Journal of the Nigerian Association of Mathematical Physics
Volume **12 (May, 2008), 485 - 494**

Universal hash proof system and the subset membership problem

Tola John Odule,
Department of Mathematical Sciences
Olabisi Onabanjo University, Ago-Iwoye, Ogun State, Nigeria

Abstract

The notion of a universal hash proof system is hereby introduced. Essentially, this is a special kind of non-interactive zero-knowledge proof system for a language. We showed how to construct very efficient universal hash proof systems for a general class of group-theoretic language membership problems. We showed how to construct efficient universal hash proof systems for languages related to the Decision Composite Residuosity (DCR) and Quadratic Residuosity (QR) assumptions. From these one can get corresponding public-key encryption schemes that are secure under these assumptions.

Keywords: Cryptographic Hash Function Statistical Distance, Random Oracle, Projective Hash, Subset membership

1.0 Introduction

The philosophy behind a universal one-way hash function (UOWHF) is that if, first the input is selected and subsequently the hash function, it does not help an opponent to find collisions for the hash function [1]. Collisions are only useful if first the function is fixed and subsequently one can search for two colliding inputs.

This definition was generalized in [2], where a UOWHF is defined as a three party game with an initial string supplier S, a hash function instance generator G and a collision string finder F. Here S is an oracle with unlimited computing power, and G and F are probabilistic polynomial time algorithms. The game consists of three moves:

1. S outputs an initial string $x \in \sum^n$ and sends it to both G and F.
2. G chooses an $h \in_R H_n$ independently of x and sends it to F.
3. F outputs either "?" or an $x' \in \sum^n$ such that $h(x') = h(x)$.

F wins the game if its output is not equal to "?". The input x is selected by S according to a certain distribution. In the most general case this is the collection of all ensembles with length n. If a different ensemble is introduced, a different definition is obtained. In the original definition of [3] the initial string supplier and the collision string finder were the same algorithm, which imposes the unnecessary restriction that x should be selected according to all polynomially samplable ensembles (the collision string finder has to be a polynomial time algorithm). The construction by M. Naor and M. Yung [4] also satisfies this more general definition. On the other hand their definition is less complicated: in fact it does not really make sense for S to send x to G, as G chooses subsequently h independent from x. In [2, 5] the hierarchy between different types of UOWHF has been studied.

e-mail: tee_johnny@yahoo.com

2.0 Preliminaries

We recall some basic terminology and notation.

A function $f(\lambda)$ mapping non-negative integers to non-negative reals is called negligible $(in\ \lambda)$ if for all $c \geq 1$, there exists $\lambda_0 > 0$ such that $f(\lambda) \leq 1/\lambda^c$ for all $\lambda \geq \lambda_0$. Let X and Y be random variables taking values in a finite set S. The statistical distance between X and Y is defined to be $Dist(X,Y) = \frac{1}{2}.\sum_{s\in S}|\Pr[X=s]-\Pr[Y=s]|$. Equivalently, $Dist(X,Y) = \max_{S'\subset S}|\Pr[X\in S']-\Pr[Y\in S']|$. We shall say that X and Y are $\in$ -close if $Dist(X,Y) \leq \in$.

Let $\mathbf{X} = (X_\lambda)_{\lambda\geq 0}$ and $\mathbf{Y} = (Y_\lambda)_{\lambda\geq 0}$ be sequences of random variables, where for each $\lambda \geq 0$, X_λ and Y_λ take values in a finite set S_λ. Then we say that **X** and **Y** are *statistically indistinguishable* if $Dist(X_\lambda, Y_\lambda)$ is a negligible function in λ. For computational purposes, we will generally work in a setting where the sets S_λ can be encoded as bit strings whose length is polynomial in λ. For any probabilistic algorithm A that outputs 0 or 1, we define the *distinguishing advantage for* A (with respect to X and Y) as the function $Dist_{A}^{X,Y}(\lambda) = \left|\Pr[A(1^\lambda, X_\lambda)=1] - \Pr[A(1^\lambda, Y_\lambda)=1]\right|$

Here, the notation 1^λ denotes the unary encoding of λ as a sequence of λ copies of 1, and the probability is with respect to the random coin tosses of the algorithm A and the distributions of X_λ and Y_λ. We say that **X** and **Y** are *computationally indistinguishable* if for all probabilistic, polynomial-time A, the function $Dist_{A}^{X,Y}(\lambda)$ is negligible in λ.

For a positive integer Z, Z_N denotes the ring of integers modulo N, and Z_N^* denotes the corresponding multiplicative group of units. For $a \in Z, (a \bmod N) \in Z_N$ denotes the residue class of a modulo N. For an element g of a group G, $\langle g\rangle$ denotes the subgroup of G generated by g. Likewise, for a subset U of G, $\langle U\rangle$ denotes the subgroup of G generated by U.

