1}$, the parameter $z = \langle z_i \rangle_{i=1}^n \in \mathbb{F}_p^n$ and $pk_R$ and $\omega$, the algorithm Verify checks the following equality:

$$e(hx_0, hy_0)^{pk_R} \stackrel{?}{=} e(hx_{n+1}, hy_{n+1}) \prod_{i=1}^n e(hx_i, hy_i)^{\omega^{-z_i}}$$

[6] Roughly, $\hat{\mathbf{g}}_i^{u_i^{-1}}$'s are randomized by the $\hat{\mathbf{g}}_i$'s; $\mathbf{h}_0^{u_i s}$'s are randomized by the $u_i$'s; $\mathbf{h}_0^s$ is randomized by $s$ and $us$ is randomized by $u$.

*Correctness.* The correctness property of the scheme can be proven similar to the correctness of the $\mathbb{F}_2^n$ linear relational hash from Section 3. For any, $x, y \in \mathbb{F}_p^n$ we have

$$\textsc{Hash}_1(x) = (hx_0, \langle hx_i\rangle_{i=1}^n, hx_{n+1}) = \left(\mathbf{g}_0^r, \left\langle \mathbf{g}_i^{\omega^{x_i}r}\right\rangle_{i=1}^n, \mathbf{g}_{n+1}^r\right)$$
$$\textsc{Hash}_1(y) = (hy_0, \langle hy_i\rangle_{i=1}^n, hy_{n+1}) = \left(\mathbf{h}_0^s, \left\langle \mathbf{h}_i^{\omega^{y_i}s}\right\rangle_{i=1}^n, \mathbf{h}_{n+1}^s\right)$$

Hence,

$$e(hx_{n+1}, hy_{n+1})\prod_{i=1}^n e(hx_i, hy_i)^{\omega^{-(x_i+y_i)}} = e(\mathbf{g}_{n+1}^r, \mathbf{h}_{n+1}^s)\prod_{i=1}^n e\left(\mathbf{g}_i^{\omega^{x_i}r}, \mathbf{h}_i^{\omega^{y_i}s}\right)^{\omega^{-(x_i+y_i)}}$$
$$= \prod_{i=1}^{n+1} e\left(\mathbf{g}_i^r, \mathbf{h}_i^s\right) = \prod_{i=1}^{n+1} e\left(\mathbf{g}_0^{a_i r}, \mathbf{h}_0^{b_i s}\right) = e\left(\mathbf{g}_0^r, \mathbf{h}_0^s\right)^{\sum_{i=1}^{n+1} a_i b_i} = e(hx_0, hy_0)^{pk_R}$$

This shows that our relational hash scheme correctly verifies tuples of the form $(x, y, x+y)$ for any $x, y \in \mathbb{F}_p^n$.

On the other hand, if the verification equation gets satisfied for some $z \in \mathbb{F}_p^n$, we must have

$$e\left(\mathbf{g}_0^r, \mathbf{h}_0^s\right)^{\sum_{i=1}^{n+1} a_i b_i} = e(\mathbf{g}_{n+1}^r, \mathbf{h}_{n+1}^s)\prod_{i=1}^n e\left(\mathbf{g}_i^{\omega^{x_i}r}, \mathbf{h}_i^{\omega^{y_i}s}\right)^{\omega^{-z_i}}$$
$$\implies \sum_{i=1}^n a_i b_i = \sum_{i=1}^n \omega^{x_i+y_i-z_i} a_i b_i$$

Let $U \subseteq \{1, \cdots, n\}$ be the set of indices, such that $j \in U$ if and only if $x_i + y_i \neq z_i$. Now, the above equation reduces to

$$\sum_{i\in Q}(1-\omega^{x_i+y_i-z_i})a_i b_i = 0.$$

If $x + y \neq z$, then $Q$ is non empty and we can consider a fixed $i^* \in Q$ and we have

$$a_{i^*} b_{i^*} = -(1-\omega^{x_{i^*}+y_{i^*}-z_{i^*}})^{-1}\sum_{i\in Q\setminus\{i^*\}}(1-\omega^{x_i+y_i-z_i})a_i b_i.$$

Now, if we fix $a_i$'s and $b_i$'s for $i \in Q\setminus\{i^*\}$ and consider only the randomness of $a_{i^*}$ and $b_{i^*}$, the above equation holds with probability at most $1/q$ when $\sum_{i\in Q\setminus\{i^*\}}(1-\omega^{x_i+y_i-z_i})a_i b_i \neq 0$ and with probability at most $2/q$ when $\sum_{i\in Q\setminus\{i^*\}}(1-\omega^{x_i+y_i-z_i})a_i b_i = 0$. This implies for any tuple $(x, y, z)$ with $z \neq x + y$, the verification equation gets satisfied with probability at most $2/q$. Hence the above algorithms in fact constitute a correct relational hash for linearity over $\mathbb{F}_p^n$. □

*Security.* We assume $n \log p$ is at least in the order of security parameter and the p-ary Mix DLP assumption (stated below) holds. This is a generalized version of Binary Mix DLP(Assumption 1). For $p = 2$ the assumptions are equivalent. This assumption also incorporates the case when $n$ is small (constant), but $p$ is large (exponential in security parameter $\lambda$).

**Assumption 7 (p-ary Mix DLP).** *Assuming a generation algorithm $\mathcal{G}$ that outputs a tuple $(n, q, \omega, \mathbb{G})$ such that $\mathbb{G}$ is a group of prime order $q = 1 \pmod p$, for some prime $p$ and $\omega$ is an arbitrary element from $\mathbb{Z}_q^*$ of order $p$, the p-ary Mix DLP assumption asserts that given random elements $\langle \mathbf{g}_i\rangle_{i=1}^n$ from the group $\mathbb{G}$ and $\prod_{i=1}^n \mathbf{g}_i^{\omega^{x_i}}$, for a random $x \leftarrow \mathbb{F}_p^n$, it is computationally infeasible to output $y \in \mathbb{F}_p^n$ such that,*

$$\prod_{i=1}^n \mathbf{g}_i^{\omega^{x_i}} = \prod_{i=1}^n \mathbf{g}_i^{\omega^{y_i}}.$$

*Remark 8.* Similar to proof of Binary Mix DLP(Assumption 1) in the Generic Group Model in Section E.2, we can also prove the above assumption is secure in the Generic Group Model. We skip the proof for brevity.

**Theorem 10.** *The above algorithms* $(\text{KeyGen}, \text{Hash}_1, \text{Hash}_2, \text{Verify})$ *constitute a relational hash scheme for the relation* $R = \{(x, y, z) \mid x + y = z \wedge x, y, z \in \mathbb{F}_p^n\}$. *The scheme is one-way secure under the SXDH assumption and p-ary Mix DLP assumption (Assumption 7), when $x$ and $y$'s are sampled uniformly from* $\mathbb{F}_p^n$.

*Proof.* We prove the theorem starting with a similar lemma as the $\mathbb{F}_2^n$ case. We skip the proof of this lemma as it is almost identical to the $\mathbb{F}_2^n$ lemma.

**Lemma 5.** *Under the p-ary Mix DLP(Assumption 7) and DDH Assumptions, given random elements* $\mathbf{g}, \langle \hat{\mathbf{g}}_i \rangle_{i=1}^n$ *from a group* $\mathbb{G}$ *and* $\mathbf{g}^r, \hat{\mathbf{v}} = \prod_{i=1}^n \hat{\mathbf{g}}_i^r, \langle \hat{\mathbf{z}}_i \rangle_{i=1}^n = \left\langle \hat{\mathbf{g}}_i^{\omega^{x_i} r} \right\rangle_{i=1}^n$ *for a random* $r \leftarrow \mathbb{Z}_q^*$, *and random* $x \leftarrow \mathbb{F}_p^n$ *it is hard to output* $y \in \mathbb{F}_p^n$, *such that*

$$\hat{\mathbf{v}} = \prod_{i=1}^n \hat{\mathbf{z}}_i^{\omega^{-y_i}}.$$

*Formally, for all PPT adversaries $A$ there exists a negligible function* $\mathtt{negl}()$ *such that*

$$\Pr\left[\begin{array}{c} \mathbb{G} \leftarrow \mathcal{G}(1^\lambda); \mathbf{g}, \langle \hat{\mathbf{g}}_i \rangle_{i=1}^n \leftarrow \mathbb{G}; r \leftarrow \mathbb{Z}_q^*; x \leftarrow \mathbb{F}_p^n; \\ y \leftarrow A\left(\mathbf{g}, \langle \hat{\mathbf{g}}_i \rangle_{i=1}^n, \mathbf{g}^r, \hat{\mathbf{v}} = \prod_{i=1}^n \hat{\mathbf{g}}_i^r, \langle \hat{\mathbf{z}}_i \rangle_{i=1}^n = \left\langle \hat{\mathbf{g}}_i^{\omega^{x_i} r} \right\rangle_{i=1}^n\right) \end{array} : \hat{\mathbf{v}} = \prod_{i=1}^n \hat{\mathbf{z}}_i^{\omega^{-y_i}}\right] < \mathtt{negl}(\eta)$$

Now we show that if the relational hash is not one-way secure (and we have a one-wayness adversary $B$), then we can construct an adversary $A$ breaking Lemma 5. To achieve that, consider that the adversary $A$ is given a Lemma 5 challenge $\left(\mathbf{g}, \langle \hat{\mathbf{g}}_i \rangle_{i=1}^n, \mathbf{g}^r, \prod_{i=1}^n \hat{\mathbf{g}}_i^r, \left\langle \hat{\mathbf{g}}_i^{\omega^{x_i} r} \right\rangle_{i=1}^n\right)$. We now construct the one-wayness challenge as follows: We sample $u, s$ and $\langle u_i \rangle_{i=1}^n$, all randomly from $\mathbb{Z}_q^*$. Sample $\mathbf{h}_0$ randomly from $\mathbb{G}_2$. Now we define the output of KeyGen to be $pk := (pk_1, pk_2, pk_R)$ defined as follows:

$$pk_1 := \left(\mathbf{g}, \left\langle \hat{\mathbf{g}}_i^{u_i^{-1}} \right\rangle_{i=1}^n, \mathbf{g}^u \prod_{i=1}^n \hat{\mathbf{g}}_i^{-1}\right), \qquad pk_2 := \left(\mathbf{h}_0, \langle \mathbf{h}_0^{u_i s} \rangle_{i=1}^n, \mathbf{h}_0^s\right), \qquad pk_R := us$$

Observe that $\mathbf{g}, \left\langle \hat{\mathbf{g}}_i^{u_i^{-1}} \right\rangle_{i=1}^n, \mathbf{h}_0, \langle \mathbf{h}_0^{u_i s} \rangle_{i=1}^n, \mathbf{h}_0^s$ and $us$ are all uniformly random and independent elements of their respective domains. The group element $\mathbf{g}^u \prod_{i=1}^n \hat{\mathbf{g}}_i^{-1}$ is fixed given the other elements. Hence $(pk_1, pk_2, pk_R)$ has identical distribution as the output of the original KeyGen.

$A$ publishes $(pk_1, pk_2, pk_R)$ to the adversary $B$ and then also gives a challenge hash:

$$hx := \left(\mathbf{g}^r, \left\langle \hat{\mathbf{g}}_i^{\omega^{x_i} r \cdot u_i^{-1}} \right\rangle_{i=1}^n, \mathbf{g}^{r \cdot u} \left(\prod_{i=1}^n \hat{\mathbf{g}}_i^r\right)^{-1}\right).$$

Once $B$ outputs an element $y \in \mathbb{F}_p^n$, $A$ just relays that to the Lemma 4 challenger. Now, observe that $hx$ is identically distributed as $\text{Hash}_1(x)$ for a random $x \leftarrow \mathbb{F}_p^n$. Therefore, the probability that $y = x$ is same as the advantage of $B$ against the security of the relational hash scheme. Therefore the scheme is secure given Lemma 5. ☐

*Unforgeability and Oracle Simulation Security.* This relational hash is also in fact a 2-value perfectly one-way function, albeit under a stronger hardness assumption. The hardness assumption and proof of 2-value perfectly one-wayness is analogous to the $\mathbb{F}_2^n$ case. By Theorem 8 from Section 5, that will imply this relational hash construction is also unforgeable and oracle simulation secure.

**Relational Hash for Hamming Proximity in $\mathbb{F}_p^n$.** As in the case for $\mathbb{F}_2^n$, we can also construct a relational hash for the domains $X, Y = \mathbb{F}_p^n$ and the relation $R_\delta = \{(x,y) \mid \texttt{dist}(x,y) \leq \delta \ \wedge \ x,y \in \mathbb{F}_p^n\}$, where dist is the p-ary hamming distance and $\delta$ is a positive integer less than $n$. We use a family of $(n,k,d)$ linear error correcting codes (ECC) $\mathcal{C}$ of alphabet size $p$ and a relational hash for linearity in $\mathbb{F}_p^k$: $(\textsc{KeyGenLinear}, \textsc{HashLinear}_1, \textsc{HashLinear}_2, \textsc{VerifyLinear})$. The construction, correctness and security are analogous to the binary case.

## E Justification of New Hardness Assumptions

In this section we justify the new hardness assumptions proposed in this paper by either showing each as an implication of a more standard assumption or by proving it in the Generic Group Model [Sho97].

