Relational Hash: Probabilistic Hash for Verifying Relations, Secure against Forgery and More

Avradip Mandal and Arnab Roy

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

Abstract. Traditional cryptographic hash functions allow one to easily check whether the original plaintexts are equal or not, given a pair of hash values. Probabilistic hash functions extend this concept where given a probabilistic hash of a value and the value itself, one can efficiently check whether the hash corresponds to the given value. However, given distinct probabilistic hashes of the same value it is not possible to check whether they correspond to the same value. In this work we introduce a new cryptographic primitive called *Relational Hash* using which, given a pair of (relational) hash values, one can determine whether the original plaintexts were related or not. We formalize various natural security notions for the Relational Hash primitive - one-wayness, twin one-wayness, unforgeability and oracle simulatibility.
We develop a Relational Hash scheme for discovering linear relations among bit-vectors (elements of $\mathbb{F}_2^n$) and $\mathbb{F}_p$-vectors. Using the linear Relational Hash schemes we develop Relational Hashes for detecting proximity in terms of hamming distance. The proximity Relational Hashing schemes can be adapted to a privacy preserving biometric identification scheme, as well as a privacy preserving biometric authentication scheme secure against passive adversaries.

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

1 Introduction

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

enabling equality checking. This typically has come at the price of efficiency. In addition, the property of compression, which is desirable for deterministic hash functions, is no longer at the forefront.

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

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

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

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

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

simulatability was introduced in [Can97,CMR98] for the equality relation. Here we extend this concept for arbitrary relations.

For the equality relation, there is a simple construction which extends Canetti's scheme in [Can97]. While the [Can97] probabilistic hash on a plaintext m and randomness r is $(\mathbf{g}^r, \mathbf{g}^{rm})$, one can consider bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and define $ph_1(x_1, r_1) := (\mathbf{g}^{r_1}, \mathbf{g}^{r_1 x_1})$ and $ph_2(x_2, r_2) := (\mathbf{h}^{r_2}, \mathbf{h}^{r_2 x_2})$ with $\mathbf{g} \in \mathbb{G}_1$ and $\mathbf{h} \in \mathbb{G}_2$. Plaintext equality of two hashes (c_1, c_2) and (d_1, d_2) of different types can be done as: $e(c_1, d_2) \stackrel{?}{=} e(c_2, d_1)$. We do not develop this construction formally in the body of the paper, additionally relegating some proof sketches to the full version [MR14][2].

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

Application. A motivating application of the proximity relation hash primitive is a privacy preserving biometric identification scheme. Consider a scenario where there is a database of fingerprints of known criminals. The database should not reveal the actual fingerprints, even internally. An investigative officer might want to check, whether a candidate fingerprint digest matches with the database. Using a Relational Hash scheme for proximity relation, one can build a biometric identification scheme which guarantees complete template privacy (to the server, as well as to the investigating officer). While storing the fingerprints in the database, hashes of type 1 are used. On the other hand, the officer gets access to type 2 hash of the fingerprint template. The Relational Hash scheme will guarantee that, with access to a relational secret key the server can only verify whether the original templates are close to each other or not. To construct authentication schemes, rather than identification schemes, additional protocol layers are needed to address replay attacks and so on. Merely providing a type 2 hash of the challenge biometric template does not suffice as that can easily be replayed. We leave open the construction of such protocols building on the Relational Hash primitive. However, we show that for the case of a passive adversary attempting to recover the biometric template, a Relational Hash can be seen as a biometric authentication mechanism (Section 4).

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

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

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

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

Thus we have to resort to the simulation based security notion (FE-SIM) for any meaningful assurance of security. The only possibility result in the fully public key setting is given by [GKL$^+$13], who give a construction of FE-SIM secure encryption scheme for a class of functionalities they call "learnable" functions. They also prove that if an FE-SIM secure scheme exists for a class of functionalities, then this class must be learnable. Briefly, a 2-ary function $f(.,.)$ is learnable if, given a description of f and oracle access to $f(x,.)$, one can output the description of a function that is indistinguishable from $f_x(.)$, which is

the restriction of f on fixing the first input to x. This has to hold true with high probability even if the distinguisher is given x. One can immediately see that equality is not a learnable function. When x comes from high min-entropy distribution, it is not possible to learn the value of x efficiently by querying $f(x, .)$ on various inputs. A distinguisher can immediately thwart any such 'learnt' function by simply testing it on x.