2.1 Universal hashing

Before defining universal projective hash functions, we recall some definitions relating to the classical notion of "universal hashing" [6,7].

Let X and $\prod$ be finite, non-empty sets. Let $H = (H_k)_{k\in K}$ be a collection of functions indexed by K, so that for every $k \in K$, H_k is a function from X into $\prod$. Note that we may have $H_k = H_{k'}$ for $k \neq k'$. We call $\mathbf{F} = (H, K, X, \prod)$ a hash family, and each H_k a hash function.

***Definition* 2.1**

Let $\mathbf{F} = (H, K, X, \prod)$ be a hash family, and consider the probability space defined by choosing $k \in K$ at random. We call $\mathbf{F}$ pair-wise independent if for all $x, x^* \in X$ with $x \neq x^*$, it holds that $H_k(x)$ and $H_k(x^*)$ are uniformly and independently distributed over $\prod$.

Note that there are many well-known, and very simple constructions of pair-wise independent hash families.

2.2 Universal projective hashing

We now introduce the concept of universal projective hashing. Let $\mathbf{F} = (H, K, X, \prod)$ be a hash family. Let L be a non-empty, proper subset of X. Let S be a finite, non-empty set, and let $\alpha: K \to S$ be a function. Set $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$.

Definition 2.2

$\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$, defined as above, is called a *projective hash family* (for (X, L)) if for all $k \in K$, the action of H_k on L is determined by $\alpha(k)$.

In other words, for all $k \in K$, the value $\alpha(k)$ "encodes" the action of H_k on L (and possibly more than that), so that given $\alpha(k)$ and $x \in L$, the value $H_k(x)$ is uniquely determined.

Definition 2.3

Let $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$ be a projective hash family, and let $\varepsilon \geq 0$ be a real number. Consider the probability space defined by choosing $k \in K$ at random. We say that $\mathbf{H}$ is ε-*universal* if for all $s \in S$, $x \in X \setminus L$, and $\pi \in \prod$, it holds that:

$$\Pr[H_k(x) = \pi \wedge \alpha(k) = s] \leq \varepsilon \Pr[\alpha(\mathrm{k}) = \mathrm{s}]$$

We say that $\mathbf{H}$ is $\varepsilon - universal_2$ if for all $s \in S$, $x, x^* \in X$ and $\pi, \pi^* \in \prod$ with $x \notin L \cup \{x^*\}$, it holds that:

$$\Pr[H_k(x) = \pi \wedge H_k(x^*) = \pi^* \wedge \alpha(k) = \mathrm{s}] \leq \varepsilon \Pr[H_k(x^*) = \pi^* \wedge \alpha(\mathrm{k}) = \mathrm{s}]$$

We will sometimes refer to the value of $\mathbf{H}$ in the above definition as the *error rate* of $\mathbf{H}$.

Note that if $\mathbf{H}$ is $\varepsilon - universal_2$, then it is also $\varepsilon - universal$ (note that $|X| \geq 2$).

2.2.0 Interpretation

We can reformulate the above definition as follows. Let $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$ be a projective hash family, and consider the probability space defined by choosing $k \in K$ at random. $\mathbf{H}$ is $\varepsilon - universal$ means that conditioned on a fixed value of $\alpha(k)$, even though the value of H_k is completely determined on L, for any $x \in X \setminus L$, the value of $H_k(x)$ can be guessed with probability at most ε. $\mathbf{H}$ is $\varepsilon - universal_2$ means that in addition, for any $x^* \in X \setminus L$, conditioned on fixed values of $\alpha(k)$ and $H_k(x^*)$, for any $x \in X \setminus L$ with $x \neq x^*$, the value of $H_k(x)$ can be guessed with probability at most ε.

2.2.1 Justification

We now discuss the justification for *Definition* 2.3.

Let $\mathbf{H}$ be a projective hash family, and consider the following game played by an adversary. At the beginning of the game, $k \in K$ is chosen at random, and the adversary is given $s = \alpha(k)$. Initially, the adversary has no other information about k, but during the course of the game, he is allowed to make a sequence of oracle queries to learn more about k.

There are two types of oracle queries [8,9]. One type of oracle query is a *test query*: the adversary submits $x \in X$ and $\pi \in \prod$ to the oracle, and the oracle tells the adversary whether or not $H_k(x) = \pi$ The other type of oracle query is an *evaluation query*: the adversary submits $x^* \in X$ to the oracle, and the oracle tells the adversary the value $\pi^* = H_k(x^*)$.

During the course of the game, the adversary is allowed to make an arbitrary number of *test queries*, but only one *evaluation query*. Moreover, after the evaluation query, he is not allowed to submit (x^*, π^*) to the oracle in any subsequent test queries. We say the adversary wins the game if he submits a test query (x, π) with $x \in X \setminus L$ and $H_k(x) = \pi$.