### E.1 Binary Mix DLP is as Hard as Random Modular Subset Sum

We recall the Binary Mix DLP assumption (Assumption 1) from Section 3.

**Assumption 1. (Binary Mix DLP) :** *Assuming a generation algorithm $\mathcal{G}$ that outputs a tuple $(n, q, \mathbb{G})$ such that $\mathbb{G}$ is a group of prime order $q$, the Binary Mix DLP assumption asserts that given random elements $\langle \mathbf{g}_i \rangle_{i=1}^n$ from the group $\mathbb{G}$ and $\prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}}$, for a random $x \leftarrow \mathbb{F}_2^n$, it is computationally infeasible to output $y \in \mathbb{F}_2^n$ such that*

$$\prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}} = \prod_{i=1}^n \mathbf{g}_i^{(-1)^{y_i}}.$$

**Theorem 11.** *The Binary-Mix-DLP assumption (Assumption 1) is implied by the Random-Modular-Subset-Sum assumption (Assumption 6).*

*Proof.* We show, given a Binary-Mix-DLP attacker $\mathcal{A}$, we can solve any Random-Modular-Subset-Sum challenge $(\langle a_i \rangle_{i=1}^n, c)$. Suppose, the Binary-Mix-DLP attacker $\mathcal{A}$ works in a group $\mathbb{G}$ of order $q$, which has a generator $\mathbf{g}$. We invoke $\mathcal{A}$ on input

$$\left( \langle \mathbf{g}^{a_i} \rangle_{i=1}^n, \mathbf{g}^{-2c + \sum_{i=1}^n a_i} \right).$$

If $\mathcal{A}$ successfully outputs $\tau \in \mathbb{F}_2^n$ as a solution to the above Binary-Mix-DLP problem, $\tau$ is also a solution to the Random-Modular-Subset-Sum challenge. □

### E.2 Hardness of Binary Mix DLP and Decisional Binary Mix: Proof in the Generic Group Model

In this section we show Binary Mix DLP(Assumption 1) and Decisional Binary Mix(Assumption 2) hold in the Generic Group Model. Let $\mathcal{A}$ be a probabilistic polynomial time (PPT) generic group adversary. Following [Sho97], the generic group model is implemented by choosing a random encoding $\sigma : \mathbb{G} \to \{0,1\}^m$ (where $m >> \log q$). Instead of working directly with group elements, $\mathcal{A}$ works with images of group elements under $\sigma$. This implies, all $\mathcal{A}$ can do, is test for elemental equality. $\mathcal{A}$ is also given access to following two oracles:

- **Group Action Oracle** : Given $\sigma(g_1)$ and $\sigma(g_2)$, it returns $\sigma(g_1 g_2)$.
- **Group Inversion Oracle** : Given $\sigma(g)$, it returns $\sigma(g^{-1})$.

We also assume, $\mathcal{A}$ queries the oracles with encoding of the elements it has previously seen. This assumption holds, because the probability of choosing a string which is also a image of $\sigma$ is negligible (as $m >> \log q$).

**Theorem 12.** *The Binary Mix DLP assumption (Assumption 1) holds in the Generic Group Model.*

*Proof.* We consider an algorithm $\mathcal{B}$ playing the following game with $\mathcal{A}$. Algorithm $\mathcal{B}$ chooses $n$ bit strings uniformly in $\{0,1\}^m$:

$$\left\langle \sigma_g^i \right\rangle_{i=1}^n, \sigma_g$$

and gives them to $\mathcal{A}$. Internally, $\mathcal{B}$ keeps track of the encoded elements using polynomials in the ring

$$\mathbb{F}_q[R_1, \cdots, R_n, T_g].$$

To maintain consistency with the bit strings given to $\mathcal{A}$, $\mathcal{B}$ creates a list $L$ of pairs $(F, \sigma)$ where $F$ is a polynomial in the ring specified above and $\sigma$ is the encoding of a group element. The polynomial $F$ represents the exponent of the encoded element. Initially, $L$ is set to,

$$L_0 = \{\left\langle (R_i, \sigma_g^i) \right\rangle_{i=1}^n, (T_g, \sigma_g)\}.$$

Algorithm $\mathcal{B}$ simulates the oracles as follows.

- **Group Action** : Given two strings $\sigma_i, \sigma_j$, $\mathcal{B}$ recovers the corresponding polynomials $F_i, F_j$ and computes $F_i + F_j$. If $F_i + F_j$ is already in $L$, $\mathcal{B}$ returns the corresponding bit string; otherwise it returns a uniform bit string $\sigma \in \{0,1\}^m$ and stores $(F_i + F_j, \sigma)$ in $L$.
- **Group Inversion** : Given an element $\sigma$, $\mathcal{B}$ recovers the corresponding polynomial representation $F$ and computes $-F$. If the polynomial $-F$ is already in $L$, $\mathcal{B}$ returns the corresponding bit string; otherwise it returns a uniform bit string $\sigma \in \{0,1\}^m$ and stores $(-F, \sigma)$ in $L$.

After $\mathcal{A}$ queried the oracles, it outputs a $y \in \mathbb{F}_2^n$. At this point, $\mathcal{B}$ chooses random $x \leftarrow \mathbb{F}_2^n$ and $\langle r_i \rangle_{i=1}^n$ from $\mathbb{Z}_q$ at random. $\mathcal{B}$ sets:

$$\langle R_i \rangle_{i=1}^n = \langle r_i \rangle_{i=1}^n$$
$$T_g = \sum_{i=1}^n (-1)^{x_i} r_i$$

$\mathcal{A}$ wins the game if one of the following is true.

- **Case - I :** $T_g = \sum_{i=1}^n (-1)^{y_i} R_i$
- **Case - II :** Simulation of $\mathcal{B}$ is inconsistent.

Suppose $\boldsymbol{x}$ is the random variable corresponding to random choice of $x$. Now, we find an upper bound for the Case - I probability.

$$\begin{aligned}
\Pr\left[T_g = \sum_{i=1}^{n}(-1)^{y_i}R_i\right] &= \Pr\left[\sum_{i=1}^{n}(-1)^{\mathbf{x_i}}R_i = \sum_{i=1}^{n}(-1)^{y_i}R_i\right] \\
&= \Pr\left[\sum_{i=1}^{n}(-1)^{\mathbf{x_i}}R_i = \sum_{i=1}^{n}(-1)^{y_i}R_i \middle| \boldsymbol{x} = y\right]\Pr[\boldsymbol{x} = y] \\
&\quad + \sum_{\substack{\eta\in\mathbb{F}_2^n \\ \eta\neq 0^n}} \Pr\left[\sum_{i=1}^{n}(-1)^{\mathbf{x_i}}R_i = \sum_{i=1}^{n}(-1)^{y_i}R_i \middle| \boldsymbol{x} = y+\eta\right]\Pr[\boldsymbol{x} = y+\eta] \\
&= \frac{1}{2^n} + \frac{1}{2^n}\sum_{\substack{\eta\in\mathbb{F}_2^n \\ \eta\neq 0^n}} \Pr\left[\sum_{i=1}^{n}(-1)^{y_i}((-1)^{\eta_i}-1)R_i = 0 \middle| \boldsymbol{x} = y+\eta\right] \\
&= \frac{1}{2^n} + \frac{1}{2^n}\sum_{\substack{\eta\in\mathbb{F}_2^n \\ \eta\neq 0^n}} \Pr\left[-2\sum_{i:\eta_i=1}(-1)^{y_i}R_i = 0\right] \\
&= \frac{1}{2^n} + \frac{1}{2^n}\sum_{\substack{\eta\in\mathbb{F}_2^n \\ \eta\neq 0^n}} \frac{1}{q} \\
&\leq \frac{1}{2^n} + \frac{1}{q}
\end{aligned}$$

The list $L$, is initially set to $L_0$. New polynomials get added to the list, because of invocation of Group Action and Group Inversion oracles by $\mathcal{A}$. However, the operations of these two oracles never increase the degree of the polynomials present in the list $L$. In other words, any polynomial $F_i \in L$ is of the form:

$$F_i = \sum_{k=1}^{n} a_k^i R_k + +c^i T_g,$$

where $\langle a_k^i \rangle_{k=1}^n$, $c^i$ are some constants from $\mathbb{Z}_q$.

We need to show, two distinct polynomials $F_i$ and $F_j$ can collide after substituting random values of $\langle r_i \rangle_{i=1}^n$, $x$ only with negligible probability. In other words, we need to find an upper bound of the probability that the polynomial

$$F_i - F_j = \sum_{k=1}^{n}(a_k^i - a_k^j)R_k + (c^i - c^j)T_g$$

hits a zero.

Lemma 6 shows, this upper bound is actually $\frac{1}{q} + \frac{1}{2^n}$. Now, if an adversary makes $t$ queries, the size of the list $L$ can be upper bounded by $|L_0| + t = n + t + 1$. Hence, the probability that the $\mathcal{B}$ 's simulation is inconsistent is at most

$$(n+t+1)^2\left(\frac{1}{q} + \frac{1}{2^n}\right).$$

After adding the upper bound of the probability of Case-I along with it, we get an upper bound of $\mathcal{A}$'s advantage as,

$$\left((n+t+1)^2 + 1\right)\left(\frac{1}{q} + \frac{1}{2^n}\right).$$

□

**Lemma 6.** *For any nonzero polynomial*

$$F = \sum_{i=1}^{n} a_i R_i + cT_g,$$

*in* $\mathbb{F}_q[R_1, \cdots, R_n, T_g]$ *the probability that the polynomial hits a zero is at most*

$$\frac{1}{q} + \frac{1}{2^n},$$

*where* $\langle r_i \rangle_{i=1}^n$ *are chosen from* $\mathbb{Z}_q$ *at random,* $x$ *is chosen from* $\mathbb{F}_2^n$ *at random and* $\langle R_i \rangle_{i=1}^n$*,* $T_g$ *are substituted as follows:*

$$\langle R_i \rangle_{i=1}^n = \langle r_i \rangle_{i=1}^n$$
$$T_g = \sum_{i=1}^{n} (-1)^{x_i} r_i.$$

*Proof.* $\boldsymbol{x}$ be the random variables corresponding to the random choices of $x$ from $\mathbb{F}_2^n$.

$$\Pr[F = 0] = \Pr\left[\sum_{i=1}^{n} (a_i + (-1)^{\boldsymbol{x_i}} c) R_i = 0\right]$$

Now we evaluate an upper bound for the right hand expression in various cases, depending on the values of the constants $\langle a_i \rangle_{i=1}^n, c$.

**Case I -** ($c \neq 0$): For any $u \in \mathbb{F}_2^n$,

$$\Pr\left[\sum_{i=1}^{n} (a_i + (-1)^{\boldsymbol{x_i}} c) R_i = 0\right] \leq (1 - \frac{1}{2^n})\frac{1}{q} + \frac{1}{2^n}.$$

Note, the $(1 - \frac{1}{2^n})$ factor comes from the fact that, all $a_i$'s might take the values $\pm c$, and in that case all coefficients of $R_i$'s becomes zero with probability $\frac{1}{2^n}$.

**Case II -** ($c = 0$) : There erists $i^*$, s.t. $a_{i^*} \neq 0$ (otherwise, $F$ becomes a zero polynomial).

$$\Pr\left[\sum_{i=1}^{n} (a_i + (-1)^{\boldsymbol{x_i}} c) R_i = 0\right] = \frac{1}{q}.$$

Combining both the cases, we get

$$\Pr\left[\sum_{i=1}^{n} (a_i + (-1)^{\boldsymbol{x_i}} c) R_i = 0\right] \leq \frac{1}{q} + \frac{1}{2^n}.$$

□

**Theorem 13.** *The Decisional Binary Mix assumption (Assumption 2) holds in the Generic Group Model.*

*Proof.* We consider an algorithm $\mathcal{B}$ playing the following game with $\mathcal{A}$. Algorithm $\mathcal{B}$ chooses $2n + 1$ bit strings uniformly in $\{0, 1\}^m$:

$$\langle \sigma_g^i \rangle_{i=1}^n, \langle \sigma_f^i \rangle_{i=1}^n, \sigma_g, \sigma_f^0, \sigma_f^1,$$

and gives them to $\mathcal{A}$. Internally, $\mathcal{B}$ keeps track of the encoded elements using polynomials in the ring

$$\mathbb{F}_q[R_1, \cdots, R_n, S_1, \cdots, S_n, T_g, T_{f,0}, T_{f,1}].$$

To maintain consistency with the bit strings given to $\mathcal{A}$, $\mathcal{B}$ creates a list $L$ of pairs $(F, \sigma)$ where $F$ is a polynomial in the ring specified above and $\sigma$ is the encoding of a group element. The polynomial $F$ represents the exponent of the encoded element. Initially, $L$ is set to,

$$L_0 = \{\left\langle (R_i, \sigma_g^i) \right\rangle_{i=1}^n, \left\langle (S_i, \sigma_f^i) \right\rangle_{i=1}^n, (T_g, \sigma_g), (T_{f,0}, \sigma_f^0), (T_{f,1}, \sigma_f^1)\}.$$

Algorithm $\mathcal{B}$ simulates the oracles as follows.

