Relational Hash

Avradip Mandal and Arnab Roy

Fujitsu Laboratories of America
Sunnyvale, CA 94085

Abstract. Traditional cryptographic hash functions allow one to easily check whether the original plain-texts 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 plain-texts were related or not. We formalize various natural security notions for the relational hash primitive - 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 these linear relational hash schemes we develop relational hashes for detecting proximity in terms of hamming distance. These proximity relational hashing scheme can be adapted to a privacy preserving biometric authentication scheme.

We also introduce the notion of *relational encryption*, which is a regular semantically secure public key encryption for any adversary which only has access to the public key. However, a semi-trusted entity can be given a relational key using which it can discover relations among ciphertexts, but still cannot decrypt and recover the plaintexts.

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

1 Introduction

We propose a cryptographic primitive called *relational hash*, which is an extension of traditional cryptographic hash functions. Any collision resistance hash function h, is in fact a relational hash for the equality relation. Given $h(x_1)$ and $h(x_2)$, one can easily verify whether $x_1 \stackrel{?}{=} x_2$, just by checking $h(x_1) \stackrel{?}{=} h(x_2)$. However, a probabilistic hash function [Can97] ph does not admit efficient checking of $x_1 \stackrel{?}{=} x_2$, given $ph(x_1, r_1)$ and $ph(x_2, r_2)$, where the r_i's are randomnesses used for hashing.

So the question arises whether we can build probabilistic hash functions which admit verification of equality given just the hashes. Extending further, given the hashes and a relation R, can we efficiently determine if the underlying plaintexts are in the relation R? We give positive answers in this direction and construct a linear relational hash scheme. In our scheme, for any $x, y \in \mathbb{F}_2^n$, given just the hashes of x and y it is possible to verify whether $x + y \stackrel{?}{=} z$ (for any $z \in \mathbb{F}_2^n$). 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 $\mathbb{F}_p^n$.

For any relational hash primitive, we formalize a few natural and desirable security properties, namely one-wayness, unforgeability and oracle simulatibility. The notion of oracle simulatibility was introduced in [Can97,CMR98] for the equality relation. Here we extend the definition of the concept for arbitrary relations. We show that our linear relational hash constructions satisfy all three security notions.

While the linear relational hash is itself interesting from a theoretical point of view, it has little practical significance. However, 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. The most motivating application of the proximity relation hash primitive is a privacy preserving biometric authentication scheme. Using a relational hash scheme for proximity relation, one can build a biometric authentication scheme which guarantees template privacy. During both the registration and authentication phase of the protocol the client always sends a hashed version of the 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. Existing biometric authentication schemes, e.g. fuzzy vault [JS02], fuzzy commitment [JW99] and secure sketch [DRS04,DS05] based schemes only guarantee privacy during the registration phase. Compared to that, the relational hash based scheme would guarantee template privacy in both the registration and authentication phase.

We point out that the existence of a general purpose obfuscator $\mathcal{O}$ does not necessarily imply that we can build relational hash functions for any relation. Suppose we publish $\mathcal{O}(R_x(\cdot,\cdot))$ and $\mathcal{O}(R_y(\cdot,\cdot))$ for domain elements x and y, where $R_x(\cdot,\cdot)$ is a function which outputs $R(x, y', z')$, taking y', z' input ($R_y(\cdot,\cdot)$ is defined similarly). The verifier will have the power to evaluate $R(x, y', z')$ and $R(x', y, z')$ for any (x', y', z') of his choice, but still the verifier will be unable to find out $R(x, y, z')$. Hence the obfuscations do not qualify as relational hashes.

Using relational hash in a black box manner we can also build another cryptographic primitive called *relational encryption.* Relational encryption has the following remarkable property: with just the knowledge of a public key it is a semantically secure public key encryption scheme. However, a semi-trusted entity can be given a *relational key* using which it can discover a given relation among ciphertexts, but still not be able to decrypt and recover the plaintexts.

Conceptually, the relational encryption primitive is similar to multi-input functional encryption [GKL$^+$13,GGJS13]. However, unlike multi-input functional encryption, which is defined in a generalized setting and the goal is to evaluate a function $f(x_1, \cdots, x_n)$ with n inputs $x_1, \cdots, x_n$, we only consider relations of a specific nature and security guarantees of varying strength, ranging from one-wayness to oracle simulatability. This allows us to build our primitive in a public key setting, which is not possible in the framework of multi-input functional encryption for these particular relations.

Organization of the paper. In Section 2, we formally define the notion of relational hash and its desired security properties. In the following section (Section 3), we construct a relational hash for linearity over $\mathbb{F}_2^n$. This construction is secure under the standard SXDH assumption and a new hardness assumption Binary Mix DLP (Assumption 1). In Appendix C, we show that the Binary Mix DLP assumption can actually be reduced to the more standard Random Modular Subset Sum assumption (Assumption 7). As an added assurance in Appendix D, we show that the Binary Mix DLP assumption is also secure in the Generic Group Model [Sho97].