That completes the description of the game. Note that in this game, the adversary's strategy is quite arbitrary, and need not be efficiently computable. Moreover, the strategy may be adaptive, in the sense that an oracle query made by the adversary may depend in an arbitrary way on all information available to the adversary at that time.

It is easy to see from the definition that if $\mathbf{H}$ is $\varepsilon - universal_2$, then regardless of the adversary's strategy, he wins the game with probability at most $Q.\varepsilon$, where Q is a bound on the number of test queries made by the adversary. Note that while this property is a consequence of the definition of $\varepsilon - universal_2$, it is not necessarily equivalent to the definition of $\varepsilon - universal_2$. In fact, this property suffices to prove the main results of this paper, and indeed, all we need is this property in the case where x^* is chosen at random from $X \setminus L$, and where the adversary is computationally bounded.

2.2.2 Initial constructions

Families satisfying *Definition* 2.2 are simple to construct, at least from a combinatorial point of view. For instance, let $\mathbf{F} = (H, K, X, \prod)$ be a pair-wise independent hash family, let L be a non-empty, finite subset of X, and let $\pi_0 \in \prod$. Then let $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$, where for all $k \in K$ and $x \in X$, we define $H_k'(x) = \pi_0$ if $x \in L$, and $H_k'(x) = H_k(x)$, otherwise. We also define $S = \{\pi_0\}$ and $\alpha(k) = \pi_0$ for all $k \in K$. It is clear that $\mathbf{H}$ is a $1/|\mathbf{H}| - universal_2$ projective hash family. However, in our applications later on, we want these hash functions to be efficiently computable on all of X, even if L is hard to distinguish from $X \setminus L$. Therefore, this trivial "solution"" is not useful in our context.

2.3 Smooth projective hashing

We will need a variation of universal projective hashing, which we call smooth projective hashing. Let $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$ be a projective hash family. We define two random variables, $U(\mathbf{H})$ and $V(\mathbf{H})$, as follows. Consider the probability space defined by choosing $k \in K$ at random, $x \in X \setminus L$ at random, and $\pi' \in \prod$ at random. We set $U(\mathbf{H}) = (x, s, \pi')$ and $V(\mathbf{H}) = (x, s, \pi)$, where $s = \alpha(k)$ and $\pi = H_k(x)$.

***Definition* 2.4**

Let $\varepsilon \geq 0$ be a real number. A projective hash family $\mathbf{H}$ is $\varepsilon - smooth$ if $U(\mathbf{H})$ and $V(\mathbf{H})$ are ε-close.

2.4 Approximations to projective hash families

Our definitions of *universal* and $universal_2$ projective hash families are quite strong: so strong, in fact, that in many instances it is impossible to efficiently implement them. However, in all our applications, it is sufficient to efficiently implement a projective hash family that effectively approximates a *universal* or $universal_2$ projective hash family. To this end, we define an appropriate notion of distance between projective hash families.

Let $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$ be a projective hash family. Consider the distribution defined by sampling $k \in K$ at random, and define the random variable $View(\mathbf{H}) = (H_k, \alpha(k))$. Note that $View(\mathbf{H})$ comprises the value of H_k at all points $x \in X$.

***Definition* 2.5**

Let $\delta \geq 0$ be a real number.

Let $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$ and $\mathbf{H}^* = (H^*, K^*, X, L, \prod, S, \alpha^*)$ be projective hash families. We say that $\mathbf{H}$ and $\mathbf{H}^*$ are δ-close if $View(\mathbf{H})$ and $View(\mathbf{H})$ are δ-close.

Note that if $\mathbf{H}$ and $\mathbf{H}^*$ are δ-close for some "small" value of δ, and if $\mathbf{H}^*$ is $\varepsilon - universal$ or $\varepsilon - universal_2$ for some "small" value of ε, this does not imply that $\mathbf{H}$ is $\varepsilon' - universal$ or $\varepsilon' - universal_2$ for any particularly small value of ε'. However, if $\mathbf{H}$ and $\mathbf{H}^*$ are δ-close and $\mathbf{H}^*$ is ε-smooth, then it is clear that $\mathbf{H}$ is $(\varepsilon + \delta)$-smooth.

2.5 Initial reductions

We show the initial reductions among the various notions introduced. Most of the reductions given here are primarily theoretically motivated. Later on, in a specialized context, we present reductions that are considerably more efficient.

2.5.1 Reducing the error rate

Let $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$ be an ε-universal (respectively, -universal$_2$) projective hash family. The construction below reduces the error rate from ε to ε^t, by simple t-fold "parallelization."