- **Group Action** : Given two strings $\sigma_i, \sigma_j$, $\mathcal{B}$ recovers the corresponding polynomials $F_i, F_j$ and computes $F_i + F_j$. If $F_i + F_j$ is already in $L$, $\mathcal{B}$ returns the corresponding bit string; otherwise it returns a uniform bit string $\sigma \in \{0,1\}^m$ and stores $(F_i + F_j, \sigma)$ in $L$.
- **Group Inversion** : Given an element $\sigma$, $\mathcal{B}$ recovers the corresponding polynomial representation $F$ and computes $-F$. If the polynomial $-F$ is already in $L$, $\mathcal{B}$ returns the corresponding bit string; otherwise it returns a uniform bit string $\sigma \in \{0,1\}^m$ and stores $(-F, \sigma)$ in $L$.

After $\mathcal{A}$ queried the oracles, it outputs a bit $b'$. At this point, $\mathcal{B}$ chooses a bit $b$ at random and $\langle r_i \rangle_{i=1}^n, \langle s_i \rangle_{i=1}^n$ from $\mathbb{Z}_q$ at random. $\mathcal{B}$ also chooses $x, y$ from $\mathbb{F}_2^n$ at random. $\mathcal{B}$ sets:

$$\begin{aligned}
\langle R_i \rangle_{i=1}^n &= \langle r_i \rangle_{i=1}^n \\
\langle S_i \rangle_{i=1}^n &= \langle s_i \rangle_{i=1}^n \\
T_g &= \sum_{i=1}^n (-1)^{x_i} r_i \\
T_{f,b} &= \sum_{i=1}^n (-1)^{x_i} s_i \\
T_{f,1-b} &= \sum_{i=1}^n (-1)^{y_i} s_i
\end{aligned}$$

If the simulation provided by $\mathcal{B}$ is consistent, it reveals nothing about $b$. This means $\mathcal{A}$ can only guess the correct value of $b$ with probability 1/2. The simulation can be inconsistent, only if the random choices of $b, \langle r_i \rangle_{i=1}^n, \langle s_i \rangle_{i=1}^n, x, y$ by $\mathcal{B}$ produce a collision(i.e. two different polynomials taking the same value) in the list $L$.

The list $L$, is initially set to $L_0$. New polynomials get added to the list, because of invocation of Group Action and Group Inversion oracles by $\mathcal{A}$. However, the operations of these two oracles never increase the degree of the polynomials present in the list $L$. In other words, any polynomial $F_i \in L$ is of the form:

$$F_i = \sum_{k=1}^n a_k^i R_k + \sum_{k=1}^n b_k^i S_k + c^i T_g + d^i T_{f,0} + e^i T_{f,1},$$

where $\left\langle a_k^i \right\rangle_{k=1}^n, \left\langle b_k^i \right\rangle_{k=1}^n, c^i, d^i, e^i$ are some constants from $\mathbb{Z}_q$.

We need to show, two distinct polynomials $F_i$ and $F_j$ can collide after substituting random values of $b, \langle r_i \rangle_{i=1}^n, \langle s_i \rangle_{i=1}^n, x, y$ only with negligible probability. In other words, we need to find an upper bound of the probability that the polynomial

$$F_i - F_j = \sum_{k=1}^n (a_k^i - a_k^j) R_k + \sum_{k=1}^n (b_k^i - b_k^j) S_k + (c^i - c^j) T_g + (d^i - d^j) T_{f,0} + (e^i - e^j) T_{f,1}$$

hits a zero.

Lemma 7 shows, this upper bound is actually $\frac{1}{q} + \frac{1}{2^n}$. Now, if an adversary makes $t$ queries, the size of the list $L$ can be upper bounded by $|L_0| + t = 2n + t + 3$. Hence, the probability that the $\mathcal{B}$ 's simulation is inconsistent is at most

$$(2n + t + 3)^2 \left( \frac{1}{q} + \frac{1}{2^n} \right),$$

which is an upper bound of the advantage of any generic group adversary such as $\mathcal{A}$. □

**Lemma 7.** *For any nonzero polynomial*

$$F = \sum_{i=1}^{n} a_i R_i + \sum_{i=1}^{n} b_i S_i + cT_g + dT_{f,0} + eT_{f,1},$$

*in* $\mathbb{F}_q[R_1, \cdots, R_n, S_1, \cdots, S_n, T_g, T_{f,0}, T_{f,1}]$ *the probability that the polynomial hits a zero is at most*

$$\frac{1}{q} + \frac{1}{2^n},$$

*where bit* $b$ *is chosen at random,* $\langle r_i \rangle_{i=1}^n, \langle s_i \rangle_{i=1}^n$ *are chosen from* $\mathbb{Z}_q$ *at random,* $x, y$ *are chosen from* $\mathbb{F}_2^n$ *at random and* $\langle R_i \rangle_{i=1}^n$, $\langle S_i \rangle_{i=1}^n$, $T_g$, $T_{f,0}$, $T_{f,1}$ *are substituted as follows:*

$$\begin{aligned}
\langle R_i \rangle_{i=1}^n &= \langle r_i \rangle_{i=1}^n \\
\langle S_i \rangle_{i=1}^n &= \langle s_i \rangle_{i=1}^n \\
T_g &= \sum_{i=1}^{n} (-1)^{x_i} r_i \\
T_{f,b} &= \sum_{i=1}^{n} (-1)^{x_i} s_i \\
T_{f,1-b} &= \sum_{i=1}^{n} (-1)^{y_i} s_i.
\end{aligned}$$

*Proof.* At first we upper bound the probability $\Pr[F = 0 | b = 0]$. $\boldsymbol{x}$ and $\boldsymbol{y}$ be the random variables corresponding to the random choices of $x$ and $y$ from $\mathbb{F}_2^n$.

$$\Pr[F = 0 | b = 0] = \Pr\left[\sum_{i=1}^{n} (a_i + (-1)^{\boldsymbol{x_i}} c) R_i + \sum_{i=1}^{n} (b_i + (-1)^{\boldsymbol{x_i}} d + (-1)^{\boldsymbol{y_i}} e) S_i = 0\right]$$

Now we evaluate an upper bound for the right hand expression in various cases, depending on the values of the constants $\langle a_i \rangle_{i=1}^n, \langle b_i \rangle_{i=1}^n, c, d, e$.

**Case I -** $(c \neq 0)$: For any $u \in \mathbb{F}_2^n$,

$$\Pr\left[\sum_{i=1}^{n} (a_i + (-1)^{\boldsymbol{x_i}} c) R_i = u\right] \leq (1 - \frac{1}{2^n}) \frac{1}{q} + \frac{1}{2^n}.$$

Note, the $(1 - \frac{1}{2^n})$ factor comes from the fact that, all $a_i$'s might take the values $\pm c$, and in that case all coefficients of $R_i$'s becomes zero with probability $\frac{1}{2^n}$. The additive $\frac{1}{2^n}$ term comes for the special case $u = 0$.

**Case II -** ($c = 0$ and there exists $i^*$, s.t. $a_{i^*} \neq 0$): For any $u \in \mathbb{F}_2^n$,

$$\Pr\left[\sum_{i=1}^{n} (a_i + (-1)^{\boldsymbol{x_i}} c) R_i = u\right] = \frac{1}{q}.$$

**Case III -** ($c = 0$ , $a_i = 0$ for all $i$, $d \neq 0$, $e \neq 0$): For any $u \in \mathbb{F}_2^n$,

$$\Pr\left[\sum_{i=1}^{n} (b_i + (-1)^{\boldsymbol{x_i}} d + (-1)^{\boldsymbol{y_i}} e) S_i = u\right] \leq (1 - \frac{1}{4^n}) \frac{1}{q} + \frac{1}{4^n}.$$

Note, the $(1 - \frac{1}{4^n})$ factor comes from the fact that, all $b_i$'s might take the values $(\pm d \pm e)$, and in that case all coefficients of $S_i$'s becomes zero with probability $\frac{1}{4^n}$. The additive $\frac{1}{4^n}$ term comes for the special case $u = 0$.

**Case IV -** ($c = 0$ , $a_i = 0$ for all $i$, $d = 0$, $e \neq 0$): For any $u \in \mathbb{F}_2^n$,

$$\Pr\left[\sum_{i=1}^{n}(b_i + (-1)^{\boldsymbol{x_i}} d + (-1)^{\boldsymbol{y_i}} e)S_i = u\right] \leq (1 - \frac{1}{2^n})\frac{1}{q} + \frac{1}{2^n}.$$

Note, the $(1 - \frac{1}{2^n})$ factor comes from the fact that, all $b_i$'s might take the values $\pm e$, and in that case all coefficients of $S_i$'s becomes zero with probability $\frac{1}{2^n}$. The additive $\frac{1}{2^n}$ term comes for the special case $u = 0$.

**Case V -** ($c = 0$ , $a_i = 0$ for all $i$, $d \neq 0$, $e = 0$): For any $u \in \mathbb{F}_2^n$,

$$\Pr\left[\sum_{i=1}^{n}(b_i + (-1)^{\boldsymbol{x_i}} d + (-1)^{\boldsymbol{y_i}} e)S_i = u\right] \leq (1 - \frac{1}{2^n})\frac{1}{q} + \frac{1}{2^n}.$$

Note, the $(1 - \frac{1}{2^n})$ factor comes from the fact that, all $b_i$'s might take the values $\pm d$, and in that case all coefficients of $S_i$'s becomes zero with probability $\frac{1}{2^n}$. The additive $\frac{1}{2^n}$ term comes for the special case $u = 0$.

**Case VI -** ($c = 0$ , $a_i = 0$ for all $i$, $d = 0$, $e = 0$): There exists $j^*$, s.t. $b_{j^*} \neq 0$ (otherwise, $F$ becomes the zero polynomial). Hence, for any $u \in \mathbb{F}_2^n$,

$$\Pr\left[\sum_{i=1}^{n}(b_i + (-1)^{\boldsymbol{x_i}} d + (-1)^{\boldsymbol{y_i}} e)S_i = u\right] = \frac{1}{q}.$$

Hence, combining all cases together we have,

$$\begin{aligned}\Pr[F = 0|b = 0] &= \Pr\left[\sum_{i=1}^{n}(a_i + (-1)^{\boldsymbol{x_i}} c)R_i + \sum_{i=1}^{n}(b_i + (-1)^{\boldsymbol{x_i}} d + (-1)^{\boldsymbol{y_i}} e)S_i = 0\right] \\ &\leq \frac{1}{q} + \frac{1}{2^n}\end{aligned}$$

With a similar analysis, we can also show

$$\Pr[F = 0|b = 1] \leq \frac{1}{q} + \frac{1}{2^n}.$$

□

### E.3 Twin One-wayness of $\mathbb{F}_2^n$ Linear Relational Hash: Proof of Theorem 2

Below, Assumption 8 is the twin one-way security requirement of Theorem 2 and of our scheme.

**Assumption 8.** *Assuming a generation algorithm $\mathcal{G}$ that outputs a tuple $(n, q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ such that $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order $q$ and $e$ is a bilinear pairing mapping elements of $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$, then for random $\mathbf{g} \leftarrow \mathbb{G}_1, \mathbf{h} \leftarrow \mathbb{G}_2$, $\langle a_i \rangle_{i=1}^{n+1}$ , $\langle b_i \rangle_{i=1}^{n+1}$ , $r, s \leftarrow \mathbb{Z}_q$, $x \leftarrow \mathbb{F}_2^n$, given access to*

$$\left(\begin{array}{c} \mathbf{g}, \langle \mathbf{g}^{a_i} \rangle_{i=1}^{n+1}, \mathbf{g}^r, \left\langle \mathbf{g}^{(-1)^{x_i} a_i r} \right\rangle_{i=1}^{n}, \mathbf{g}^{a_{n+1} r} \\ \mathbf{h}, \left\langle \mathbf{h}^{b_i} \right\rangle_{i=1}^{n+1}, \mathbf{h}^s, \left\langle \mathbf{h}^{(-1)^{x_i} b_i s} \right\rangle_{i=1}^{n}, \mathbf{h}^{b_{n+1} s}, \\ k = \sum_{i=1}^{n+1} a_i b_i, e \end{array}\right),$$

*it is hard to output $x$.*

In this section we show Assumption 8 holds in the generic group model. In particular we show,

**Theorem 14.** *For random* $\mathbf{g} \leftarrow \mathbb{G}_1, \mathbf{h} \leftarrow \mathbb{G}_2$, $\langle a_i \rangle_{i=1}^{n+1}, \langle b_i \rangle_{i=1}^{n+1}, r, s \leftarrow \mathbb{Z}_q$, $x \leftarrow \mathbb{F}_2^n$, *if an adversary* $\mathcal{A}$ *is given access to*

$$\begin{pmatrix} \mathbf{g}, \langle \mathbf{g}^{a_i} \rangle_{i=1}^{n+1}, \mathbf{g}^r, \left\langle \mathbf{g}^{(-1)^{x_i} a_i r} \right\rangle_{i=1}^{n}, \mathbf{g}^{a_{n+1} r} \\ \mathbf{h}, \left\langle \mathbf{h}^{b_i} \right\rangle_{i=1}^{n+1}, \mathbf{h}^s, \left\langle \mathbf{h}^{(-1)^{x_i} b_i s} \right\rangle_{i=1}^{n}, \mathbf{h}^{b_{n+1} s}, \\ k = \sum_{i=1}^{n+1} a_i b_i \end{pmatrix},$$