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. Afterwards, in Section 5, we extend the linear relational hash construction over $\mathbb{F}_2^n$ (from Section 3) to $\mathbb{F}_p^n$.

In Section 6, we formally define the notion of sparse and biased relations. We argue that only biased relations allow a strong notion of security of relational hash functions. We also discuss the relations between various security notions of relational hash functions in this section. In the following section (Section 6), we show that the linear relational hash construction from Section 3 actually satisfies a stronger security property (Oracle Simulatability), albeit from a stronger hardness assumption called Decisional Binary Mix. In Appendix D, we also argue the Decisional Binary Mix assumption is in fact secure in the Generic Group Model.

Finally, in Section 7.1, we introduce the notion of relational encryption, and show how we can build relational encryption schemes based on relation hash schemes from previous sections.

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 λ.

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.

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 algorithms:

- A randomized key generation algorithm: $\textsc{KeyGen}(1^\lambda)$ outputs key pk from key space K_λ.
- 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 λ.
- 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 λ.
- The deterministic verification algorithm: $\textsc{Verify} : K_\lambda \times \textsc{RangeX}_\lambda \times \textsc{RangeY}_\lambda \times Z_\lambda \to \{0, 1\}$.

In the rest of the paper we will drop the subscript λ 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 1 (Relational Hash - Correctness). *A* relational hash *scheme for a relation* $R \subseteq X \times Y \times Z$ *is a tuple of algorithms* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *satisfying the following correctness condition. Compute:*

$$pk \leftarrow \textsc{KeyGen}(1^\lambda), \qquad hx \leftarrow \textsc{Hash}_1(pk, x), \qquad hy \leftarrow \textsc{Hash}_2(pk, y),$$
$$b \leftarrow \textsc{Verify}(pk, hx, hy, z)$$

Then for any $(x, y, z) \in X \times Y \times Z$ *we require that* $b \cong R(x, y, z)$ *with overwhelming probability (calculated over the internal randomness of* $\textsc{KeyGen}$, $\textsc{Hash}_1$ *and* $\textsc{Hash}_2$*).*

Note that a trivially correct relational hash scheme is a set of identity functions as hash algorithms and the relation itself as verify algorithm. Of course, as we will see such instantiations of relational hash schemes are not secure.

Security: However, the notion of security for a relational hash is not straight forward. It will depend in the context where the relational hash is going to be used and also on the a priori information available to the adversary. Recall, for a regular hash function one of the weakest form of security is one-wayness. A function f is called one-way if given $f(x)$, it is hard to output x.

Definition 2 (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 Probabilistic Polynomial Time (PPT) adversary A_1, there exists a negligible function* $\texttt{negl}()$, *such that* $\Pr[A_1(pk, hx) = x] < \texttt{negl}(\lambda)$.
- *For any Probabilistic Polynomial Time (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 $\mathcal{Y}$.*

In this work, we are mostly interested in *sparse* relations (Definition 6). 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, such that $\textsc{Verify}(pk, hx, hy)$ outputs 1. Formally,

Definition 3 (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 adversary A_1, there exists a negligible function* $\texttt{negl}()$, *such that*

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

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

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

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 authors were 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 $(x, y, z') \in R$.

Definition 4 (Security of Relational Hash - Oracle Simulation). *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 said to be oracle simulation secure with respect to the distributions $\mathcal{X}$ and $\mathcal{Y}$ 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* $\textsc{KeyGen}$*), there exists a negligible function* $\texttt{negl}()$, *such that*

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

where $x \leftarrow \mathcal{X}, y \leftarrow \mathcal{Y}$ and $pk \leftarrow \textsc{Keygen}(1^\lambda)$.

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. 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 $U \subseteq \{1, \cdots, n\}$ be the set of indices, such that $i \in U$ 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 $\mathbb{F}_2^n$. □

Security. 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*

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

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

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

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 assumption and Binary Mix DLP assumptions, when x and y are sampled uniformly from $\mathbb{F}_2^n$.*

We now prove the theorem, assuming a lemma (Lemma 4) which holds under the SXDH and Binary Mix DLP assumptions. While the lemma is formally stated and proved in Appendix B, we provide an intuitive description here. The lemma states that given $\left(\mathbf{g}, \langle \hat{\mathbf{g}}_i \rangle_{i=1}^n, \mathbf{g}^r, \prod_{i=1}^n \hat{\mathbf{g}}_i^r, \left\langle \hat{\mathbf{g}}_i^{(-1)^{x_i} r} \right\rangle_{i=1}^n\right)$, it is hard to compute $x \in \mathbb{F}_2^n$. 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 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 $\textsc{KeyGen}$ 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 [1]. 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:

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

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. □

Unforgeability and Oracle Simulation Security. In Section 6, we show this relational hash is in fact a 2-value perfectly one-way function, albeit under a stronger hardness assumption. By Theorem 7 from Section 6, that will imply this relational hash construction is also unforgeable and oracle simulation secure.