Let t be a positive integer, and let $\overline{\mathbf{H}} = (\overline{H}, K^t, X, L, \prod^t, S^t, \overline{\alpha})$, where $\overline{\mathbf{H}}$ and $\overline{\alpha}$ are defined as follows.

For $\vec{k} = (k_1, \Lambda, k_t) \in K^t$ and $x \in X$, we define $H_{\vec{k}}(x) = (H_{k_1}(x), \Lambda, H_{k_t}(x))$, and we define $\overline{\alpha}(\vec{k}) = (\alpha(k_1), \Lambda, \alpha(k_t))$.

The proof of the following lemma is straightforward, and is left to the reader.

***Lemma* 2.1**

Let $\mathbf{H}$ *and* $\overline{\mathbf{H}}$ *be as in the above construction. If* $\mathbf{H}$ *is an* ε*-universal (respectively,* $-universal_2$*) projective hash family, then* $\overline{\mathbf{H}}$ *is an* ε^t*-universal (respectively,* $-universal_2$*) projective hash family.*

2.5.2 From universal projective to universal2 projective

Let $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$ be an ε-universal projective hash family. The next construction turns $\mathbf{H}$ into an $\varepsilon - universal_2$ projective hash family $\mathbf{H}^{\mu}$ for (X, L).

Let us assume that we have injective functions $\Gamma: X \to \{0,1\}^n$ and $\Gamma': \prod \to \{0,1\}^{n'}$ for some appropriately large positive integers n and n'.

Let $\mathbf{H}^{\mu} = (\mathbf{H}^{\mu}, \mathrm{K}^{2n}, X, L, \{0,1\}^{n'}, S^{2n}, \alpha^{\mu})$, where $\mathbf{H}^{\mu}$ and $\alpha^{\dagger}$ are defined as follows.

For $\vec{k} = (k_{1,0}, k_{1,1} \Lambda, k_{n,0}, k_{n,1}) \in K^{2n}$ and $x \in X$ with $\Gamma(x) = (\gamma_1, \Lambda, \gamma_n) \in \{0,1\}^n$, we define

$$\mathbf{H}^{\mu}_{\mathbf{k}}(x) = \bigoplus_{i=1}^{n} \Gamma'\left(H_{k_i, \tau_i}(x)\right)$$

and $$\alpha^{\mu}(k) = (\alpha(k_{1,0}), \alpha(k_{1,1}), \Lambda\ , \alpha(k_{n,0}), \alpha(k_{n,1})),$$

Here, " $\oplus$ " denotes the bit-wise "exclusive or" operation on n' -bit strings.

***Lemma* 2.2**

Let $\mathbf{H}$ *and* $\mathbf{H}^{\mu}$ *be as defined in the above construction. If* $\mathbf{H}$ *is an* ε *-universal projective hash family, then* $\mathbf{H}^{\mu}$ *is an* $\varepsilon - universal_2$ *projective hash family.*

Proof:

It is immediate that Definition 2.2 is satisfied.

The proof that *Definition* 2.3 is satisfied follows from a simple "conditioning argument," the details of which we now provide.

Consider the probability space defined by choosing $\vec{k} \in K^{2n}$ at random. To show that $\mathbf{H}^{\mu}$ is $\varepsilon - universal_2$, we have to show that for any $x, x^* \in X$ with $x \notin L \cup \{x^*\}$, conditioned on any fixed values of $H_{\vec{k}}^{\mu}(x)$ and $\alpha^{\mu}(\vec{k})$ the value of $H_{\vec{k}}^{\mu}(x)$ can be guessed with probability at most ε.

Let $\Gamma(x) = (\gamma_1, \Lambda\ , \gamma_n) \in \{0,1\}^n$ and $\Gamma(x^*) = (\gamma_1^*, \Lambda\ , \gamma_n^*) \in \{0,1\}^n$. Since $x \neq x^*$, we must have $\gamma_i \neq \gamma_i^*$ for some $1 \leq i \leq n$, and without loss of generality, let us assume that $i = n$.

In addition to conditioning on fixed values of $H_{\vec{k}}^{\mu}(x)$ and $\alpha^{\mu}(\vec{k})$, let us further condition on fixed values of $k_{1,0}, k_{1,1}, \Lambda\ , k_{n-1,0}, k_{n-1,1}$, as well as k_{n,γ_n^*} (consistent with the fixed values of $H_{\vec{k}}^{\mu}(x)$ and $\alpha^{\mu}(\vec{k})$. In this conditional probability space, the value of $H_{\vec{k}}^{\mu}(x)$ determines the value of $H_{k_{n,\gamma n}}(x)$ and thus, if the value of $H_{\vec{k}}^{\mu}(x)$ could be guessed with probability greater than ε, then so could the value of $H_{k_{n,\gamma n}}(x)$. But since $\mathbf{H}$ is ε-universal, it follows that the value of $H_{k_{n,\gamma n}}(x)$ cannot be guessed with probability greater than ε. We conclude that value of $H_{\vec{k}}^{\mu}(x)$ cannot be guessed with probability greater than ε in this conditional probability space. Since this holds for all fixed values of $k_{1,0}, k_{1,1}, \Lambda\ , k_{n-1,0}, k_{n-1,1}$ and k_{n,γ_n^*} under consideration, it holds as well in the conditional probability space where just $H_{\vec{k}}^{\mu}(x)$ and $\alpha^{\mu}(\vec{k})$ are fixed, which proves the theorem.