*then in the generic group model the adversary* $\mathcal{A}$ *can output* $x$ *with probability at most* $1/2^n + 1/q + \left((2n+4+q_1)^2 + (2n+4+q_2)^2 + q_T^2\right)(2/(q-1) + 5/q + 1/2^n)$, *where* $q_1, q_2, q_T$ *are the number oracle queries by made by* $\mathcal{A}$ *to* $\mathbb{G}_1, \mathbb{G}_2$ *and* $\mathbb{G}_T$ *oracles.*

For a proof in the generic group model, following the techniques of Boneh et al. [BBS04] and Appendix E.2, we can give $2n+4$ random encodings for elements in each of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ to the adversary. We also need to provide a random integer $k \leftarrow \mathbb{Z}_q$ as the relational key $\sum_{i=1}^{n+1} a_i b_i$. $\left\langle \sigma_i^1 \right\rangle_{i=1}^{2n+4}$, $\left\langle \sigma_i^2 \right\rangle_{i=1}^{2n+4}$ be those random encodings corresponding to the elements in $\mathbb{G}_1$ and $\mathbb{G}_2$. We need to keep track of these encoded elements using functions $F_i^1, F_i^2, F_i^T$ in variables $A_1, \cdots, A_{n+1}, B_1, \cdots, B_n, R, S, X_1, \cdots, X_n$. We need to maintain three lists $L_1, L_2, L_T$ (one each for the groups $\mathbb{G}_1$ and $\mathbb{G}_2$, and one for the target group $\mathbb{G}_T$). The lists would be initialized as follows:

| $L_1$ | $L_2$ | $L_T$ |
|---|---|---|
| $(F_1^1 = 1, \quad \sigma_1^1)$ | $(F_1^2 = 1, \quad \sigma_1^2)$ | |
| $(F_2^1 = A_1, \quad \sigma_2^1)$ | $(F_2^2 = B_1, \quad \sigma_2^2)$ | |
| $\vdots$ | $\vdots$ | |
| $(F_{n+1}^1 = A_n, \quad \sigma_{n+1}^1)$ | $(F_{n+1}^2 = B_n, \quad \sigma_{n+1}^2)$ | |
| $(F_{n+2}^1 = A_{n+1}, \quad \sigma_{n+2}^1)$ | $(F_{n+2}^2 = A_{n+1}^{-1}\left(k - \sum_{i=1}^n A_i B_i\right), \quad \sigma_{n+2}^2)$ | |
| $(F_{n+3}^1 = R, \quad \sigma_{n+3}^1)$ | $(F_{n+3}^2 = S, \quad \sigma_{n+3}^2)$ | |
| $(F_{n+4}^1 = (-1)^{X_1} A_1 R, \quad \sigma_{n+4}^1)$ | $(F_{n+4}^2 = (-1)^{X_1} B_1 S, \quad \sigma_{n+4}^2)$ | |
| $\vdots$ | $\vdots$ | |
| $(F_{2n+3}^1 = (-1)^{X_n} A_n R, \sigma_{2n+3}^1)$ | $(F_{2n+3}^2 = (-1)^{X_n} B_n S, \quad \sigma_{2n+3}^2)$ | |
| $(F_{2n+4}^1 = A_{n+1} R, \quad \sigma_{2n+4}^1)$ | $(F_{2n+4}^2 = A_{n+1}^{-1}\left(k - \sum_{i=1}^n A_i B_i\right) S, \sigma_{2n+4}^2)$ | |

Now as the adversary will make queries to the group action oracles (in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$) , group inversion oracles (in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$) and the pairing oracle. We will be adding more elements to the above lists. Any new function $F_i^1$ in $L_1$ would be linear in $\mathbb{Z}_q$ in terms of $F_1^1, \cdots, F_{2n+4}^1$. Similarly any new function $F_i^2$ in $L_2$ would be linear in $\mathbb{Z}_q$ in terms of $F_1^2, \cdots, F_{2n+4}^2$. Moreover, any function $F_i^T$ in $L_T$ would be linear in $\mathbb{Z}_q$ in terms of $\left\langle F_{j_1}^1 F_{j_2}^2 \right\rangle_{1 \le j_1, j_2 \le 2n+4}$. If the adversary makes $q_1$ many group action and group inversion queries to $\mathbb{G}_1$ oracle, $q_2$ many group action and group inversion queries to $\mathbb{G}_2$ oracle and $q_T$ many queries to $\mathbb{G}_T$ oracle (i.e. pairing queries as well as group action and inversion queries), we know size of the lists $L_1, L_2, L_T$ can be upper bounded as follows:

$$|L_1| \le (2n+4) + q_1, |L_2| \le (2n+4) + q_2, |L_T| \le q_T.$$

In the end, the adversary will return some $x' \in \mathbb{F}_2^n$ and we will randomly assign values to

$$A_1, \cdots, A_{n+1}, B_1, \cdots, B_n, R, S \leftarrow \mathbb{Z}_q$$

and

$$X_1, \cdots, X_n \leftarrow \{0, 1\}.$$

Clearly $X_1 \cdots X_n$ will take the value $x'$ only with probability $1/2^n$. We will also abort the simulation, if the random assigned value of $A_{n+1}$ is zero which makes $A_{n+1}^{-1}$ undefined.

However, we also need show our simulation is consistent with high probability. That is after assigning random values to the variables the functions in $L_1$, $\left\langle F_{i_1}^1 \right\rangle_{1 \le i_1 \le |L_1|}$ should not collide (also in $L_2$ and $L_T$).

We follow an approach similar to Appendix E.2 to upper-bound collision probability in $L_T$. Any difference function $F^T_{i_1} - F^T_{i_2}$ (for $1 \leq i_1, i_2 \leq |L_T|$) can take the following form:

$$F^T_{i_1} - F^T_{i_2} = \sum_{1 \leq j_1, j_2 \leq 2n+4} \ell^{j_2}_{j_1} F^1_{j_1} F^2_{j_2},$$

where $\left\langle \ell^{j_2}_{j_1} \right\rangle_{1 \leq j_1, j_2 \leq 2n+4}$ are constants in $\mathbb{Z}_q$. We need to show for all possible constants $\left\langle \ell^{j_2}_{j_1} \right\rangle_{1 \leq j_1, j_2 \leq 2n+4} \in \mathbb{Z}_q$. either $F^T_{i_1} - F^T_{i_2}$ attains zero with negligible probability or $F^T_{i_1} - F^T_{i_2}$ is a trivially zero polynomial. We also observe, this upper-bound of $\Pr[F^T_{i_1} - F^T_{i_2} = 0]$ for all possible constants is also an upper-bound on $\Pr[F^1_{i_1} - F^1_{i_2} = 0]$, as well as $\Pr[F^2_{i_1} - F^2_{i_2} = 0]$. This is true because, for $1 \leq i_1, i_2 \leq |L_1|$, $F^1_{i_1} - F^1_{i_2}$ can be written as $\sum_{1 \leq j_1 \leq 2n+4} \ell^1_{j_1} F^1_{j_1} F^2_1$.

We observe, $F^T_{i_1} - F^T_{i_2}$ is a trivially zero polynomial, if for some $c_1, c_2 \in \mathbb{Z}_q$

$$\begin{aligned} \ell^1_1 = -kc_1, \ell^{n+3}_{n+3} &= -kc_2 \\ \ell^2_2 = \ell^3_3 = \cdots = \ell^{n+2}_{n+2} &= c_1 \\ \ell^{n+4}_{n+4} = \ell^{n+5}_{n+5} = \cdots = \ell^{2n+4}_{2n+4} &= c_2 \\ \text{if}(j_1 \neq j_2), \ell {j_1}^{j_2} &= 0. \end{aligned}$$

We have already mentioned, if $A_{n+1}$ becomes zero, then $A^{-1}_{n+1}$ is undefined and we abort the simulation. For rest of the analysis, we will assume $A_{n+1}$ is randomly sampled from $\mathbb{Z}_q \setminus \{0\}$.

For $1 \leq i, j \leq n, i \neq j$, let $\textsc{Coeff}(F^T_{i_1} - F^T_{i_2}, A_i B_j)$ be the coefficient of $A_i B_j$ in $F^T_{i_1} - F^T_{i_2}$. We have,

$$\begin{aligned} \textsc{Coeff}(F^T_{i_1} - F^T_{i_2}, A_i B_j) = &\left(\ell^{j+1}_{i+1} - \ell^{n+2}_{i+1} A_j A^{-1}_{n+1}\right) + \left(\ell^{n+j+3}_{i+1}(-1)^{X_j} - \ell^{2n+4}_{i+1} A_j A^{-1}_{n+1}\right) S \\ &+ \left(\ell^{j+1}_{n+i+3}(-1)^{X_i} - \ell^{n+2}_{n+i+3} A_j A^{-1}_{n+1}(-1)^{X_i}\right) R \\ &+ \left(\ell^{n+j+3}_{n+i+3}(-1)^{X_i \oplus X_j} - \ell^{2n+4}_{n+i+3}(-1)^{X_i} A_j A^{-1}_{n+1}\right) RS \end{aligned}$$

Unless $\ell^{j+1}_{i+1} = \ell^{n+2}_{i+1} = \ell^{n+j+3}_{i+1} = \ell^{2n+4}_{i+1} = \ell^{j+1}_{n+i+3} = \ell^{n+2}_{n+i+3} = \ell^{n+j+3}_{n+i+3} = \ell^{2n+4}_{n+i+3} = 0$, $\textsc{Coeff}(F^T_{i_1} - F^T_{i_2}, A_i B_j)$ can be a zero polynomial over $R$ and $S$ with probability at most $1/q$ (randomness over choice of $A_j$ and any fixed $A_{n+1}, X_i, X_j$). With probability at least $1 - 1/q$, $\textsc{Coeff}(F^T_{i_1} - F^T_{i_2}, A_i B_j)$ is a non-zero polynomial over $R$ and $S$ with maximum total degree 2. For a random choice of $R$ ans $S$, it can attain zero with probability at most $2/q$. Hence, for random choice of $A_j, R$ and $S$, $\textsc{Coeff}(F^T_{i_1} - F^T_{i_2}, A_i B_j)$ becomes zero with probability at most $3/q$. In other words, $(F^T_{i_1} - F^T_{i_2})[A_i, B_j]$ is a non-zero polynomial over $A_i, B_j$ (maximum total degree 2) with probability at least $(1 - 3/q)$ (for a random choice of other variables). For random choice of $A_i$ and $B_j$, this quadratic non-zero polynomial can attain zero with probability at most $2/q$. Hence, unless $\ell^{j+1}_{i+1} = \ell^{n+2}_{i+1} = \ell^{n+j+3}_{i+1} = \ell^{2n+4}_{i+1} = \ell^{j+1}_{n+i+3} = \ell^{n+2}_{n+i+3} = \ell^{n+j+3}_{n+i+3} = \ell^{2n+4}_{n+i+3} = 0$ for $1 \leq i, j \leq n, i \neq j$, $F^T_{i_1} - F^T_{i_2}$ attains zero with probability at most $5/q$.

**Case A: Figure 2** Let us consider the case when $\ell^{j+1}_{i+1} = \ell^{n+2}_{i+1} = \ell^{n+j+3}_{i+1} = \ell^{2n+4}_{i+1} = \ell^{j+1}_{n+i+3} = \ell^{n+2}_{n+i+3} = \ell^{n+j+3}_{n+i+3} = \ell^{2n+4}_{n+i+3} = 0$ for $1 \leq i, j \leq n, i \neq j$. Figure 2 shows the $\ell^j_i$ values, where darkened squares denote zero.