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 $R_\delta = \{(x, y) \mid \texttt{dist}(x, y) \le \delta \wedge x, y \in \mathbb{F}_2^n\}$, where `dist` is the hamming distance and δ is a positive integer less than n. [2] 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 number of one's in the binary representation of x. For any error vector $e \in \mathbb{F}_2^n$, with $\texttt{weight}(e) \le d/2$ and $m \in \mathbb{F}_2^k$ we have,

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

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,

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

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

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

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

[1] 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.

[2] Note, the notion of relational hash is defined over 3-tuple relations (Definition 8). 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 $(x, y) \in R_\delta$ for all $z \in Z$.

$\textsc{Hash}_2$ is defined similarly.

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

Theorem 2. *The above algorithms* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *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* $\mathbb{F}_2^k$*.*

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

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

If $\texttt{dist}(x, y) \leq \delta$, $\textsc{Decode}(hx_1 + hx_2)$ 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 $z = \textsc{Decode}(hx_1 + hx_2)$ can never be same as $(r + s)$, because $\texttt{dist}(\textsc{Encode}(r+s), hx_1 + hx_2) = \texttt{dist}(x, y) > \delta$. 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 2) 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 3) 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

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

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. □

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

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

$\textsc{KeyGen}$*:* 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 order p. Let ω 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 $\textsc{KeyGen}$ as $pk := (\omega, pk_1, pk_2, pk_R)$ defined as follows:

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

$\textsc{Hash}_1$*:* 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:

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

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

$\textsc{Verify}$*:* 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 ω, the algorithm $\textsc{Verify}$ checks the following equality:

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

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 details see Appendix E.

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 λ).

Assumption 2 (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 ω 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,*

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

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

$$\Pr\left[\begin{array}{c}(n, q, \omega, \mathbb{G}) \leftarrow \mathcal{G}(1^\lambda); x \leftarrow \mathbb{F}_p^n, \langle \mathbf{g}_i \rangle_{i=1}^n \leftarrow \mathbb{G}; y \leftarrow A\left(\langle \mathbf{g}_i \rangle_{i=1}^n, \prod_{i=1}^n \mathbf{g}_i^{\omega^{x_i}}\right) : \\ \prod_{i=1}^n \mathbf{g}_i^{\omega^{x_i}} = \prod_{i=1}^n \mathbf{g}_i^{\omega^{y_i}}\end{array}\right] < \texttt{negl}(\lambda).$$

Theorem 3. *The above algorithms* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *constitute a relational hash scheme for the relation $R = \{(x, y, z) \mid x + y = z \wedge x, y, z \in \mathbb{F}_p^n\}$. The scheme is one-way secure under the SXDH assumption and p-ary Mix DLP assumption, when x and y's are sample uniformly from $\mathbb{F}_p^n$.*

A proof sketch is given in Appendix F.

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 7 from Section 6, 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 $\texttt{dist}$ is the p-ary hamming distance and δ 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.

6 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 define 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 the 2-POW property is actually a sufficient condition for simulation security, as well as unforgeability (when the relation is sparse).

Definition 5. *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:*

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

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 $\mathcal{Y}$.

Definition 6. *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:*

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

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 independent 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 sprase relation in second coordinate with respect to $\mathcal{Y}$.

Remark. We note that a biased relation may not be such that it is sparse or its complement is sparse. A counterexample is a 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.

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

Theorem 4. *If a relational hash scheme* $(\textsc{KeyGen}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{Verify})$ *for a relation R is unforgeable for probability distributions independent 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 unforgeablity challenge (pk, cx_1), such that $pk \leftarrow \textsc{KeyGen}(1^\lambda)$, $cx_1 := \textsc{Hash}_1(pk, x)$ for some $x \leftarrow \mathcal{X}$; we can just get $(y, z) \leftarrow A(\lambda)$ 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) function. This is a useful property, because if we can show a relational hash function is 2-POW, that would immediately imply the relational hash function is simulation secure, as well as unforgeable (if the relation is sparse).

Definition 7 (2-value Perfectly One-Way function). *Let, $\mathcal{X}$ be a probability distribution over X. $H = \{h_k\}_{k \in K}$ 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,*

$$|\Pr[D(k, h_k(x, r_1), h_k(x, r_2)) = 1] - \Pr[D(k, h_k(x_1, r_1), h_k(x_2, r_2)) = 1]| < \texttt{negl}(\lambda),$$

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 unifromly at random from randomness space U.

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

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

Theorem 5. *Given a relational hash scheme* (KEYGEN, HASH$_1$, HASH$_2$, VERIFY) *for a relation R, if* HASH$_1$ *is 2-value Probabilistic One-Way with respect to $\mathcal{X}$ and* KEYGEN*, then R is a biased relation in the 1st co-ordinate with respect to $\mathcal{X}$.*

Proof. We are given that,

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

Suppose R is 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 $\Pr[x \leftarrow X, (y, z) \leftarrow A(\lambda) : R(x, y, z) \neq R(w, y, z)]$ 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 $(y, z) \leftarrow A(\lambda)$, 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 6, stated below shows if a relational hash is 2-POW, then it is also oracle simulation secure.