The following construction is a variation on *Lemma* 2.2. It extends the sets X and L by taking the Cartesian product of these sets with a fixed, finite set E. Such extensions will prove useful in the sequel.

Let $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$ be an ε-universal projective hash family. Let E be a non-empty, finite set.

Let us assume that we have injective functions $\Gamma: X \times E \to \{0,1\}^n$ and $\Gamma': \prod \to \{0,1\}^{n'}$ for some appropriately large positive integers n and n'.

Let $\mathbf{H}^{\beta} = (H^{\beta}, K^{2n}, X \times E, L \times E, \{0,1\}^{n'}, S^{2n}, \alpha^{\beta})$, where H^{β} and α^{β} are defined as follows. For

$$\vec{k} = (k_{1,0}, k_{1,1} \Lambda\ , k_{n,0}, k_{n,1}) \in K^{2n},$$

and $(x, e) \in X \times E$ with $\Gamma(x, e) = (\gamma_1, \Lambda\ , \gamma_n) \in \{0,1\}^n$, we define

$$H_{\vec{k}}^{\beta}(x, e) = \bigoplus_{i=1}^{n} \Gamma'(H_{ki,\gamma i}(x))$$

and

$$\alpha^{\beta}(\vec{k}) = (\alpha(k_{1,0}), \alpha(k_{1,1}), \Lambda\ , \alpha(k_{n,0}), \alpha(k_{n,1}))$$

The proof of the following lemma is essentially the same as the proof of *Lemma* 2.2.

***Lemma* 2.3**

Let $\mathbf{H}$ and $\mathbf{H}^{\beta}$ be as defined in the above construction. If $\mathbf{H}$ is an ε-universal projective hash family, then $\mathbf{H}^{\beta}$ is an $\varepsilon - universal_2$ projective hash family.

2.5.3 From universal projective to smooth projective

Let $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$ be an ε-universal projective hash family. The next construction turns $\mathbf{H}$ into a δ-smooth projective hash family $\mathbf{H}^*$ for (X, L), where the hash outputs are a-bit strings, provided ε and a are not too big, and δ is not too small.

The construction is a simple application of the Leftover Hash Lemma (Entropy Smoothing Lemma; (see, e.g., [10, p. 86]).

Let $\mathbf{F} = (\breve{H}, \breve{K}, \prod, \breve{\prod})$ be a pair-wise independent hash family, where $\breve{\prod} = \{0,1\}^n$ for some integer $a \geq 1$. Such a hash family can easily be constructed using well-known and quite practical techniques based on arithmetic in finite fields. We do not discuss this any further here. Let

$$\mathbf{H}^* = (H^*, K \times K, X, L, \breve{\prod}, S \times K, \alpha^*),$$

where H^* and α^* are defined as follows. For

$$k \in K,\ \breve{k} \in \breve{K}, \text{ and } x \in X,$$

we define

$$H^*_{k,\breve{k}}(x) = \breve{H}_{\breve{k}}(H_k(x)),$$

and we define

$$\alpha^*(k, \breve{k}) = (\alpha(k), \breve{k}).$$

***Lemma* 2.4**

Let $\mathbf{H}$, , $\mathbf{F}$ $\mathbf{H}^*$, and a be as in the above construction. Suppose that $\mathbf{H}$ is an ε-universal projective hash family. For any integer $b \geq 0$ such that $a + 2b \leq \log_2(1/\in)$, $\mathbf{H}^*$ is a $2^{-(b+1)}$-smooth projective hash family.

Proof:

It is clear that $\mathbf{H}^*$ satisfies the basic requirements of a projective hash family. Consider the random variables $U(\mathbf{H}^*)$ and $V(\mathbf{H}^*)$, as defined in the paragraph preceding *Definition 3*. That is, consider the probability space where $k \in K$, $\breve{k} \in \breve{K}$, $x \in X \setminus L$, and $\breve{\pi}' \in \breve{\prod}$ are chosen at random, and set

$$U(\mathbf{H}^*) = (x, s, \breve{k}, \breve{\pi}') \text{ and } V(\mathbf{H}^*) = (x, s, \breve{k}, \breve{\pi}),$$

where $s = \alpha(k)$ and $\breve{\pi} = (\breve{H}_{\breve{k}}(H_k(x)))$.