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

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

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

Organization of the paper. In Section 2, we formally define the notion of Relational Hash and its desired security properties. In Section 3, we construct a Relational Hash for linearity over $\mathbb{F}_2^n$, with extension to $\mathbb{F}_p^n$. In Section 4, we show how to construct a proximity (in terms of hamming distance) Relational Hash using a linear Relational Hash and a linear error correcting code. In Section 5, we describe relations among notions of security for constructing Relational Hashes for various relations. Standard hardness assumptions are summarized in Appendix A. We defer the proof of unproven theorems in this paper to the full version [MR14].

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.

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

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

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

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

$$\Pr\left[\begin{array}{l} pk \leftarrow \textsc{KeyGen}(1^\lambda) \\ hx \leftarrow \textsc{Hash}_1(pk, x) \\ hy \leftarrow \textsc{Hash}_2(pk, y) \end{array} : \quad \textsc{Verify}(pk, hx, hy, z) \equiv R(x, y, z)\right] \approx 1$$

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

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

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

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

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

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

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

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

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

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

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

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

- $pk \leftarrow \textsc{KeyGen}(1^\lambda)$, $x \leftarrow \mathcal{X}$, $y \leftarrow \mathcal{Y}$, $hx \leftarrow \textsc{Hash}_1(pk, x)$, $hy \leftarrow \textsc{Hash}_2(pk, y)$
- *For any PPT adversary A_1, there exists a negligible function* $\texttt{negl}()$*, such that:* $\Pr[(hy', z) \leftarrow A_1(pk, hx) \wedge \textsc{Verify}(pk, hx, hy', z) = 1] < \texttt{negl}(\lambda)$
- *For any PPT 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 the author was interested in regular probabilistic hash functions. In case of Relational Hash functions, we want to say that: having $hx = \textsc{Hash}_1(pk, x)$ gives no information on x, besides the ability to evaluate the value of $R(x, y, z)$ for any y, z chosen from their respective domains. Similarly, $hy = \textsc{Hash}_1(pk, y)$ should not provide any extra information other than the ability to evaluate the value of $R(x, y, z)$ for any $x \in X$ and $z \in Z$. Also, having access to both hx and hy, one should be able to only evaluate $R(x, y, z)$ for any $z \in Z$.

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

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

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

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

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

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

where $(x,y) \leftarrow \Psi$ and $pk \leftarrow \text{KeyGen}(1^\lambda)$.

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. Correctness of the scheme follows from standard algebraic manipulation of pairing operations. Details are given in [MR14].

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

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

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

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

In the Binary Mix DLP problem, the difference is that the $\mathbf{g}_i$'s are independently random and that instead of raising the $\mathbf{g}_i$'s to the powers 0 or 1, we raise them to the powers ± 1. This is, of course, not a formal proof of its hardness. In [MR14], we show that the Binary Mix DLP assumption can actually be reduced to the more standard Random Modular Subset Sum assumption [Lyu05]. As an added assurance, in [MR14], we show that the Binary Mix DLP assumption is also secure in the Generic Group Model [Sho97].

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

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

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

claims our scheme is indeed twin one-way secure in this case and is proved in [MR14].

Theorem 2. *The above algorithms* (KeyGen, Hash_1, Hash_2, Verify) *constitute a Relational Hash scheme for the relation* $R = \{(x, y, z) \mid x+y = z \wedge x, y, z \in \mathbb{F}_2^n\}$. *The scheme is twin one-way secure in the generic group model, when* x *is sampled uniformly from* $\mathbb{F}_2^n$ *and* $y = x$.

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

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

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

it is easy to construct

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

for any $t \in \mathbb{F}_2^n$. Hash_1 is also homomorphic in a similar manner. However, this does not really refute any of our security claims. In fact, in next section we will see this linear homomorphism gives us strong security guarantee for relation hash construction for hamming proximity (Theorem 4).