Theorem 6. *If a relational hash scheme* (KEYGEN, HASH$_1$, HASH$_2$, VERIFY) *is such that both* HASH$_1$ *and* HASH$_2$ *are individually 2-value Probabilistic One-Way for distributions* ($\mathcal{X}$, KEYGEN) *and* ($\mathcal{Y}$, 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:*

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

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

We postpone the proof to Appendix G. Finally, we show if a relational hash is 2-POW as well as sparse, then it must be unforgeable.

Theorem 7. *If a relational hash scheme* (KEYGEN, HASH$_1$, HASH$_2$, VERIFY) *for a sparse relation R with respect to independent probability distributions $\mathcal{X}$ and $\mathcal{Y}$, is such that* HASH$_1$ *(*HASH$_2$*) is 2-value Probabilistic One-Way for distribution $\mathcal{X}$ ($\mathcal{Y}$) and* 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 $\textsc{Hash}_2(pk, y, s), z$, 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 $\textsc{Hash}_2(pk, y, s), z$, 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 HASH$_1$. □

Stronger Security Properties for the Relational Hash Constructions. In Theorem 8, 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 3). In Appendix D (Theorem 11) we show that this assumption holds in 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 6 and Theorem 7 we get that the relational hash construction from Section 3 is actually oracle simulation secure as well as unforgeable with respect to the uniform distributions over $\mathbb{F}_2^n$.

Assumption 3 (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:*

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

More formally, for all PPT adversaries A, there exists a negligible function `negl()` *such that*

$$\left| \begin{array}{c} \Pr\left[\begin{array}{c} (n, q, \mathbb{G}) \leftarrow \mathcal{G}(1^\lambda); x \leftarrow \mathbb{F}_2^n, \langle \mathbf{g}_i \rangle_{i=1}^n \leftarrow \mathbb{G}, \langle \mathbf{f}_i \rangle_{i=1}^n \leftarrow \mathbb{G}; \\ A\left(\langle \mathbf{g}_i \rangle_{i=1}^n, \prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}}, \langle \mathbf{f}_i \rangle_{i=1}^n, \prod_{i=1}^n \mathbf{f}_i^{(-1)^{x_i}} \right) \to 1 \end{array} \right] \\ - \Pr\left[\begin{array}{c} (n, q, \mathbb{G}) \leftarrow \mathcal{G}(1^\lambda); x, y \leftarrow \mathbb{F}_2^n, \langle \mathbf{g}_i \rangle_{i=1}^n \leftarrow \mathbb{G}, \langle \mathbf{f}_i \rangle_{i=1}^n \leftarrow \mathbb{G}; \\ A\left(\langle \mathbf{g}_i \rangle_{i=1}^n, \prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}}, \langle \mathbf{f}_i \rangle_{i=1}^n, \prod_{i=1}^n \mathbf{f}_i^{(-1)^{y_i}} \right) \to 1 \end{array} \right] \end{array} \right| < \mathtt{negl}(\lambda)$$

Theorem 8. *The algorithms* (KeyGen, Hash$_1$, Verify) *in Section 3 constitute a 2-value Probabilistic One Way Function for the uniform distribution on $\mathbb{F}_2^n$. In more details, sample random elements $\mathbf{g}_0 \leftarrow \mathbb{G}_1, \mathbf{h}_0 \leftarrow \mathbb{G}_2, \langle a_i \rangle_{i=1}^{n+1}, \langle b_i \rangle_{i=1}^{n+1} \leftarrow \mathbb{Z}_q^*$. Let $\mathbf{g}_i = \mathbf{g}_0^{a_i}$ and $\mathbf{h}_i = \mathbf{h}_0^{b_i}$ and $pk_R = \sum_{i=1}^{n+1} a_i b_i$. Let $pk = (\langle \mathbf{g}_i \rangle_{i=0}^{n+1}, \langle \mathbf{h}_i \rangle_{i=0}^{n+1}, pk_R)$. Under the Decisional Binary Mix 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 postpone the proof to Appendix H.

7 Applications

We enlist two interesting applications of relational hash functions: relational encryption and privacy preserving biometric authentication.

7.1 Relational Encryption

The concept of relational hash functions extends naturally to relational encryption. With just the knowledge of a public key it is a semantically secure public key encryption scheme. However, a semi-trusted entity can be given a relational key using which it can discover a given relation among ciphertexts, but still not be able to decrypt and recover the plaintext. In Appendix I we formally define the notion of relational encryption and its desired security properties. We also give black box relational encryption constructions from a relational hash primitive and a CPA secure encryption scheme. Moreover, we also give more efficient non-black box constructions starting from relational hash constructions for linearity and proximity from previous sections.