Consider any conditional probability space where particular values of $x \in X \setminus L$ and $s \in S$ are fixed, and let $U(\mathbf{H}^* \mid x, s)$ and $V(\mathbf{H}^* \mid x, s)$ be the random variables in this conditional probability space corresponding to $U(\mathbf{H}^*)$ and $V(\mathbf{H}^*)$. In such a conditional probability space, by the definition of $\in$-universal projective hashing, the distribution of $H_k(x)$ has min-entropy at least $\log_2(1/\in)$, and $\breve{k}$ is uniformly and independently distributed over $\breve{K}$. The Leftover Hash Lemma then directly implies that

$U(\mathbf{H}^* \mid x, s)$ and $V(\mathbf{H}^* \mid x, s)$ are $2^{-(b+1)}$-close. Since this bound holds uniformly for all x, s, it follows that $U(\mathbf{H}^*)$ and $V(\mathbf{H}^*)$ are also $2^{-(b+1)}$-close.

3.0 Subset membership problems

In this section we define a class of languages with some natural cryptographic indistinguishability properties. The definitions below capture the natural properties of well-known cryptographic problems such as the Quadratic Residuosity and Decision Diffie-Hellman problems, as well as others.

A *subset membership problem* $\mathbf{M}$ specifies a collection $(I_\lambda)_{\lambda \geq 0}$ of distributions. For every value of a security parameter $\lambda \geq 0$, I_λ is a probability distribution of instance descriptions. An instance description Λ specifies the following:

- Finite, non-empty sets X, L, and W, such that L is a proper subset of X.
- A binary relation $R \subset X \times W$.

For all $\lambda \geq 0$, $[I_\lambda]$ denotes the instance descriptions that are assigned non-zero probability in the distribution I_λ. We write $\Lambda[X, L, W, R]$ to indicate that the instance Λ specifies X, L, W and R as above.

For $x \in X$ and $w \in W$ with $(x, w) \in R$, we say that w is a witness for x. Note that it would be quite natural to require that for all $x \in X$, we have $(x, w) \in R$ for some $w \in W$ if and only

if $x \in L$, and that the relation R is efficiently computable; however, we will not make these requirements here, as they are not necessary for our purposes. The actual role of a witness will become apparent in the next section.

A subset membership problem also provides several algorithms. For this purpose, we require that instance descriptions, as well as elements of the sets X and W, can be uniquely encoded as bit strings of length polynomially bounded in λ. The following algorithms are provided:

- *a probabilistic*, polynomial time sampling algorithm that on input 1^λ for $\lambda \geq 0$ samples an instance Λ according to the distribution I_λ.

We do not require that the output distribution of the sampling algorithm and I_λ are equal; rather, we only require that they are $l(\lambda)$-close, where $l(\lambda)$ is a negligible function. In particular, with negligible probability, the sampling algorithm may output something that is not even an element of $[I_\lambda]$. We call this algorithm the instance sampling algorithm of $\mathbf{M}$, and we call the statistical distance $l(\lambda)$ discussed above its approximation error.

- *a probabilistic*, polynomial time sampling algorithm that takes as input 1^λ for $\lambda \geq 0$ and an instance $\Lambda[X, L, W, R] \in [I_\lambda]$, and outputs a random $x \in L$, together with a witness $w \in W$ for x.

We do not require that the distribution of the output value x and the uniform distribution on L are equal; rather, we only require that they are $l'(\lambda)$-close, where $l'(\lambda)$ is a negligible function. However, we do require that the output x is always in L.

We call this algorithm the *subset sampling algorithm* for $\mathbf{M}$, and we call the statistical distance $l'(\lambda)$ discussed above its approximation error.

- *a deterministic*, polynomial time algorithm that takes as input 1^{λ} for $\lambda \geq 0$, an instance $\Lambda[X, L, W, R] \in [I_{\lambda}]$, and $\varsigma \in \{0,1\}^{*}$, and checks whether ς is a valid binary encoding of an element of X.

This completes the definition of a subset membership problem.

We next define the notion of a *hard* subset membership problem. Essentially, this means that it is computationally hard to distinguish random elements of L from random elements of $X \setminus L$. We now formulate this notion more precisely.

Let $\mathbf{M}$ be a subset membership problem as above. We define two sequences of random variables, $(U_{\lambda}(\mathbf{M}))_{\lambda \geq 0}$ and $(V_{\lambda}(\mathbf{M}))_{\lambda \geq 0}$, as follows. Fix $\lambda \geq 0$, and consider the probability space defined by sampling $\Lambda[X, L, W, R]$ from I_{λ}, and choosing $x \in L$ at random and $x' \in X - L$ at random. Set $U_{\lambda}(\mathbf{M}) = (\Lambda, x)$ and $V_{\lambda}(\mathbf{M}) = (\Lambda, x')$.