For $1 \leq i \leq n$, let $\textsc{Coeff}(F^T_{i_1} - F^T_{i_2}, A_i)$ be the coefficient of $A_i$ in $F^T_{i_1} - F^T_{i_2}$. Now, ignoring the darkened zero $\ell^j_i$ values as per Figure 2 we have,

$$\begin{aligned} \textsc{Coeff}(F^T_{i_1} - F^T_{i_2}, A_i) = &\left(\ell^1_{i+1} + \left(\ell^{i+1}_{i+1} - \ell^{n+2}_{n+2} - \ell^{n+2}_1 A^{-1}_{n+1}\right) B_i\right) \\ &+ \left(\ell^{n+3}_{i+1} + \left(\ell^{n+i+3}_{i+1}(-1)^{X_i} - \ell^{2n+4}_{n+2} - \ell^{2n+4}_1 A^{-1}_{n+1}\right) B_i\right) S \\ &+ \left(\ell^1_{n+i+3}(-1)^{X_i} + \left(\ell^{i+1}_{n+i+3}(-1)^{X_i} - \ell^{n+2}_{2n+4} - \ell^{n+2}_{n+3} A^{-1}_{n+1}\right) B_i\right) R \\ &+ \left(\ell^{n+3}_{n+i+3}(-1)^{X_i} + \left(\ell^{n+i+3}_{n+i+3} - \ell^{2n+4}_{2n+4} - \ell^{2n+4}_{n+3} A^{-1}_{n+1}\right) B_i\right) RS \end{aligned}$$

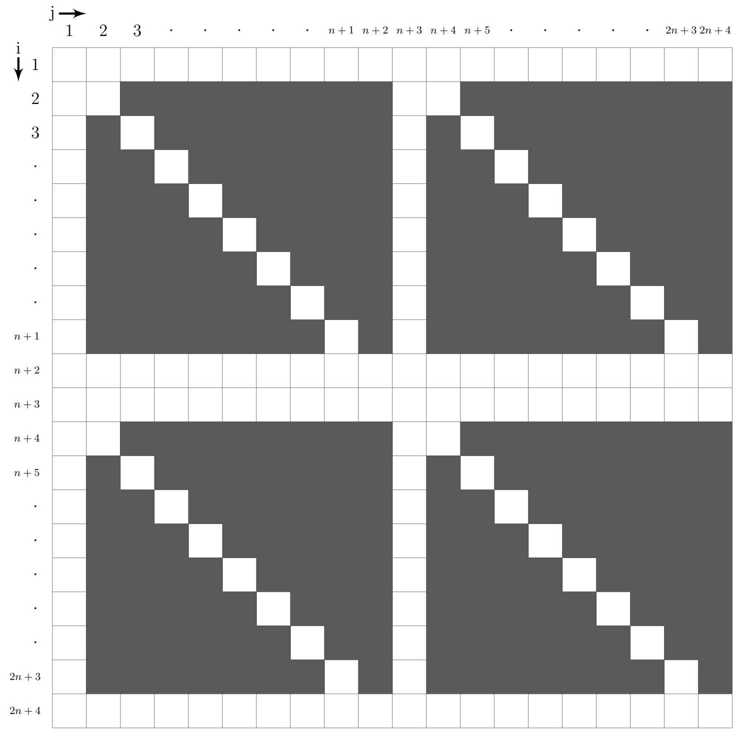


**Fig. 2.** Case A: Grid of $\ell_i^j$ values, darkened squares denote zero

Unless, $\ell_{i+1}^1 = \ell_{i+1}^{n+3} = \ell_{n+i+3}^1 = \ell_{n+i+3}^{n+3} = 0$, for a random choice of $B_i$ and any choice of $A_{n+1}^{-1}$ and $X_i$, $\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, A_i)[R, S]$ will be a non zero polynomial (of maximum total degree 2) over $R, S$ with probability at least $1 - 1/q$.

Now, if we consider the case $\ell_{i+1}^1 = \ell_{i+1}^{n+3} = \ell_{n+i+3}^1 = \ell_{n+i+3}^{n+3} = 0$, unless $\ell_1^{n+2} = \ell_1^{2n+4} = \ell_{n+3}^{n+2} = \ell_{n+3}^{2n+4} = 0$ for any choice of $X_i$ and random choice of $A_{n+1}^{-1}$ and $B_i$, $\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, A_i)[R, S]$ will be a non zero polynomial (of maximum total degree 2) over $R, S$ with probability at least $1 - (1/(q-1) + 1/q)$.

In the case $\ell_{i+1}^1 = \ell_{i+1}^{n+3} = \ell_{n+i+3}^1 = \ell_{n+i+3}^{n+3} = \ell_1^{n+2} = \ell_1^{2n+4} = \ell_{n+3}^{n+2} = \ell_{n+3}^{2n+4} = 0$, unless $\ell_{i+1}^{i+1} = \ell_{n+2}^{n+2}$ and $\ell_{n+i+3}^{n+i+3} = \ell_{2n+4}^{2n+4}$ for any choice of $X_i$ and random choice of $B_i$, $\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, A_i)[R, S]$ will be a non zero polynomial (of maximum total degree 2) over $R, S$ with probability at least $1 - (1/q)$.

All together, we have unless $\ell_{i+1}^1 = \ell_{i+1}^{n+3} = \ell_{n+i+3}^1 = \ell_{n+i+3}^{n+3} = \ell_1^{n+2} = \ell_1^{2n+4} = \ell_{n+3}^{n+2} = \ell_{n+3}^{2n+4} = 0$, $\ell_{i+1}^{i+1} = \ell_{n+2}^{n+2}$ and $\ell_{n+i+3}^{n+i+3} = \ell_{2n+4}^{2n+4}$ for any choice of $X_i$ and random choice of $B_i$ and $A_{n+1}^{-1}$, $\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, A_i)[R, S]$ is a non zero polynomial (of maximum total degree 2) over $R, S$ with probability at least $1-(1/(q-1)+1/q)$. For random choices of $R$ and $S$, the non zero polynomial $\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, A_i)[R, S]$ can attain zero with probability at most $2/q$. Hence, with probability at least $1 - (1/(q-1) + 3/q)$, $(F_{i_1}^T - F_{i_2}^T)[A_i]$ is a non zero polynomial over $A_i$ (the probability is taken over all the random variables except $A_i$). This non-zero polynomial has degree at most 2, and for random choice of $A_i$ this can attain zero with probability at most $2/q$. Hence, unless $\ell_{i+1}^1 = \ell_{i+1}^{n+3} = \ell_{n+i+3}^1 = \ell_{n+i+3}^{n+3} = \ell_1^{n+2} = \ell_1^{2n+4} = \ell_{n+3}^{n+2} = \ell_{n+3}^{2n+4} = 0$, $\ell_{i+1}^{i+1} = \ell_{n+2}^{n+2}$ and $\ell_{n+i+3}^{n+i+3} = \ell_{2n+4}^{2n+4}$ for $1 \le i \le n$, $F_{i_1}^T - F_{i_2}^T$ can attain zero with probability at most $1/(q-1) + 5/q$.

**Case B: Figure 3** Let us consider the case when

$$\ell_{i+1}^1 = \ell_{i+1}^{n+3} = \ell_{n+i+3}^1 = \ell_{n+i+3}^{n+3} = \ell_1^{n+2} = \ell_1^{2n+4} = \ell_{n+3}^{n+2} = \ell_{n+3}^{2n+4} = 0$$
$$\ell_{i+1}^{i+1} = \ell_{n+2}^{n+2} = c_1, \ell_{n+i+3}^{n+i+3} = \ell_{2n+4}^{2n+4} = c_2$$

for $1 \leq i \leq n$ and arbitrary constants $c_1, c_2$. Figure 3 shows the $\ell_i^j$ values, where darkened squares denote zero.

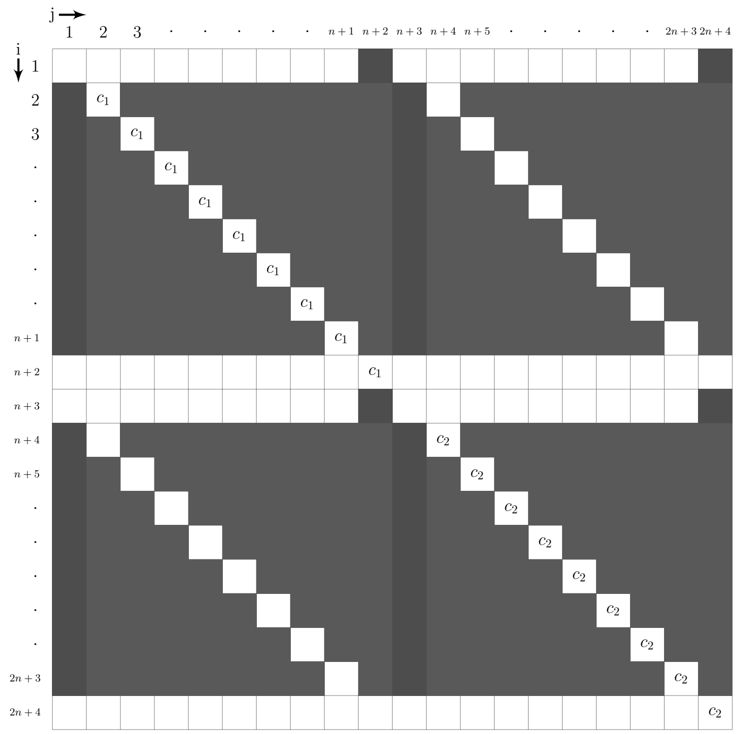

**Fig. 3.** Case B: Grid of $\ell_i^j$ values, darkened squares denote zero

In this case, we can simplify $\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, A_i)$, as follows

$$\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, A_i) = \left(\left(\ell_{i+1}^{n+i+3}(-1)^{X_i} - \ell_{n+2}^{2n+4}\right) S + \left(\ell_{n+i+3}^{i+1}(-1)^{X_i} - \ell_{2n+4}^{n+2}\right) R\right) B_i.$$

Unless, $\ell_{i+1}^{n+i+3} = \ell_{n+i+3}^{i+1} = \ell_{n+2}^{2n+4} = \ell_{2n+4}^{n+2} = 0$ for $1 \leq i \leq n$, for random choices of $X_1, \cdots, X_n$ with probability at least $1 - 1/2^n$, there exists $i^* \in \{1, \cdots, n\}$ such that $\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, A_{i^*})$ is a non-zero polynomial over $R, S$ and $B_{i^*}$, with maximum total degree 2. For random choices of $R, S$ and $B_{i^*}$ the non-zero polynomial $\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, A_{i^*})[R, S, B_{i^*}]$ can attain zero with probability at most $2/q$, which would imply $F_{i_1}^T - F_{i_2}^T$ can attain zero with probability at most $1/2^n + 4/q$.

**Case C: Figure 4** Let us consider the case

$$\ell_{i+1}^{n+i+3} = \ell_{n+i+3}^{i+1} = \ell_{n+2}^{2n+4} = \ell_{2n+4}^{n+2} = 0 \text{ for } 1 \leq i \leq n.$$

Figure 4 shows the $\ell_i^j$ values, where darkened squares denote zero.

For $1 \leq j \leq n$, let $\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, B_j)$ be the coefficient of $B_j$ in $F_{i_1}^T - F_{i_2}^T$. Now, using $\ell_i^j$ values as per Figure 4 we have,

$$\begin{aligned}\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, B_j) = &\left(\ell_1^{j+1} + \ell_{n+2}^{j+1} A_{n+1}\right) + \left(\ell_{n+3}^{j+1} + \ell_{2n+4}^{j+1} A_{n+1}\right) R \\ &+ \left(\ell_1^{n+j+3} + \ell_{n+2}^{n+j+3} A_{n+1}\right) (-1)^{X_j} S + \left(\ell_{n+3}^{n+j+3} + \ell_{2n+4}^{n+j+3} A_{n+1}\right) (-1)^{X_j} RS\end{aligned}$$

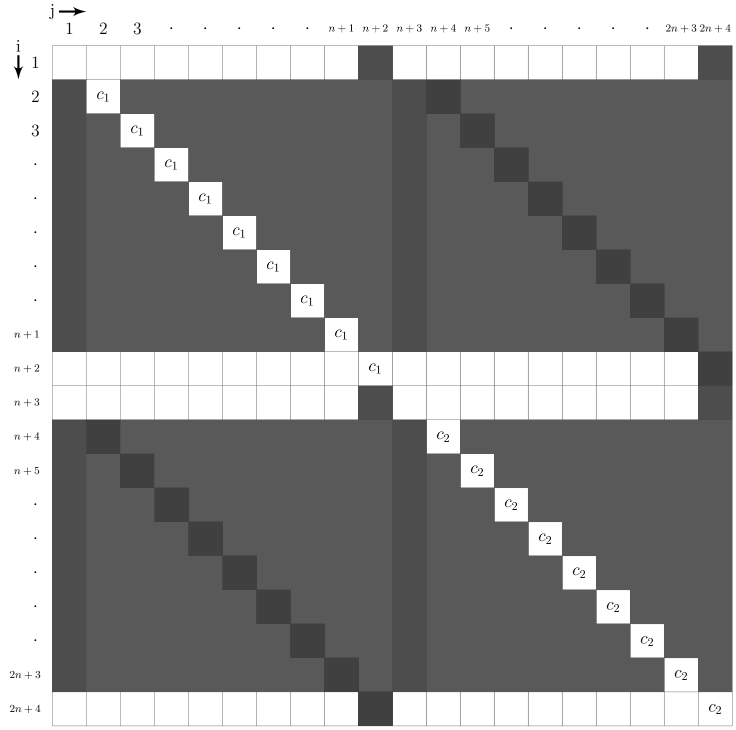

**Fig. 4.** Case C: Grid of $\ell_i^j$ values, darkened squares denote zero

With a similar argument as before, we can show unless for $1 \leq j \leq n$, $\ell_1^{j+1} = \ell_{n+2}^{j+1} = \ell_{n+3}^{j+1} = \ell_{2n+4}^{j+1} = \ell_1^{n+j+3} = \ell_{n+2}^{n+j+3} = \ell_{n+3}^{n+j+3} = \ell_{2n+4}^{n+j+3} = 0$, $F_{i_1}^T - F_{i_2}^T$ can attain zero with probability at most $1/(q-1) + 3/q$.

**Case D: Figure 5** Let us consider the case

$$\ell_1^{j+1} = \ell_{n+2}^{j+1} = \ell_{n+3}^{j+1} = \ell_{2n+4}^{j+1} = \ell_1^{n+j+3} = \ell_{n+2}^{n+j+3} = \ell_{n+3}^{n+j+3} = \ell_{2n+4}^{n+j+3} = 0,$$

for $1 \leq j \leq n$. Figure 5 shows the $\ell_i^j$ values, where darkened squares denote zero.