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

Relational Hash for Linearity in $\mathbb{F}_p^n$: For any prime p, we can choose the order q of the bilinear groups to be exponential in the security parameters as well as equal to 1 (mod p). This means the group $\mathbb{Z}_q^*$ has a subgroup $\mathbb{J}_p$ of prime order p. Let ω be an arbitrary generator of $\mathbb{J}_p$. We can publish this arbitrary generator as part of the public key. For Hash_1 evaluation (similarly in Hash_2), we can simply calculate hx_i as $\mathbf{g}_i^{\omega^{x_i} r}$ (instead of $\mathbf{g}_i^{(-1)^{x_i} r}$). Similarly during verification, instead of checking $e(hx_0, hy_0)^{pk_R} \stackrel{?}{=} e(hx_{n+1}, hy_{n+1}) \prod_{i=1}^n e(hx_i, hy_i)^{(-1)^{z_i}}$, we can just check $e(hx_0, hy_0)^{pk_R} \stackrel{?}{=} e(hx_{n+1}, hy_{n+1}) \prod_{i=1}^n e(hx_i, hy_i)^{\omega^{-z_i}}$. We provide the details in [MR14].

4 Relational Hash for Hamming Proximity

In this section we construct a Relational Hash for the domains $X, Y = \mathbb{F}_2^n$ and the relation[4] $R_\delta = \{(x, y) \mid \texttt{dist}(x, y) \leq \delta \wedge x, y \in \mathbb{F}_2^n\}$, where `dist` is

[4] Note that Relational Hash is defined over 3-tuple relations (Definition 2). However, here proximity encryption is defined over 2-tuple relations. 2-tuple relations can

the hamming distance and δ is a positive integer less than n. Specifically, we construct a Relational Hash for proximity from a family of binary (n, k, d) linear error correcting codes (ECC) $\mathcal{C}$ and a Relational Hash for linearity in $\mathbb{F}_2^k$: (KeyGenLinear, HashLinear$_1$, HashLinear$_2$, VerifyLinear).

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

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

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 $hx := (hx_1, hx_2)$.

Hash$_2$ is defined similarly.

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

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

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

---

be regarded as special cases of 3-tuple relations, where the third entry does not matter. E.g. the relation $R'_\delta \subseteq \mathbb{F}_2^n \times \mathbb{F}_2^n \times Z$ (where Z is any non empty domain) and $(x, y, z) \in R'_\delta$ if and only if $(x, y) \in R_\delta$.

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

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

where the e_i's are sampled from some known noise distribution Ξ, such that with high probability we have $\texttt{weight}(e_i) \leq \delta$. We now show that, given $\textsc{Hash}_1(x)$ and any number of samples[5] $\textsc{Hash}_2(y_i)$, the attacker cannot output z, such that $\texttt{dist}(x, z) \leq \delta$. The proof, which is a reduction to twin one-wayness of the linear Relational Hash is given in [MR14].

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

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

If we assume that the biometric templates of individuals follow uniform distribution over $\{0,1\}^n$, then Theorem 3 would imply that the server can never recover the original biometric template x. Moreover, the unforgeability property guarantees that even if the server's database gets leaked to an attacker then also the attacker cannot come up with a forged hy', which would authenticate the

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

attacker. Theorem 4 will guarantee that even with access to the registered hash and several authentication transcripts from the same individual, the biometric template will remain private to the server.

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

5 Relation among Notions of Security for Relational Hashes

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

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

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

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

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

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

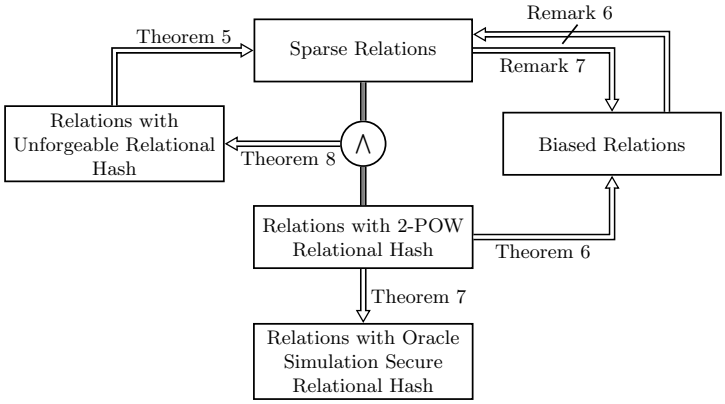

Fig. 1. Relationship among Types of Relations. Arrowhead indicates direction of implication. Strike on an arrow indicates the existence of a counter-example.

