Smooth Projective Hashing and Two-Message Oblivious Transfer

Shai Halevi
IBM Research

Yael Tauman Kalai*
Microsoft Research

October 31, 2010

Abstract

We present a general framework for constructing two-message oblivious transfer protocols using a modification of Cramer and Shoup's notion of smooth projective hashing (2002). This framework is an abstraction of the two-message oblivious transfer protocols of Naor and Pinkas (2001) and Aiello et al. (2001), whose security is based on the Decisional Diffie Hellman Assumption. In particular, we give two new oblivious transfer protocols. The security of one is based on the Quadratic Residuosity Assumption, and the security of the other is based on the N'th Residuosity Assumption. Compared to other applications of smooth projective hashing, in our context we must deal also with maliciously chosen parameters, which raises new technical difficulties.

We also improve on prior constructions of factoring-based smooth universal hashing, in that our constructions *do not require that the underlying RSA modulus is a product of safe primes.* (This holds for the schemes based on the Quadratic Residuosity Assumption as well as the ones based on the N'th Residuosity Assumption.) In fact, we observe that the safe-prime requirement is unnecessary for many prior constructions. In particular, the factoring-based CCA secure encryption schemes due to Cramer-Shoup, Gennaro-Lindell, and Camenisch-Shoup remain secure even if the underlying RSA modulus is not a product of safe primes.

1 Introduction

In [5], Cramer and Shoup introduced the first practical encryption scheme that was proved CCA secure in the standard model, with security based on the Decisional Diffie Hellman (DDH) Assumption. They later presented an abstraction of this scheme based on a notion that they called "smooth projective hashing" [6]. This abstraction yielded two new CCA secure encryption schemes; the security of one is based on the Quadratic Residuosity Assumption [12] and the security of the other is based on the N'th Residuosity Assumption [19].[1] The notion of smooth projective hashing was later used by Gennaro and Lindell [10] in the context of key generation from humanly memorizable passwords. That work abstracts and generalizes an earlier protocol for this problem due to Katz et al. [14], whose security is based on the DDH Assumption.

In this work we use smooth projective hashing to construct efficient two-message oblivious transfer protocols. Our work follows the same pattern, in that it abstracts and generalizes earlier

*Much of this work was carried out while visiting IBM T.J. Watson Research, New York. Also supported in part by NSF CyberTrust grant CNS-0430450.

[1]The N'th Residuosity Assumption is also referred to in the literature as the Decisional Composite Residuosity Assumption and as Paillier's Assumption.

protocols for this problem [17, 1] whose security is based on the DDH Assumption. Using smooth projective hashing in this context raises a new issue. Specifically, we must deal with the case that the hash family itself is chosen maliciously by the adversary. To this end, we add an extra requirement to the definition of smooth projective hashing. This issue did not arise in the previous two applications because these were either in the public key model or in the common reference string model, and in these models the parameters can be assumed to be generated honestly.

We show that even with this additional requirement, we can still construct smooth projective hashing from any of the following assumptions: the DDH Assumption, the N'th Residuosity Assumption, and the Quadratic Residuosity Assumption. Moreover, for the last two constructions we can prove security even when the underlying RSA modulus is not a product of safe primes. We note that all previous factoring-based constructions of smooth projective hashing did rely on the assumption that the underlying RSA modulus is a product of safe primes.

1.1 Oblivious Transfer

Oblivious transfer, first introduced by Rabin [20], is a central primitive in modern cryptography. It is a protocol between a *sender*, holding two strings γ_0 and γ_1, and a *receiver* holding a choice bit b. At the end of the protocol the receiver should learn the string of his choice (i.e., γ_b) but learn nothing about the other string. The sender, on the other hand, should learn nothing about the receiver's choice b. Oblivious transfer serves as the basis of a wide range of cryptographic tasks. Most notably, any secure multi-party computation can be based on a secure oblivious transfer protocol [21, 11, 15]. Oblivious transfer has been studied in several variants, all of which have been shown to be equivalent. The variant considered in this paper is the one by Even, Goldreich and Lempel [9] (a.k.a. 1-out-of-2 oblivious transfer), shown to be equivalent to Rabin's original definition by Crépeau [7].