Let $\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, S)$ be the coefficient of $S$ in $F_{i_1}^T - F_{i_2}^T$. Now, using $\ell_i^j$ values as per Figure 5 we have,

$$\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, S) = \ell_1^{n+3} + \ell_{n+2}^{n+3} A_{n+1} + \left(\ell_{n+3}^{n+3} + kc_2\right) R + \ell_{2n+4}^{n+3} A_{n+1} R$$

With a similar argument as before, we can show unless $\ell_1^{n+3} = \ell_{n+2}^{n+3} = \ell_{2n+4}^{n+3} = 0$ and $\ell_{n+3}^{n+3} = -kc_2$, $F_{i_1}^T - F_{i_2}^T$ can attain zero with probability at most $1/(q-1) + 2/q$.

**Case E: Figure 6** Let us consider the case

$$\ell_1^{n+3} = \ell_{n+2}^{n+3} = \ell_{2n+4}^{n+3} = 0 \text{ and } \ell_{n+3}^{n+3} = -kc_2.$$

Figure 6 shows the $\ell_i^j$ values, where darkened squares denote zero.

Let $\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, R)$ be the coefficient of $R$ in $F_{i_1}^T - F_{i_2}^T$. Now, using $\ell_i^j$ values as per Figure 6 we have,

$$\textsc{Coeff}(F_{i_1}^T - F_{i_2}^T, R) = \ell_{n+3}^1 + \ell_{2n+4}^1 A_{n+1}.$$

With a similar argument as before, we can show unless $\ell_{n+3}^1 = \ell_{2n+4}^1 = 0$, $F_{i_1}^T - F_{i_2}^T$ can attain zero with probability at most $1/(q-1) + 1/q$.

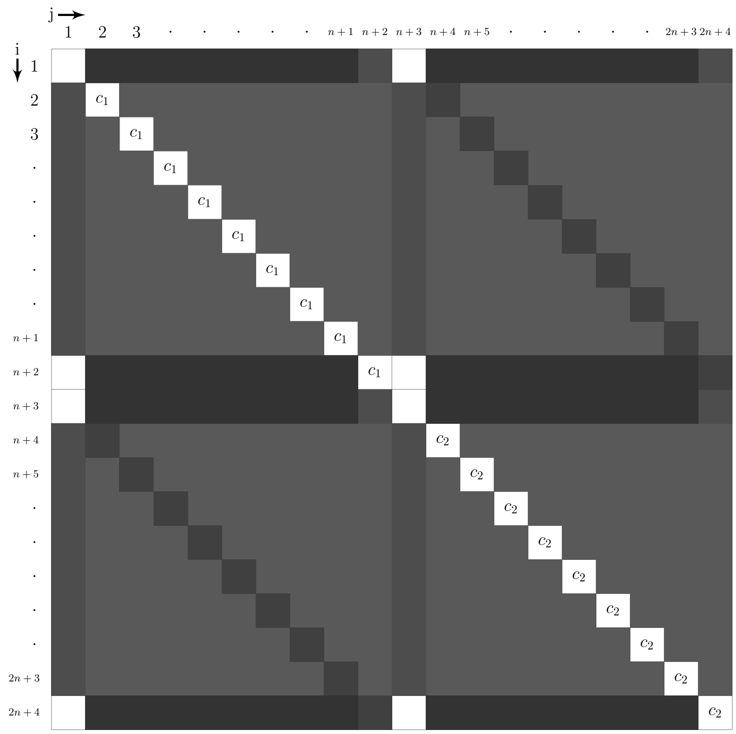

**Fig. 5.** Case D: Grid of $\ell_i^j$ values, darkened squares denote zero

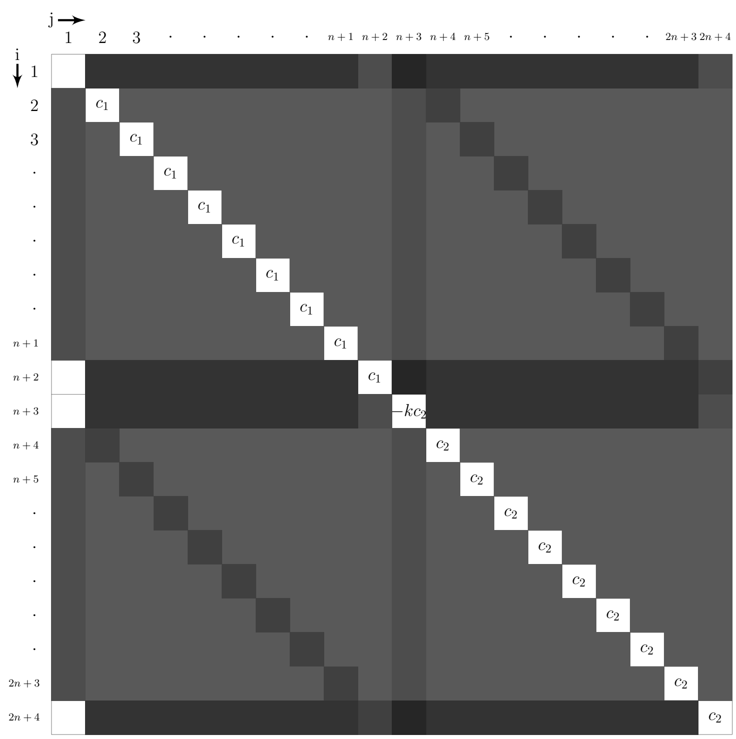

**Fig. 6.** Case E: Grid of $\ell_i^j$ values, darkened squares denote zero

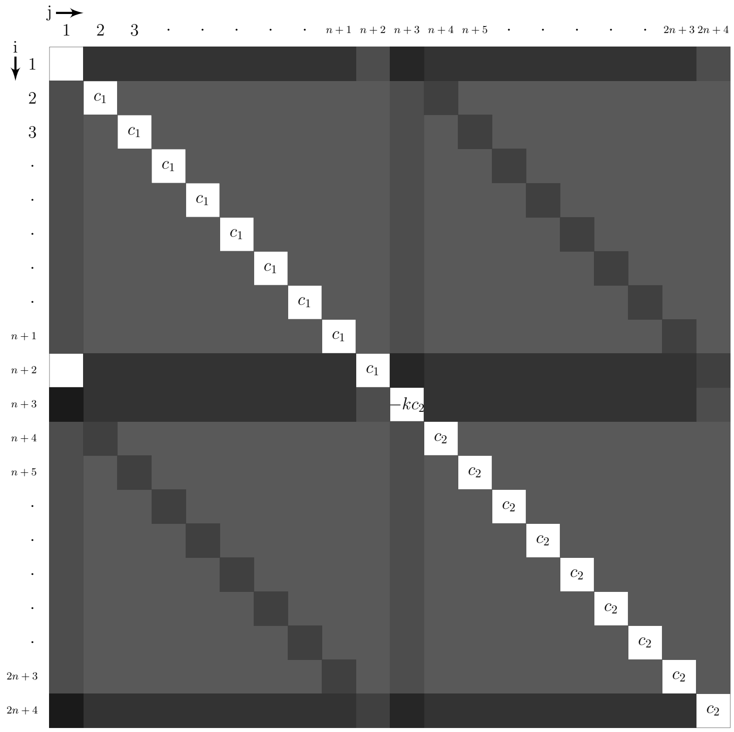


**Fig. 7.** Case F: Grid of $\ell_i^j$ values, darkened squares denote zero

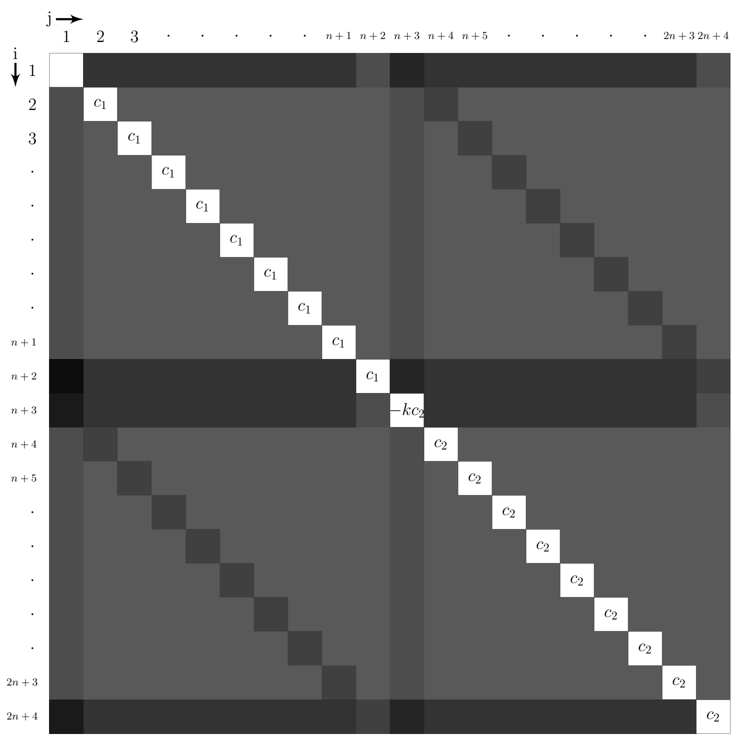


**Fig. 8.** Case G: Grid of $\ell_i^j$ values, darkened squares denote zero

**Case F: Figure 7** Let us consider the case

$$\ell^1_{n+3} = \ell^1_{2n+4} = 0.$$

Figure 7 shows the $\ell^j_i$ values, where darkened squares denote zero.

Let $\textsc{Coeff}(F^T_{i_1} - F^T_{i_2}, A_{n+1})$ be the coefficient of $A_{n+1}$ in $F^T_{i_1} - F^T_{i_2}$. Now, using $\ell^j_i$ values as per Figure 7 we have,

$$\textsc{Coeff}(F^T_{i_1} - F^T_{i_2}, A_{n+1}) = \ell^1_{n+2}.$$

Hence, unless $\ell^1_{n+2} = 0$, $F^T_{i_1} - F^T_{i_2}$ being quadratic polynomial in $A_{n+1}$ can attain zero with probability at most $2/(q-1)$.

**Case G: Figure 8** Let us consider the case, $\ell^1_{n+2} = 0$. Figure 8 shows the $\ell^j_i$ values, where darkened squares denote zero.

Finally, using $\ell^j_i$ values as per Figure 8 we have,

$$F^T_{i_1} - F^T_{i_2} = \ell^1_1 + kc_1.$$

If $\ell^1_1 \neq -kc_1$, $F^T_{i_1} - F^T_{i_2}$ can never attain zero, otherwise we have

$$\begin{aligned}
\ell^1_1 = -kc_1, \ell^{n+3}_{n+3} &= -kc_2 \\
\ell^2_2 = \ell^3_3 = \cdots = \ell^{n+2}_{n+2} &= c_1 \\
\ell^{n+4}_{n+4} = \ell^{n+5}_{n+5} = \cdots = \ell^{2n+4}_{2n+4} &= c_2 \\
\text{if}(j_1 \neq j_2), \ell {j_1}^{j_2} &= 0,
\end{aligned}$$

the condition for trivial zero.

Hence for non-trivial choice $\ell^j_i$ constants, $\Pr[F^T_{i_1} - F^T_{i_2} = 0]$ can be bounded by $\big(2/(q-1) + 5/q + 1/2^n\big)$, and collision probability in either of the lists $L_1, L_2$ or $L_T$ gets bounded by $\big((2n+4+q_1)^2 + (2n+4+q_2)^2 + q_T^2\big)\big(2/(q-1) + 5/q + 1/2^n\big)$. Our simulator also aborts when $A_{n+1}$ is zero, which happens with probability $1/q$. $X_1 \cdots X_n$ can match the attackers guess $x'$ only with probability $1/2^n$. Hence, all together attacker $\mathcal{A}$'s success probability gets bounded by $1/2^n + 1/q + \big((2n+4+q_1)^2 + (2n+4+q_2)^2 + q_T^2\big)\big(2/(q-1) + 5/q + 1/2^n\big)$. $\square$

## F Correctness and Security of the Proximity Relational Hash: Proofs of Theorem 3 and Theorem 4

*Correctness.* For any $x, y \in \mathbb{F}_2^n$, we have

$$\begin{aligned}
&\textsc{Hash}_1(x) = (hx_1, hx_2) = (x + \textsc{Encode}(r), \textsc{HashLinear}_1(pk_{lin}, r)) && \text{for some random } r \in \mathbb{F}_2^k \\
&\textsc{Hash}_2(y) = (hy_1, hy_2) = (y + \textsc{Encode}(s), \textsc{HashLinear}_2(pk_{lin}, s)) && \text{for some random } s \in \mathbb{F}_2^k
\end{aligned}$$

If $\texttt{dist}(x,y) \leq \delta$, $\textsc{Decode}(hx_1 + hy_1)$ will output $(r+s)$ and the tuple

$$(\textsc{HashLinear}_1(pk_{lin}, r), \textsc{HashLinear}_2(pk_{lin}, s), r+s)$$

will get verified by $\textsc{VerifyLinear}$. This shows the above proximity hash correctly verifies tuples $(x, y)$, for any $x, y \in \mathbb{F}_2^n$ and $\texttt{dist}(x,y) \leq \delta$.