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

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

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

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

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

Definition 8 (2-value Perfectly One-Way function). *Let $\mathcal{X}$ be a probability distribution over X. Let $H = \{h_k\}_{k \in K}$ be a keyed probabilistic function family*

with domain X and randomness space U, where the key k gets sampled from a probability distribution $\mathcal{K}$ over K. H is 2-value perfectly one-way (POW) with respect to $\mathcal{X}$ and $\mathcal{K}$ if for any PPT distinguisher D,

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

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

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

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

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

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

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

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

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

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

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

Proof. We are given that,

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

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

Theorem 7, stated below, shows that if a Relational Hash is 2-POW, then it is also oracle simulation secure.

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

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

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

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

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

Proof. Assume that the scheme is not unforgeable. This means that given $(pk, \textsc{Hash}_1(pk, x, r))$ for $x \leftarrow \mathcal{X}$, there is an attacker A, which outputs $\textsc{Hash}_2(pk, y, s)$ and z, such that $R(x, y, z) = 1$, with non-negligible probability. Using A, we now

build an attacker B which distinguishes the distributions $(pk, \text{HASH}_1(pk, x, r_1), \text{HASH}_1(pk, x, r_2))$ and $(pk, \text{HASH}_1(pk, x, r_1), \text{HASH}_1(pk, x', r_2))$ with non-negligible probability. Given $(pk, \text{HASH}_1(pk, x, r_1), \text{HASH}_1(pk, w, r_2))$, B sends $\text{HASH}_1(pk, x, r_1)$ to A. With non-negligible probability A outputs $\text{HASH}_2(pk, y, s)$ and z, such that $R(x, y, z) = 1$. Now since R is a sparse relation, if $w \neq x$, then with non-negligible probability $R(w, y, z) = 0$, whereas if $w = x$, then $R(w, y, z) = 1$. Now $R(w, y, z)$ can be efficiently evaluated by computing $\text{VERIFY}(pk, \text{HASH}_1(pk, w, r_2), \text{HASH}_2(pk, y, s), z)$. 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 9, we show that the Relational Hash construction for linearity over $\mathbb{F}_2^n$ from Section 3 is actually a 2-value perfectly one-way function. This property is based on a stronger hardness assumption called Decisional Binary Mix (Assumption 2). In [MR14] 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 7 and Theorem 8 we get that the Relational Hash construction from Section 3 is actually oracle simulation secure as well as unforgeable with respect to the independent uniform distributions over $\mathbb{F}_2^n$.

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

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

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

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

than what the simulation world can provide. We leave it as an open problem to construct 2-POW and Oracle Simulation secure Relational Hashes for proximity.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

MR14. Avradip Mandal and Arnab Roy. Relational hash. Cryptology ePrint Archive, Report 2014/394, 2014. `http://eprint.iacr.org/2014/394`.

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

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

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

A Hardness Assumptions

We summarize the standard hardness assumptions used in this paper.

Assumption 3 (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*

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

Assumption 4 (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 5 (Random Modular Subset Sum [Lyu05]). *Assuming a generation algorithm $\mathcal{G}$ that outputs a tuple (n, q),where q is prime,the Random Modular Subset Sum assumption asserts that given random elements $\langle a_i \rangle_{i=1}^n$ from the group $\mathbb{Z}_q$ and $c = \sum_{i=1}^n \epsilon_i a_i$ for a random $\epsilon \leftarrow \{0,1\}^n$, it is hard to output $\eta \in \{0,1\}^n$ such that*

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

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

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