Conceptually, the relational encryption primitive is similar to multi-input functional encryption [GKL$^+$13,GGJS13]. However, unlike multi-input functional encryption, which is defined in a generalized setting and the goal is to evaluate a function $f(x_1, \cdots, x_n)$ with n inputs $x_1, \cdots, x_n$, we only consider relations of a specific nature and security guarantees of varying strength, ranging from one-wayness to oracle simulatability. This allows us to build our primitive in a public key setting, which is not possible in the framework of multi-input functional encryption for these particular relations.

7.2 Privacy Preserving Biometric Authentication

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. Existing biometric authentication schemes, e.g. fuzzy vault [JS02], fuzzy commitment [JW99] and secure sketch [DRS04,DS05] based schemes only guarantee privacy during the registration phase. 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 (TTP) runs $\textsc{KeyGen}$ and publishes $pk \leftarrow \textsc{KeyGen}$. 2. During Registration, client generates biometric template $x \in \{0,1\}^n$ and sends $hx = \textsc{Hash}_1(pk, x)$ to server. 3. During Authentication, client generates biometric template $y \in \{0,1\}^n$ and sends $hy = \textsc{Hash}_2(pk, y)$ to server. 4. Server authenticates the user 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 2 from Section 4 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. Also, if we use a relational encryption (where a trusted third party will run $\textsc{KeyGen}$, the server will have the relational key, the private key will be thrown away) instead of a relational hash, then we have the added security that only the server can check relations and authenticate the users. Even if the server's database gets leaked to an attacker, without access to the relational key it will look like a semantically secure encrypted database.

In spite of these strong guarantees there are a few drawbacks of our privacy preserving authentication scheme, which remain open problems for further exploration. 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.

In addition, Theorem 2 is no longer applicable if the server receives $hx = \textsc{Hash}_1(pk, x)$ and $hy = \textsc{Hash}_2(pk, y)$, where x and y do not come from independent uniform distributions. In practice, this is an important case: for example, if x and y are generated by the same person they clearly come from dependent distributions with $\texttt{dist}(x, y) \le \delta$. We still conjecture that even in this case the proximity relational hash construction from Section 4 remains one-way (albeit the security will probably depend on a stronger hardness assumption).

References

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Berlin, Germany.

Can97. Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Berlin, Germany.

CMR98. Ran Canetti, Daniele Micciancio, and Omer Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 131–140, Dallas, Texas, USA, May 23–26, 1998. ACM Press.

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, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer, Berlin, Germany.

DS05. Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 654–663, Baltimore, Maryland, USA, May 22–24, 2005. ACM Press.

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: 6th Conference on Computer and Communications Security*, pages 28–36, Kent Ridge Digital Labs, Singapore, November 1–4, 1999. ACM Press.

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

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Berlin, Germany.

A Hardness Assumptions

We summarize the standard hardness assumptions used in this paper.

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*

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

Assumption 5 (XDH [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 eXternal decisional Diffie-Hellman (XDH) assumption asserts that the Decisional Diffie-Hellman (DDH) problem is hard in* one *of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$.*

Assumption 6 (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 7 (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 $\sum_{i=1}^n \epsilon_i a_i$ for a random $\epsilon \leftarrow \{0,1\}^n$, it is hard to output $\eta \in \{0,1\}^n$ such that*

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

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

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

B Lemma for Proving Security of $\mathbb{F}_2^n$ Linear Relational Hash

In this section we go through a sequence of lemmas, leading to Lemma 4, which is directly used in the 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, 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 $\mathtt{negl}()$ *such that*

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

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. $(\langle \mathbf{g}_i \rangle_{i=1}^n, \mathbf{w} = \prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}})$ 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.

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

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 χ 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 χ 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 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 $\texttt{negl}()$ *such that*

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

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. □

C 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*

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

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

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

Theorem 9. *The Binary-Mix-DLP assumption (Assumption 1) can be reduced to the Random-Modular-Subset-Sum assumption (Assumption 7).*

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, τ is also a solution to the Random-Modular-Subset-Sum challenge. □

D Generic Group Model

In this section we show Assumption 1 and Assumption 3 hold in 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 σ. 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 σ is negligible (as $m >> \log q$).

Theorem 10. *The Binary Mix DLP assumption (Assumption 1) holds in 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 $\{0,1\}^m$:

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

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, σ) where F is a polynomial in the ring specified above and σ is the encoding of a group element. The polynomial F represents the exponent of the encoded element. Initially, L is set to,

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

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

– **Group Action** : Given two strings σ_i, σ_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 σ, $\mathcal{B}$ 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.

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

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 $\left\langle a_k^i \right\rangle_{k=1}^n, c^i$ 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 5 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 5. *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:*

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

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

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

$\square$

Theorem 11. *The Decisional Binary Mix assumption (Assumption 3) holds in 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 $\{0, 1\}^m$:

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

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, σ) where F is a polynomial in the ring specified above and σ 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 σ_i, σ_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 σ, $\mathcal{B}$ 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 $\langle a_k^i\rangle_{k=1}^n, \langle b_k^i\rangle_{k=1}^n, c^i, d^i, e^i$ 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 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 $|L_0| + t = 2n + t + 3$. 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 6. *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,

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

With a similar analysis, we can also show

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

□

E Correctness of the $\mathbb{F}_p^n$ Linear Relational Hash

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

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

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 $\mathbb{F}_p^n$. □

F Proof Sketch of One-wayness of the $\mathbb{F}_p^n$ Linear Relational Hash

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. Detailed proof will be provided in the full version.