On the other hand, if $\texttt{dist}(x,y) > \delta$ and $\textsc{Verify}$ outputs $\textsc{Accept}$, then the output of $z = \textsc{Decode}(hx_1 + hy_1)$ can never be same as $(r+s)$, because $\texttt{dist}(\textsc{Encode}(r+s), hx_1 + hy_1) = \texttt{dist}(x,y) > \delta$. Also, from correctness of linear relational hash we know $\textsc{VerifyLinear}(pk_{lin}, hx_2, hy_2, z)$ outputs $\textsc{Accept}$ only with negligible probability (for any $z \neq r+s$). Hence the above algorithms constitute a correct relational hash for proximity over $\mathbb{F}_2^n$.

*One-wayness.* We show that if there exists an attacker $A$ breaking one-way security (Definition 3) for the proximity hash scheme with non-negligible probability, then we can build an attacker which breaks the one-way security for the linear relational hash scheme with non-negligible probability.

Let $(pk_{lin}, hx_{lin} = \textsc{HashLinear}_1(pk_{lin}, r))$ be the linear relational hash challenge for some random $r \leftarrow \mathbb{F}_2^k$. We choose random $x' \leftarrow \mathbb{F}_2^n$, and give

$$pk := (\textsc{Encode}, \textsc{Decode}, pk_{lin})$$
$$hx := (x', hx_{lin})$$

to the attacker $A$. Clearly $hx$ is indistinguishable from a proximity hash of a random $m \leftarrow \mathbb{F}_2^n$. If $A$ breaks one-wayness of the proximity hash with non-negligible probability and outputs $m'$, then we can output

$$\textsc{Decode}(m' + x').$$

With non-negligible probability this value will be same as $r$, breaking one-wayness of linear relational hash.

*Unforgeability.* We show if there exists an attacker $A$ breaking unforgeability (Definition 5) for the proximity hash scheme with non-negligible probability, then we can build an attacker which breaks the unforgeability security property for the linear relational hash scheme with non-negligible probability.

Let $(pk_{lin}, hx_{lin} = \textsc{HashLinear}_1(pk_{lin}, r))$ be the linear relational hash challenge for some random $r \leftarrow \mathbb{F}_2^k$. We choose random $x' \leftarrow \mathbb{F}_2^n$, and give

$$pk := (\textsc{Encode}, \textsc{Decode}, pk_{lin})$$
$$hx := (x', hx_{lin})$$

to the attacker $A$. Clearly $hx$ is indistinguishable from a proximity hash of a random $m \leftarrow \mathbb{F}_2^n$. If $A$ breaks unforgeability of the proximity hash with non-negligible probability and outputs $(m', hy_{lin})$, then we know $\textsc{VerifyLinear}$ must accept the input $(pk_{lin}, hx_{lin}, hy_{lin}, \textsc{Decode}(x' + m'))$. Hence,

$$(hy_{lin}, \textsc{Decode}(x' + m'))$$

will be a valid forgery breaking the linear relational hash challenge. □

*Proof (of Theorem 4).* For $x \leftarrow \mathbb{F}_2^n$ and $e_1, \cdots, e_t \leftarrow \Xi$, suppose there exists an attacker $A$, such that given $(\textsc{Hash}_1(x), \textsc{Hash}_2(x+e_1)), \cdots, \textsc{Hash}_2(x+e_t))$, $A$ can output $x' \in \mathbb{F}_2^n$ with non-negligible probability satisfying the condition $\texttt{dist}(x', x) \leq \delta$. Theorem 2 says the linear relation hash is twin one-way secure if Assumption 8 holds. Now we show, using $A$, we can construct another attacker $B$, which breaks Theorem 2 in Section 3. Attacker $B$ has access to Theorem 2 challenge

$$(pk_{lin}, hx_{lin} = \textsc{HashLinear}_1(pk_{lin}, r), hy_{lin} = \textsc{HashLinear}_2(pk_{lin}, r)),$$

for some random $r \in \mathbb{F}_2^k$. $B$ needs to output $r$ with non-negligible probability. Attacker $B$ chooses random $x_0 \leftarrow \mathbb{F}_2^n$, random $s_1, \cdots, s_t \leftarrow \mathbb{F}_2^k$ and error samples $e_1, \cdots, e_t \leftarrow \Xi$. In Remark 2 (Section 3), we observed that attacker $B$ can easily construct $\textsc{HashLinear}_2(pk_{lin}, r+s)$ from $s$ and $\textsc{HashLinear}_2(pk_{lin}, r)$. Let us denote the implicit (to $B$) quantity $x_0 + \textsc{Encode}(r)$ by $\tilde{x}$. Attacker $B$, sends the following challenge to $A$,

$$pk := (\textsc{Encode}, \textsc{Decode}, pk_{lin})$$
$$\textsc{Hash}_1(\tilde{x}) := (x_0, \textsc{HashLinear}_1(pk_{lin}, r))$$
$$\textsc{Hash}_2(\tilde{x} + e_1) := (x_0 + e_1 + \textsc{Encode}(s_1), \textsc{HashLinear}_2(pk_{lin}, r + s_1)).$$
$$\cdots$$
$$\textsc{Hash}_2(\tilde{x} + e_t) := (x_0 + e_t + \textsc{Encode}(s_t), \textsc{HashLinear}_2(pk_{lin}, r + s_t)).$$

If attacker $A$ outputs $m' \in \mathbb{F}_2^n$, with non-negligible probability we have

$$\texttt{dist}(m', \tilde{x}) \leq \delta,$$

or equivalently $\texttt{dist}(m' + x_0, \textsc{Encode}(r)) \leq \delta$. Hence, with non-negligible probability $\textsc{Decode}(m' + x_0)$ will be same as $r$. □

## G 2-POW implies Oracle Simulation: Proof of Theorem 7

We, recall in Definition 6 Oracle Simulation Security implies having access to hash value $\textsc{Hash}_1(pk, x)$ is not more useful than a relational oracle $R_x(\cdot,\cdot)$ for predicting $P(pk, x)$, for all predicates $P$. At first, we prove Lemma 8, which says having access to a relational hash value, which is 2-POW is not really helpful for predicting the value of any predicate.

**Lemma 8.** *If a probabilistic function family $\{h_k\}_{k \in K}$ with domain $X$ and randomness space $U$ is 2-value perfectly One-Way with respect to probability distributions $\mathcal{X}$ (over $X$) and $\mathcal{K}$ (over $K$), then for all predicates $P(\cdot,\cdot)$ and all PPTs $A$:*

$$|\Pr[A(k, h_k(x, r)) = P(k, x)] - \Pr[A(k, h_k(x', r)) = P(k, x)]| \leq \texttt{negl}(\lambda).$$

*Here, $x$ and $x'$ are independently sampled from $\mathcal{X}$, $k \leftarrow \mathcal{K}$ and $r$ comes from a uniform distribution over randomness space $U$*

*Proof.* Let $D(k, y_0, y_1)$ be the distinguisher that outputs 1 iff $A(y_0) = A(y_1)$. For every $x$ and $k$ define $Q_{x,k} \stackrel{\text{def}}{=} \Pr[A(k, h_k(x, r)) = 1]$. Now we have,

$$\begin{aligned}
&|\Pr[A(k, h_k(x, r)) = P(k, x)] - \Pr[A(k, h_k(x', r)) = P(k, x)]| \\
&\leq \Delta(\ \langle A(k, h_k(x, r)), k, x\rangle\ ,\ \langle A(k, h_k(x', r)), k, x\rangle\ ) \\
&= \text{Exp}_{x,k}[|Q_{x,k} - \text{Exp}_x[Q_{x,k}]|] \\
&\leq \text{Exp}_k\left[\sqrt{\text{Var}_x[Q_{x,k}]}\right] \\
&\leq \sqrt{\text{Exp}_k\left[\text{Var}_x[Q_{x,k}]\right]} \\
&= \sqrt{\text{Exp}_{k,x}\left[Q_{x,k}^2\right] - \text{Exp}_k[\text{Exp}_x[Q_{x,k}]^2]} \\
&= \sqrt{\frac{1}{2}\left|\Pr[D(k, h_k(x, r_1), h_k(x, r_2)) = 1] - \Pr[D(k, h_k(x_1, r_1), h_k(x_2, r_2)) = 1]\right|}
\end{aligned}$$

□

Now we prove Theorem 7. Let $C$ be an adversary which given $pk, \textsc{Hash}_1(pk, x), \textsc{Hash}_2(pk, y)$ outputs a single bit. Let S be the adversary that, given $pk$, randomly selects $x' \leftarrow \mathcal{X}$ and $y' \leftarrow \mathcal{Y}$, and outputs $C(pk, \textsc{Hash}_1(pk, x'), \textsc{Hash}_2(pk, y'))$.

We now have,

$$\begin{aligned}
&\left|\begin{matrix}\Pr[C(pk, \textsc{Hash}_1(pk, x), \textsc{Hash}_2(pk, y)) = P(pk, x, y)] \\ - \Pr[S^{R_x, R_y, R_{x,y}}(pk) = P(pk, x, y)]\end{matrix}\right| \\
&= \left|\begin{matrix}\Pr[C(pk, \textsc{Hash}_1(pk, x), \textsc{Hash}_2(pk, y)) = P(pk, x, y)] \\ - \Pr[C(pk, \textsc{Hash}_1(pk, x'), \textsc{Hash}_2(pk, y')) = P(pk, x, y)]\end{matrix}\right| \\
&\leq \left|\begin{matrix}\Pr[C(pk, \textsc{Hash}_1(pk, x), \textsc{Hash}_2(pk, y)) = P(pk, x, y)] \\ - \Pr[C(pk, \textsc{Hash}_1(pk, x), \textsc{Hash}_2(pk, y')) = P(pk, x, y)]\end{matrix}\right| + \texttt{negl}(\lambda) \\
&\qquad \text{(Since } \textsc{Hash}_1 \text{ is a 2-value POW and by Lemma 8.)} \\
&\leq \texttt{negl}(\lambda) \\
&\qquad \text{(Since } \textsc{Hash}_2 \text{ is a 2-value POW and by Lemma 8.)}
\end{aligned}$$

□

## H 2-POW Property of the Linear Relational Hash: Proof of Theorem 9

We want to show that, for randomly chosen $\mathbf{g}_0 \leftarrow \mathbb{G}_1, \mathbf{h}_0 \leftarrow \mathbb{G}_2, \langle a_i \rangle_{i=1}^{n+1}, \langle b_i \rangle_{i=1}^{n+1} \leftarrow \mathbb{Z}_q^*$, if we define $\mathbf{g}_i = \mathbf{g}_0^{a_i}$, $\mathbf{h}_i = \mathbf{h}_0^{b_i}$, $pk_R = \sum_{i=1}^{n+1} a_i b_i$ and $pk = (\langle \mathbf{g}_i \rangle_{i=0}^{n+1}, \langle \mathbf{h}_i \rangle_{i=0}^{n+1}, pk_R)$, then under the Decisional Binary Mix(Assumption 2) and DDH assumptions, the following distributions are computationally indistinguishable given random $r, s \leftarrow \mathbb{Z}_q^*$ and random $x, y \leftarrow \mathbb{F}_2^n$:

$$\left(pk, \quad \mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^{n}, \mathbf{g}_{n+1}^r, \quad \mathbf{g}_0^s, \left\langle \mathbf{g}_i^{(-1)^{x_i} s} \right\rangle_{i=1}^{n}, \mathbf{g}_{n+1}^s \right)$$

and

$$\left(pk, \quad \mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^{n}, \mathbf{g}_{n+1}^r, \quad \mathbf{g}_0^s, \left\langle \mathbf{g}_i^{(-1)^{y_i} s} \right\rangle_{i=1}^{n}, \mathbf{g}_{n+1}^s \right).$$

We start with the following lemma.