Much work has been devoted to improving the efficiency of oblivious-transfer protocols, motivated by the fact that oblivious transfer is many times the main efficiency bottleneck with in protocols for secure multi-party computation. In particular, constructing round-efficient oblivious transfer protocols is an important task. Indeed, Naor and Pinkas [17, Protocol 4.1] and Aiello et al. [1] independently constructed a *two-message* (1-round) oblivious transfer protocol based on the DDH Assumption. Their work was the starting point of our work.

1.2 Smooth Projective Hashing

Smooth projective hashing was introduced by Cramer and Shoup [6]. Informally, a *projective hash family* is a family of keyed hash functions with two types of keys: the primary *hashing key* that can be used to compute the hash function on every point in its domain, and a *projective key* that can only be used to compute the hash function on a "special subset" of its domain. Moreover, to efficiently compute the hash value using the projective key one also needs a "witness" for membership in the special subset. (The domain is typically denoted by X and the special subset is typically denoted by L.) A projective hash family is *smooth* if the projective key gives (almost) no information about the value of the hash function on points outside the special subset. An important property that is used in all the applications of such families is that it is hard to distinguish members of the special subset from non-members. This is called the *hard subset membership* property.

1.3 Oblivious transfer from smooth projective hashing

We present a methodology for constructing a two-message oblivious transfer protocol from any (variant of a) smooth projective hash family. In particular, the protocols in [17, 1] can be viewed as a special case of this methodology. Moreover, we show that this methodology gives rise to two new oblivious transfer protocols; one based on the Quadratic Residuosity Assumption, and the other based on the N'th Residuosity Assumption. Similarly to the protocols from [17, 1], our protocols achieve a indistinguishability-based security notion, not a simulation-based notion, see Section 3 for further discussion of this point.

The basic idea. Given a smooth projective hash family with the hard subset membership property, consider the following two-message oblivious transfer protocol. Recall that the sender S takes as input a pair of strings γ_0, γ_1, and the receiver R takes as input a choice bit b.

$R \to S$: Generate the hashing parameters Λ (that define the domain X and the special subset L). Choose a random triple (x_0, x_1, w_b) where $x_b \overset{\$}{\leftarrow} L$, w_b is a "witness" for membership of $x_b \in L$, and $x_{1-b} \overset{\$}{\leftarrow} X \setminus L$. Send (Λ, x_0, x_1).

$S \to R$: Choose independently at random a primary hashing key k and a corresponding projective keys pk. Send pk along with $y_0 \leftarrow \gamma_0 \oplus H_k(x_0)$ and $y_1 \leftarrow \gamma_1 \oplus H_k(x_1)$.

R: Retrieve γ_b by computing $y_b \oplus H_k(x_b)$, using the witness w_b and the projective key pk_b.

The functionality of the protocol is implied by the projection property. The security of the receiver is implied by the hardness of the subset membership problem on X, since guessing the value of b implies distinguishing between a random member and a random non-member. The security of the sender (against an honest-but-curious receiver) is implied by the smoothness property of the hash family. Specifically, given a random projective key pk and any element in $x \in X \setminus L$, the value $H_k(x)$ is statistically indistinguishable from random. Thus, the message y_{1-b} gives no information about γ_{1-b} (since $x_{1-b} \in X \setminus L$).

Malicious receivers. The above protocol works for an honest-but-curious receiver, but the security of the sender is no longer guaranteed when considering malicious receivers. For example, a malicious receiver might choose $x_0, x_1 \in L$ and learn both γ_0 and γ_1. To overcome this problem, we extend the notion of smoothness so that it is possible to verify that the function is smooth on at least one of x_0, x_1. Note that this must hold even if the hashing parameters Λ are maliciously chosen by the receiver.

Implementing this extended notion in the context of the DDH assumption is straightforward [17, 1]. Loosely speaking, in this case the hashing parameters consist of a prime p and two elements $g_0, g_1 \in \mathbb{Z}_p^*$ of prime order $q|(p-1)$. The hashing domain is $X \overset{\text{def}}{=} \langle g_0 \rangle \times \langle g_1 \rangle = \{(g_0^{r_0}, g_1^{r_1}) : r_0, r_1 \in \mathbb{Z}_q\}$, the special subset is $L \overset{\text{def}}{=} \{(g_0^r, g_1^r) : r \in \mathbb{Z}_q\}$, and the witness is the exponent r. To enable the sender to verify that two elements x_0, x_1 are not both in L, we instruct the receiver to generate x_0, x_1 by choosing at random two distinct elements $r_0, r_1 \in \mathbb{Z}_q$, setting $x_b \leftarrow (g_0^{r_0}, g_1^{r_0})$, $w_b \leftarrow r_0$, and $x_{1-b} \leftarrow (g_0^{r_0}, g_1^{r_1})$. Notice that x_b is uniformly distributed in L, x_{1-b} is uniformly distributed in $X \setminus L$, and the sender can easily verify that at least one of x_0, x_1 is not in L by merely checking that they agree on their first coordinate and differ on their second coordinate.