Lemma 7. *Under the p-ary Mix DLP 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 $\texttt{negl}()$ *such that*

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

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 7. To achieve that, consider that the adversary A is given a Lemma 7 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 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 7. □

G Proof of Theorem 6

We, recall in Definition 4 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:*

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

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 $Q_{x,k} \overset{\text{def}}{=} \Pr[A(k, h_k(x, r)) = 1]$. Now we have,

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

□

Now we prove Theorem 6. 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,

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

□

H Proof of Theorem 8

We prove this theorem using the following lemma.

Lemma 9. *Under the Decisional Binary Mix 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

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

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 $\mathbb{F}_2^n$:

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

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

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

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 (DBM) assumption. Hence we have: $Dist_0 \approx_{DDH,DBM} Dist_1$. □

Now we proceed to prove Theorem 8. Specifically, we show that an adversary for distinguishing the distributions (Let's call them Δ_0 and Δ_1) in Theorem 8 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 Δ_0/Δ_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 Δ_0. In the case that z is random, $Tuple$ is from the distribution Δ_1. □

I Relational Encryption

Definition 8 (Relational Encryption). *A* Relational Encryption *scheme for a relation $R \subseteq X \times Y \times Z$ is a tuple of algorithms* (KeyGen, Enc$_1$, Dec$_1$, Enc$_2$, Dec$_2$, Verify) *satisfying the* *following correctness conditions:*

$$\begin{aligned}&(pk_1, sk_1, pk_2, sk_2, sk_R) \leftarrow \textsc{KeyGen}(1^\lambda)\\&cx \leftarrow \textsc{Enc}_1(pk_1, x)\\&cy \leftarrow \textsc{Enc}_2(pk_2, y)\\&b \leftarrow \textsc{Verify}(sk_R, cx, cy, z)\end{aligned}$$

then we require that $b \cong R(x, y, z)$ *with overwhelming probability.*

Definition 9 (Security Definition). *We define a Relational Encryption scheme* ($\textsc{KeyGen}$, $\textsc{Enc}_1$, $\textsc{Enc}_2$, $\textsc{Dec}_1$, $\textsc{Dec}_2$, $\textsc{Verify}$) *to be IND-CPA secure if the following hold:*

- *Let* $\mathrm{K}_1(1^\lambda)$ *be the algorithm that runs* $\textsc{KeyGen}(1^\lambda)$*, takes its output* $(pk_1, sk_1, pk_2, sk_2, sk_R)$ *and outputs* (pk_1, sk_1)*. Then* $(\mathrm{K}_1, \textsc{Enc}_1, \textsc{Dec}_1)$ *is IND-CPA secure.*
- *Let* $\mathrm{K}_2(1^\lambda)$ *be the algorithm that runs* $\textsc{KeyGen}(1^\lambda)$*, takes its output* $(pk_1, sk_1, pk_2, sk_2, sk_R)$ *and outputs* (pk_2, sk_2)*. Then* $(\mathrm{K}_2, \textsc{Enc}_2, \textsc{Dec}_2)$ *is IND-CPA secure.*

Moreover, the relational encryption scheme is one-way secure or unforgeable or oracle simulation secure if condition 1, 2 or 3 holds. Let $\mathrm{K}_R(1^\lambda)$ *be the algorithm that runs* $\textsc{KeyGen}(1^\lambda)$*, takes its output* $(pk_1, sk_1, pk_2, sk_2, sk_R)$ *and outputs* $pk = (pk_1, pk_2, sk_R)$*. Suppose,* $\mathcal{X}$ *and* $\mathcal{Y}$ *be independent probability distributions over* X *and* Y*.*

1. *The relation encryption scheme is one-way secure for the probability distributions* $\mathcal{X}$ *and* $\mathcal{Y}$ *if*
$$(\mathrm{K}_R, \textsc{Enc}_1, \textsc{Enc}_2, \textsc{Verify})$$
is a one-way secure relational hash for the probability distributions $\mathcal{X}$ *and* $\mathcal{Y}$*.*
2. *The relation encryption scheme is unforgeable for the probability distributions* $\mathcal{X}$ *and* $\mathcal{Y}$ *if*
$$(\mathrm{K}_R, \textsc{Enc}_1, \textsc{Enc}_2, \textsc{Verify})$$
is an unforgeable relational hash for the probability distributions $\mathcal{X}$ *and* $\mathcal{Y}$*.*
3. *The relation encryption scheme is oracle simulation secure for the probability distributions* $\mathcal{X}$ *and* $\mathcal{Y}$ *if*
$$(\mathrm{K}_R, \textsc{Enc}_1, \textsc{Enc}_2, \textsc{Verify})$$
is an oracle simulation secure relational hash for the probability distributions $\mathcal{X}$ *and* $\mathcal{Y}$*.*

I.1 Black Box Construction of Relational Encryption Scheme from a Relational Hash

Given a relational hash scheme,

$$(\textsc{KeyGenHash}, \textsc{Hash}_1, \textsc{Hash}_2, \textsc{VerifyHash})$$

for a relation $R \subseteq X \times Y \times Z$ and an IND-CPA secure encryption scheme

$$(\textsc{KeyGenCPA}, \textsc{EncCPA}, \textsc{DecCPA})$$

we can build a relation encryption scheme for the same relation R as follows.