**Lemma 9.** *Under the Decisional Binary Mix(Assumption 2) and DDH assumptions, the following distributions are computationally indistinguishable given random elements $\mathbf{g}_0, \langle \mathbf{g}_i \rangle_{i=1}^n \leftarrow \mathbb{G}$ and $r, s \leftarrow \mathbb{Z}_q^*$ and random $x, y \leftarrow \mathbb{F}_2^n$ :*

$$\left( \mathbf{g}_0, \langle \mathbf{g}_i \rangle_{i=1}^n, \quad \mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^{n}, \prod_{i=1}^n \mathbf{g}_i^r, \quad \mathbf{g}_0^s, \left\langle \mathbf{g}_i^{(-1)^{x_i} s} \right\rangle_{i=1}^{n}, \prod_{i=1}^n \mathbf{g}_i^s \right)$$

*and*

$$\left( \mathbf{g}_0, \langle \mathbf{g}_i \rangle_{i=1}^n, \quad \mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^{n}, \prod_{i=1}^n \mathbf{g}_i^r, \quad \mathbf{g}_0^s, \left\langle \mathbf{g}_i^{(-1)^{y_i} s} \right\rangle_{i=1}^{n}, \prod_{i=1}^n \mathbf{g}_i^s \right).$$

*Proof.* We show that the following distributions are indistinguishable:

$$Dist_0 \stackrel{\text{def}}{=} \left( \mathbf{g}_0, \langle \mathbf{g}_i \rangle_{i=1}^n, \quad \mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^{n}, \prod_{i=1}^n \mathbf{g}_i^r, \quad \mathbf{g}_0^s, \left\langle \mathbf{g}_i^{(-1)^{x_i} s} \right\rangle_{i=1}^{n}, \prod_{i=1}^n \mathbf{g}_i^s \right)$$

and

$$Dist_0' \stackrel{\text{def}}{=} \left( \mathbf{g}_0, \langle \mathbf{g}_i \rangle_{i=1}^n, \quad \mathbf{f}_0, \langle \mathbf{f}_i \rangle_{i=1}^n, \prod_{i=1}^n \mathbf{f}_i^{(-1)^{x_i}}, \quad \mathbf{h}_0, \langle \mathbf{h}_i \rangle_{i=1}^n, \prod_{i=1}^n \mathbf{h}_i^{(-1)^{x_i}} \right),$$

where the $\mathbf{g}_i, \mathbf{f}_i$ and $\mathbf{h}_i$'s are sampled independently randomly.

Let

$$Dist_{0,k} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{g}_0, \langle \mathbf{g}_i \rangle_{i=1}^n, \\ \mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^{k-1}, \langle \mathbf{f}_i \rangle_{i=k}^n, \prod_{i=1}^{k-1} \mathbf{g}_i^r \cdot \prod_{i=k}^n \mathbf{f}_i^{(-1)^{x_i}}, \\ \mathbf{g}_0^s, \left\langle \mathbf{g}_i^{(-1)^{x_i} s} \right\rangle_{i=1}^{k-1}, \langle \mathbf{h}_i \rangle_{i=k}^n, \prod_{i=1}^{k-1} \mathbf{g}_i^s \cdot \prod_{i=k}^n \mathbf{h}_i^{(-1)^{x_i}} \end{pmatrix}$$

Suppose a DDH instance $(\mathbf{u}, \mathbf{v}, \mathbf{u}^r, \mathbf{w})$ is given, where the challenge is to decide whether $\mathbf{w}$ is $\mathbf{v}^r$ or random. We construct the following distribution, after choosing $s$ and $u_i$'s randomly from $\mathbb{Z}_q^*$, $\mathbf{f}_i$ and $\mathbf{h}_i$'s randomly from $\mathbb{G}$ and $x$ randomly from $\mathbb{F}_2^n$:

$$Dist_{0,k,DDH} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{u}, \langle \mathbf{u}^{u_i} \rangle_{i=1}^{k-1}, \mathbf{v}, \langle \mathbf{u}^{u_i} \rangle_{i=k+1}^n, \\ \mathbf{u}^r, \left\langle \mathbf{u}^{r(-1)^{x_i} u_i} \right\rangle_{i=1}^{k-1}, \mathbf{w}^{(-1)^{x_k}}, \langle \mathbf{f}_i \rangle_{i=k+1}^n, \prod_{i=1}^{k-1} \mathbf{u}^{r u_i} \cdot \mathbf{w} \cdot \prod_{i=k+1}^n \mathbf{f}_i^{(-1)^{x_i}}, \\ \mathbf{u}^s, \left\langle \mathbf{u}^{s(-1)^{x_i} u_i} \right\rangle_{i=1}^{k-1}, \langle \mathbf{h}_i \rangle_{i=k}^n, \prod_{i=1}^{k-1} \mathbf{u}^{s u_i} \cdot \prod_{i=k}^n \mathbf{h}_i^{(-1)^{x_i}} \end{pmatrix}$$

Now, note that $Dist_{0,k,DDH}$ is identical to $Dist_{0,k}$ when $\mathbf{w}$ is random and is otherwise identical to:

$$Dist_{0,k+1/2} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{g}_0, \langle \mathbf{g}_i \rangle_{i=1}^n, \\ \mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^k, \langle \mathbf{f}_i \rangle_{i=k+1}^n, \prod_{i=1}^k \mathbf{g}_i^r \cdot \prod_{i=k+1}^n \mathbf{f}_i^{(-1)^{x_i}}, \\ \mathbf{g}_0^s, \left\langle \mathbf{g}_i^{(-1)^{x_i} s} \right\rangle_{i=1}^{k-1}, \langle \mathbf{h}_i \rangle_{i=k}^n, \prod_{i=1}^{k-1} \mathbf{g}_i^s \cdot \prod_{i=k}^n \mathbf{h}_i^{(-1)^{x_i}} \end{pmatrix}$$

By a similar reduction, $Dist_{0,k+1/2} \approx_{DDH} Dist_{0,k+1}$, leading to the conclusion $Dist_{0,k} \approx_{DDH} Dist_{0,k+1}$. Completing the chain, we have, $Dist_0 = Dist_{0,n+1} \approx_{DDH} \cdots \approx_{DDH} Dist_{0,1} = Dist'_0$.

By doing an analogous proof, we have that the following distributions are indistinguishable as well:

$$Dist_1 \stackrel{\text{def}}{=} \left( \mathbf{g}_0, \langle \mathbf{g}_i \rangle_{i=1}^n, \quad \mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^n, \prod_{i=1}^n \mathbf{g}_i^r, \quad \mathbf{g}_0^s, \left\langle \mathbf{g}_i^{(-1)^{y_i} s} \right\rangle_{i=1}^n, \prod_{i=1}^n \mathbf{g}_i^s \right)$$

and

$$Dist'_1 \stackrel{\text{def}}{=} \left( \mathbf{g}_0, \langle \mathbf{g}_i \rangle_{i=1}^n, \quad \mathbf{f}_0, \langle \mathbf{f}_i \rangle_{i=1}^n, \prod_{i=1}^n \mathbf{f}_i^{(-1)^{x_i}}, \quad \mathbf{h}_0, \langle \mathbf{h}_i \rangle_{i=1}^n, \prod_{i=1}^n \mathbf{h}_i^{(-1)^{y_i}} \right),$$

where the $\mathbf{g}_i, \mathbf{f}_i$ and $\mathbf{h}_i$'s are sampled independently randomly.

Finally observe that $Dist'_0$ and $Dist'_1$ are indistinguishable by the Decisional Binary Mix assumption (Assumption 2). Hence we have: $Dist_0 \approx_{DDH,\text{Decisional Binary Mix}} Dist_1$. □

Now we proceed to prove Theorem 9. Specifically, we show that an adversary for distinguishing the distributions (Let's call them $\Delta_0$ and $\Delta_1$) in Theorem 9 can be used to build an adversary for distinguishing $Dist_0$ and $Dist_1$. So suppose we are given a sample:

$$\left( \mathbf{g}_0, \langle \mathbf{g}_i \rangle_{i=1}^n, \quad \mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^n, \prod_{i=1}^n \mathbf{g}_i^r, \quad \mathbf{g}_0^s, \left\langle \mathbf{g}_i^{(-1)^{z_i} s} \right\rangle_{i=1}^n, \prod_{i=1}^n \mathbf{g}_i^s \right),$$

where the task is to decide if $z = x$ or independently random.

We now construct a $\Delta_0/\Delta_1$ distinguishing adversary $B$ as follows: We sample $u$, $s$ and $\langle u_i \rangle_{i=1}^n$, all randomly from $\mathbb{Z}_q^*$. Sample $\mathbf{h}_0$ randomly from $\mathbb{G}_2$. Now we define $pk$ as $(pk_1, pk_2, pk_R)$:

$$\begin{aligned} pk_1 &:= \mathbf{g}_0, \left\langle \mathbf{g}_i^{u_i^{-1}} \right\rangle_{i=1}^n, \mathbf{g}_0^u \prod_{i=1}^n \mathbf{g}_i^{-1} \\ pk_2 &:= \mathbf{h}_0, \langle \mathbf{h}_0^{u_i s} \rangle_{i=1}^n, \mathbf{h}_0^s \\ pk_R &:= us \end{aligned}$$

Observe that $\mathbf{g}_0, \left\langle \mathbf{g}_i^{u_i^{-1}} \right\rangle_{i=1}^n, \mathbf{h}_0, \langle \mathbf{h}_0^{u_i s} \rangle_{i=1}^n, \mathbf{h}_0^s$ and $us$ are all uniformly random and independent elements of their respective domains. The group element $\mathbf{g}_0^u \prod_{i=1}^n \mathbf{g}_i^{-1}$ is fixed given the other elements. Hence $(pk_1, pk_2, pk_R)$ has identical distribution as the original protocol.

$A$ then publishes the following $Tuple$ to the adversary $B$:

$$Tuple \stackrel{\text{def}}{=} \begin{pmatrix} pk, \\ \mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i} r \cdot u_i^{-1}} \right\rangle_{i=1}^n, \mathbf{g}_0^{r \cdot u} \left( \prod_{i=1}^n \mathbf{g}_i^r \right)^{-1}, \\ \mathbf{g}_0^s, \left\langle \mathbf{g}_i^{(-1)^{z_i} s \cdot u_i^{-1}} \right\rangle_{i=1}^n, \mathbf{g}_0^{s \cdot u} \left( \prod_{i=1}^n \mathbf{g}_i^s \right)^{-1} \end{pmatrix}.$$

$A$ then relays the response of $B$. In the case that $z = x$, $Tuple$ is from the distribution $\Delta_0$. In the case that $z$ is random, $Tuple$ is from the distribution $\Delta_1$. □

## I Relational Hash for Equality

We now construct a Relational Hash scheme for the domains $X, Y = \mathbb{Z}_q$ and the relation $R = \{(x, y, z) \mid x = y \wedge x, y \in \mathbb{Z}_q\}$. There is no public input $z$.

KEYGEN*:* Given the security parameter, bilinear groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are generated of prime order $q$, exponential in the security parameter, and with a bilinear pairing operator $e$. Now we sample generators $\mathbf{g} \leftarrow \mathbb{G}_1$ and $\mathbf{h} \leftarrow \mathbb{G}_2$. Now we define the output of KEYGEN as $pk := (pk_1, pk_2)$, defined as follows:

$$pk_1 := \mathbf{g}, \qquad pk_2 := \mathbf{h},$$

HASH$_1$*:* Given plaintext $x \in \mathbb{Z}_q$ and $pk_1 = \mathbf{g}$, the hash is constructed as follows: Sample a random $r \in \mathbb{Z}_q$ and then compute the following:

$$hx := (\mathbf{g}^r, \mathbf{g}^{rx})$$

HASH$_2$*:* Given plaintext $y \in \mathbb{Z}_q^*$ and $pk_2 = \mathbf{h}$, the hash is constructed as follows: Sample a random $s \in \mathbb{Z}_q^*$ and then compute the following:

$$hx := (\mathbf{h}^r, \mathbf{h}^{ry})$$

VERIFY*:* Given hashes $hx = (hx_1, hx_2)$ and $hy = (hy_1, hy_2)$, the algorithm VERIFY checks the following equality:

$$e(hx_1, hy_2) \stackrel{?}{=} e(hx_2, hy_1)$$

**Theorem 15.** *The above equality relational hash is one-way secure under the DLP assumption (Assumption 3) when the plaintext comes from the uniform distribution over $\mathbb{Z}_q$.*

*Proof.* For random $r \leftarrow \mathbb{Z}_q, x \leftarrow \mathbb{Z}_q$, let EQHASH$_1(x) = (\mathbf{g}^r, \mathbf{g}^{rx})$ be the challenge to an one-way attacker $\mathcal{A}$, which outputs $x$ with non-negligible probability. Now, given a DLP challenge $(\mathbf{g}, \mathbf{g}^y)$ we can simply choose a random $r \leftarrow \mathbb{Z}_q$ and give $(\mathbf{g}^r, \mathbf{g}^{ry})$ to $\mathcal{A}$. We send $\mathcal{A}$'s output as it is to the DLP challenger. Whenever $\mathcal{A}$ is successful, we can also successfully break the DLP challenge. □

**Theorem 16.** *The above equality relational hash is oracle simulation secure as well as unforgeable with respect to the independent uniform distributions over $\mathbb{Z}_q$, under the SXDH assumption (Assumption 5).*

*Proof.* Following [Can97,CMR98], we can show EQHASH$_1$ and EQHASH$_2$ are individually 2-value Probabilistic One-Way for uniform distributions over $\mathbb{Z}_q$. For random $x, y, r, s \leftarrow \mathbb{Z}_q$ we have

$$\begin{aligned}(\text{EQHASH}_1(x), \text{EQHASH}_1(x)) &= ((\mathbf{g}^r, \mathbf{g}^{rx}), (\mathbf{g}^s, \mathbf{g}^{sx})) \\ &\approx_{DDH} ((\mathbf{g}^r, \mathbf{g}^{rx}), (\mathbf{g}^s, \mathbf{g}^{sy})) = (\text{EQHASH}_1(x), \text{EQHASH}_1(y))\end{aligned}$$

The same argument holds for EQHASH$_2$ as well. Moreover, for independent uniform distributions equality is a sparse relation. Hence, by Theorem 7 and Theorem 8 from Section 5, we have the equality relational hash is oracle simulation secure as well as unforgeable. □