***Definition* 3.1**

Let $\mathbf{M}$ be a subset membership problem. We say that $\mathbf{M}$ is hard if $(U_{\lambda}(\mathbf{M}))_{\lambda \geq 0}$ and $(V_{\lambda}(\mathbf{M}))_{\lambda \geq 0}$ are computationally indistinguishable.

4.0 Universal hash proof systems

4.1 Hash proof systems

Let $\mathbf{M}$ be a subset membership problem, as defined in the last section, specifying a sequence $(I_{\lambda})_{\lambda \geq 0}$ of instance distributions.

A *hash proof system (HPS)* $\mathbf{P}$ for $\mathbf{M}$ associates with each instance $\Lambda[X, L, W, R]$ of $\mathbf{M}$ a projective hash family $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$ for (X, L).

Additionally, $\mathbf{P}$ provides several algorithms to carry out basic operations we have defined for an associated projective hash family; namely, sampling $k \in K$ at random, computing $\alpha(k) \in S$ given $k \in K$, computing $H_k(x) \in \prod$ given $k \in K$ and $x \in X$. We call this latter algorithm the *private evaluation algorithm* for $\mathbf{P}$. Moreover, a crucial property is that the system provides an efficient algorithm to compute $H_k(x) \in \prod$, given $\alpha(k) \in S$, $x \in L$, and $w \in W$, where w is a witness for x. We call this algorithm the *public evaluation algorithm* for $\mathbf{P}$. The system should also provide an algorithm that recognizes elements of $\prod$.

We now discuss the above-mentioned algorithms in a bit more detail. In this discussion, whenever $\Lambda[X, L, W, R] \in [I_{\lambda}]$ is fixed in some context, it is to be understood that $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$ is the projective hash family that $\mathbf{P}$ associates with Λ. These algorithms work with bit strings of length bounded by a polynomial in λ to represent elements of K, $\prod$ and S. We also assume that these algorithms use the same encodings of the sets X, L and W as the algorithms from the subset membership problem $\mathbf{M}$. The system $\mathbf{P}$ provides the following algorithms:

- *a probabilistic, polynomial time* algorithm that takes as input 1^{λ} and an instance $\Lambda \in [I_{\lambda}]$, and outputs $k \in K$, distributed uniformly over K.

- *a deterministic, polynomial time* algorithm that takes as input 1^{λ}, an instance $\Lambda \in [I_{\lambda}]$, $k \in K$, and outputs $s \in S$ such that $\alpha(k) = s$.
- *a deterministic, polynomial time* algorithm that takes as input 1^{λ}, an instance $\Lambda \in [I_{\lambda}]$, $k \in K$ and $x \in X$, and outputs $\pi \in \prod$ such that $H_k(x) \in \pi$.

This is the private evaluation algorithm.

- *a deterministic, polynomial time* algorithm that takes as input 1^{λ}, an instance $\Lambda \in [I_{\lambda}]$, $s \in S$ such that $\alpha(k) = s$ for some $k \in S$, and $x \in L$ together with a witness $w \in W$ for x, and outputs $\pi \in \prod$ such that $H_k(x) \in \pi$. This is the public evaluation algorithm.
- *a deterministic, polynomial time* algorithm that takes as input 1^{λ}, an instance $\Lambda \in [I_{\lambda}]$, and $\varsigma \in \{0,1\}^*$, and determines if ς is a valid encoding of an element of $\prod$.

4.2 Universal hash proof systems

***Definition* 4.1**

Let $\in (\lambda)$ be a function mapping non-negative integers to non-negative reals. Let $\mathbf{M}$ be a subset membership problem specifying a sequence $(I_{\lambda})_{\lambda \geq 0}$ of instance distributions. Let $\mathbf{P}$ be an *HPS* for $\mathbf{M}$.

We say that $\mathbf{P}$ is $\in (\lambda)$-universal (respectively, -universal$_2$, -smooth) if there exists a negligible function $\delta(\lambda)$ such that for all $\lambda \geq 0$ and for all $\Lambda[X, L, W, R] \in [I_{\lambda}]$, the projective hash family $\mathbf{H} = (H, K, X, L, \prod, S, \alpha)$ that $\mathbf{P}$ associates with Λ is $\delta(\lambda)$-close to an $\in (\lambda)$-universal (respectively, -universal$_2$, -smooth) projective hash family $\mathbf{H}^* = (H^*, K^*, X, L, \prod, S, \alpha^*)$.