KeyGen: Run KeyGenCPA three times and get three pairs of public/private keys.

$$(pk^1_{cpa}, sk^1_{cpa}) \leftarrow \textsc{KeyGenCPA}$$
$$(pk^2_{cpa}, sk^2_{cpa}) \leftarrow \textsc{KeyGenCPA}$$
$$(pk^3_{cpa}, sk^3_{cpa}) \leftarrow \textsc{KeyGenCPA}$$

Also run KeyGenHash get the hash public key pk_{hash}.

$$pk_{hash} \leftarrow \textsc{KeyGenHash}.$$

Define, $pk_1, sk_1, pk_2, sk_2, sk_R$ as follows:

$$\begin{aligned} pk_1 &= (pk^1_{cpa}, pk^3_{cpa}, pk_{hash}) \\ sk_1 &= sk^1_{cpa} \\ pk_2 &= (pk^2_{cpa}, pk^3_{cpa}, pk_{hash}) \\ sk_2 &= sk^2_{cpa} \\ sk_R &= (sk^3_{cpa}, pk_{hash}) \end{aligned}$$

Output

$$(pk_1, sk_1, pk_2, sk_2, sk_R).$$

Enc$_1$: Given $x \in X$, and $pk_1 = (pk^1_{cpa}, pk^3_{cpa}, pk_{hash})$ evaluate

$$\begin{aligned} cx_1 &= \textsc{EncCPA}(pk^1_{cpa}, x) \\ cx_2 &= \textsc{EncCPA}(pk^3_{cpa}, \textsc{Hash}_1(pk_{hash}, x)) \end{aligned}$$

Output

$$cx = (cx_1, cx_2)$$

Dec$_1$: Given $cx = (cx_1, cx_2)$ and $sk_1 = sk^1_{cpa}$ output

$$\textsc{DecCPA}(sk^1_{cpa}, cx_1).$$

Enc$_2$ and Dec$_2$ are defined similarly.

Verify: Given $sk_R = (sk^3_{cpa}, pk_{hash})$, $cx = (cx_1, cx_2)$, $cy = (cy_1, cy_2)$ and $z \in Z$, output

$$\textsc{VerifyHash}(pk_{hash}, \textsc{DecCPA}(sk^3_{cpa}, cx_2), \textsc{DecCPA}(sk^3_{cpa}, cy_2), z).$$

Theorem 12. *The relational encryption scheme defined by*

$$(\textsc{KeyGen}, \textsc{Enc}_1, \textsc{Enc}_2, \textsc{Dec}_1, \textsc{Dec}_2, \textsc{Verify})$$

is IND-CPA secure if (KeyGenCPA, EncCPA, DecCPA) *is an IND-CPA secure relational encryption scheme. Moreover, the relation encryption scheme is*

- *one-way secure if* (KeyGenHash, Hash$_1$, Hash$_2$, VerifyHash) *is one-way secure relational hash*
- *unforgeable if* (KeyGenHash, Hash$_1$, Hash$_2$, VerifyHash) *is unforgeable relational hash*
- *oracle-simulation secure if* (KeyGenHash, Hash$_1$, Hash$_2$, VerifyHash) *is oracle-simulation secure relational hash.*

I.2 Relational Encryption for Linearity in $\mathbb{F}_2^n$

The relational hash scheme for linearity over $\mathbb{F}_2^n$ in Section 3, is in fact a relational encryption scheme if KeyGen outputs the following secret keys,

$$sk_1 := \langle a_i \rangle_{i=1}^n$$
$$sk_2 := \langle b_i \rangle_{i=1}^n$$

along with the public keys and the relational key pk_1, pk_2, sk_R.

Enc$_1$ and Enc$_2$ are exactly same as Hash$_1$ and Hash$_2$. Decryption algorithms would work as follows.