Moreover, if this is the case, and $\in (\lambda)$ is a negligible function, then we say that $\mathbf{P}$ is strongly universal (respectively, universal$_2$, smooth). We shall call the function $\delta(\lambda)$ in the above definition the *approximation error* of $\mathbf{P}$, and we shall refer to the projective hash family $\mathbf{H}^*$ as the *idealization* of $\mathbf{H}$.

It is perhaps worth remarking that if a hash proof system is strongly universal, and the underlying subset membership problem is hard, then the problem of evaluating $H_k(x)$ for random $k \in K$ and arbitrary $x \in X$, given only x and $\alpha(k)$, must be hard. We also need an extension of this notion. The definition of an *extended HPS* $\mathbf{P}$ for $\mathbf{M}$ is the same as that of ordinary *HPS* for $\mathbf{M}$, except that for each $k \geq 0$ and for each $\Lambda = \Lambda[X, L, W, R] \in [I_k]$, the proof system $\mathbf{P}$ associates with Λ a finite set E along with a projective hash family $\mathbf{H} = (H, K, X \times E, L \times E, \prod, S, \alpha)$ for ($X \times E$, $L \times E$).

Note that in this setting, to compute $H_k(x, e)$ for $x \in L$ and $e \in E$, the public evaluation algorithm takes as input $\alpha(k) \in S$, $x \in L$, $e \in E$, and a witness $w \in W$ for x, and the private evaluation algorithm takes as input $k \in K$, $x \in X$, and $e \in E$. We shall also require that elements of E are uniquely encoded as bit strings of length bounded by a polynomial in λ, and that $\mathbf{P}$ provides an algorithm that efficiently determines whether a bit string is a valid encoding of an element of E.

Definition 4.1 can be modified in the obvious way to define extended $\in (\lambda)$-universal$_2$ *HPS's*; although we do not need any of the other notions, nor are they particularly interesting for the purpose of this exposition.

5.0 Conclusion

We introduced the notion of a universal hash proof system. Essentially, this is a special kind of non-interactive zero-knowledge proof system for a language. We do not show that universal hash proof systems exist for all NP languages, but we do show how to construct very efficient universal hash proof systems for a general class of group-theoretic language membership problems.

Given an efficient universal hash proof system for a language with certain natural cryptographic indistinguishability properties, an efficient public-key encryption schemes secure against adaptive chosen ciphertext attack [11, 12, 13] can be constructed in the standard model. Our construction only uses the universal hash proof system as a primitive: no other primitives are required, although even more efficient encryption schemes can be obtained by using hash functions with appropriate collision-resistance properties.

It was also shown that the original Cramer-Shoup scheme [12, 14] follows from our general construction, when applied to a universal hash proof system related to the Decision Diffie-Hellman (DDH) assumption.

References

[1] Odule, T.J. "Incremental Cryptography and Security of Public Hash Functions." Journal of Nigerian Association of Mathematical Physics, vol. 11 pp.467-474; 2007.

[2] Y. Zheng, T. Matsumoto, and H. Imai, "Connections between several versions of one-way hash functions," Proc. SCIS90, The 1990 Symposium on Cryptography and Information Security, Nihondaira, Japan, Jan. 31–Feb.2, 1990.

[3] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," Proc. 21st ACM Symposium on the Theory of Computing, 1990, pp. 387–394.

[4] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Proc. STOC '90, ACM Press, 1990.

[5] Y. Zheng, T. Matsumoto, and H. Imai, "Structural properties of one-way hash functions," Advances in Cryptology, Proc. Crypto'90, LNCS 537, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 285–302.

[6] J. Carter and M. Wegman. Universal classes of hash functions. Journal of Computer and System Sciences, 18:143{154, 1979.

[7] M. Wegman and J. Carter. New hash functions and their use in authentication and set equality. Journal of Computer and System Sciences, 22:265{279, 1981.

[8] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In Proc. ACM Computer and Communication Security '93, ACM Press,1993.

[9] R. Canetti, O. Goldreich, and S. Halevi. The random oracle model, revisited. In Proc.STOC '98, ACM Press, 1998.

[10] M. Luby. Pseudorandomness and Cryptographic Applications. Princeton University Press, 1996.

[11] C. Racko_ and D. Simon. Non-interactive zero knowledge proof of knowledge and chosen-ciphertext attacks. In Proc. CRYPTO '91, Springer Verlag LNCS, 1991.

[12] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. SIAM Journal on Computing, 30:391{437, 2000. Extended abstract in Proc. STOC '91, ACM Press, 1991.

[13] P. Paillier. Public-key cryptosystems based on composite degree residue classes. In Proc. EUROCRYPT '99, Springer Verlag LNCS, 1999.

[14] R. Cramer and V. Shoup. A practical public key cryptosystem secure against adaptive chosen cipher text attacks. In Proc. CRYPTO '98, Springer Verlag LNCS, 1998.