Dec$_1$*:* Given ciphertext $cx = \langle cx_i \rangle_{i=0}^{n+1}$ and $sk_1 = \langle a_i \rangle_{i=1}^n$, the plaintext is constructed, bit by bit, as follows:

$$x_i := \begin{cases} 0 & \text{if } cx_i = cx_0^{a_i} \\ 1 & \text{if } cx_i = cx_0^{-a_i} \\ \perp & \text{else} \end{cases}$$

Dec$_2$*:* Given ciphertext $cy = \langle cy_i \rangle_{i=0}^{n+1}$ and $sk_2 = \langle b_i \rangle_{i=1}^n$, the plaintext is constructed, bit by bit, as follows:

$$y_i := \begin{cases} 0 & \text{if } cy_i = cy_0^{b_i} \\ 1 & \text{if } cy_i = cy_0^{-b_i} \\ \perp & \text{else} \end{cases}$$

I.3 Relational Encryption for Proximity

It is also possible to build a Relational encryption scheme for proximity from

- a family of (n, k, d) linear error correcting code (ECC) $\mathcal{C}$.
- a IND-CPA secure encryption scheme (KeyGenCPA, EncCPA, DecCPA).
- a Relational Encryption scheme for linearity in $\mathbb{F}_2^k$:

(KeyGenLinear, EncLinear$_1$, DecLinear$_1$, EncLinear$_2$, DecLinear$_2$, VerifyLinear).

This is 'slightly' efficient than the generic construction given in Section I.1.

KeyGen*:* Given the security parameter, choose a $(n, k, 2\delta)$ linear error correcting code C, where k is in the same order as the security parameter. Run the key generator algorithms of the CPA secure encryption scheme and the relational encryption for linearity.

- (pk_{cpa}, sk_{cpa}) be the output of KeyGenCPA
- $(pk_{1,lin}, pk_{2,lin}, sk_{1,lin}, sk_{2,lin}, sk_{Rlin})$ be the output of KeyGenLinear

Now we define the output of KeyGen as follows:

$$pk_1 := (\textsc{Encode}, \textsc{Decode}, pk_{cpa}, pk_{1,lin}) \qquad sk_1 := (sk_{cpa}, sk_{1,lin})$$
$$pk_2 := (\textsc{Encode}, \textsc{Decode}, pk_{cpa}, pk_{2,lin}) \qquad sk_2 := (sk_{cpa}, sk_{2,lin})$$
$$sk_R := (sk_{cpa}, sk_{Rlin})$$

Enc$_1$*:* Given plaintext $x \in \mathbb{F}_2^n$ and $pk_1 = (\textsc{Encode}, \textsc{Decode}, pk_{cpa}, pk_{1,lin})$, the ciphertext is constructed as follows: Sample a random $r \leftarrow \mathbb{F}_2^k$ and then compute the following:

$$cx_1 := \textsc{EncCPA}(pk_{cpa}, m + \textsc{Encode}(r))$$
$$cx_2 := \textsc{EncLinear}_1(pk_{1,lin}, r)$$

Publish the final ciphertext $cx := (cx_1, cx_2)$.

Dec$_1$*:* Given ciphertext $cx = (cx_1, cx_2)$ and $sk_1 = (sk_{cpa}, sk_{1,lin})$, the plaintext is constructed, as follows:

$$x := \textsc{DecCPA}(sk_{cpa}, cx_1) + \textsc{DecLinear}_1(sk_{1,lin}, cx_2)$$

Enc$_1$ and Dec$_2$ are defined similarly.

Verify*:* Given the ciphertexts $cx := (cx_1, cx_2)$, $cy := (cy_1, cy_2)$ and $sk_R := (sk_{cpa}, sk_{Rlin})$ verification is done as follows. Note, Verify has access to Decode, which is available in the public key information.

– Recover z as

$$z := \textsc{Decode}(\textsc{DecCPA}(sk_{cpa}, cx_1) + \textsc{DecCPA}(sk_{cpa}, cy_1)).$$

  Output Reject if Decode returns $\perp$.
– Output $\textsc{VerifyLinear}(sk_{Rlin}, cx_2, cy_2, z)$.

I.4 Relational Encryption for Linearity in $\mathbb{F}_p^n$

If p is at most a polynomial in the security parameter[3] λ, i.e., $p = \lambda^{O(1)}$ then similar to Section I.2 the above mentioned relational hash, is in fact a relational encryption scheme. KeyGen would output the following secret keys,

$$sk_1 := \langle a_i \rangle_{i=1}^n$$
$$sk_2 := \langle b_i \rangle_{i=1}^n$$

along with the public keys and the relational key pk_1, pk_2, sk_R.

Enc$_1$ and Enc$_2$ are exactly same as Hash$_1$ and Hash$_2$. Decryption algorithms would work as follows.

Dec$_1$*:* Given ciphertext $cx = \langle cx_i \rangle_{i=0}^{n+1}$ and $sk_1 = \langle a_i \rangle_{i=1}^n$, the plaintext is constructed, bit by bit, as follows:

$$x_i := \begin{cases} \mu & \text{if } cx_i = cx_0^{\omega^{\mu} a_i} \text{ for some } \mu \in \mathbb{F}_p \\ \perp & \text{if no such } \mu \text{ exists} \end{cases}$$

This operation is poly-time in λ, since the number of possibilities for μ is bounded by a polynomial in λ.

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

[3] This implies n must be in the order of security parameter, otherwise p-ary-Mix-DLP assumption (Assumption 2) won't hold.