1.4 Factoring-based smooth projective hashing

Implementing smooth projective hashing with the extra verifiability property in the context of the Quadratic Residuosity Assumption and the N'th Residuosity Assumption is not as easy. This part contains the bulk of technical difficulties of this work.

On top of providing the additional verifiability property, we were also able to somewhat relax the underlying assumptions that were used in prior work. The factoring-based constructions of smooth projective hashing, in the work of Cramer and Shoup (as well as all subsequent works), were only proved secure for the special case where the RSA modulus in use is a product of safe primes. Namely, they used $N \leftarrow pq$ where p, q are distinct odd primes such that $p' = (p-1)/2$ and $q' = (q-1)/2$ are also odd primes.[2] This restriction was explained by "technical reasons," but we observe that it is not needed: in fact our factoring-based constructions can be proved secure also for "generic" RSA moduli (i.e., for any $N \leftarrow pq$ where p, q are two odd primes of the same size).

Moreover, using the same tools we can eliminate the need for safe primes in the CCA encryption schemes that are based on smooth projective hashing (i.e., they too can be implemented without safe primes). Indeed, we observe that encryption schemes in the literature that are based on the N'th Residuosity or Quadratic Residuosity Assumptions (cf. [6, 10, 3]) remain secure even when the underlying RSA modulus is not chosen as a product of safe primes. In the appendix we explain how the proofs for the existing schemes can be modified to prove this stronger result, and exemplify it in detail for the proof of Camenisch and Shoup from [3].

Eliminating the safe primes. We describe the idea that allows us to eliminate the need for safe primes in the construction of smooth projective hashing based on the N'th Residuosity Assumption. (For the construction based on the Quadratic Residuosity Assumption, the idea is similar but even simpler.)

The only place where prior work really used the safe-prime condition is in proving the hard subset membership property. Namely, one needs to prove that it is hard to distinguish the special subset of the hashing scheme from the entire domain.

In the construction based on the N'th Residuosity Assumption, the hashing parameters are the RSA modulus N and a random N'th residue g modulo N^2, such that g is a quadratic non residue with Jacobi symbol $+1$. The hashing domain is all the elements in $Z^*_{N^2}$ with Jacobi symbol $+1$, denoted by J_{N^2}, and the special subset is the subgroup $\langle g \rangle$ that is generated by g. The safe-prime condition is used to argue that, with high probability, g generates the subgroup of all the N'th residues with Jacobi symbol $+1$. This is the case since the size of J_{N^2} is $\varphi(N^2)/2 = N\varphi(N)/2 = N(p-1)(q-1)/2 = 2Np'q'$. Thus, with high probability, a random quadratic non residue in the group J_{N^2} generates the entire group J_{N^2}, which implies that, with high probability, g generates the subgroup of all the N'th residues with Jacobi symbol $+1$. Thus, the N'th Residuosity Assumption (for elements with Jacobi symbol $+1$) immediately implies that it is hard to distinguish the special subset from the entire domain.

When N is just a plain old RSA modulus, we lose the property that with high probability g generates the subgroup of all the N'th residues with Jacobi symbol $+1$ (this subgroup is typically not cyclic). We thus need a slightly more involved reduction to prove the hard subset membership property. In Lemma 5 we show that the N'th Residuosity Assumption implies that it is hard to distinguish the subgroup $L \overset{\text{def}}{=} \langle g \rangle$ from the subgroup $X \overset{\text{def}}{=} \langle g \rangle \cdot \langle 1+N \rangle$ of all the elements that can

[2] p' and q' are also called Sophie Germain primes.

be generated as $x \leftarrow g^r(1+N)^s \bmod N^2$. We therefore get a smooth projective hashing scheme with domain X and special subset L, and we have the hard subset membership property.

One can see, however, that this construction is still missing one aspect that was present in prior notions. Namely, there could be elements of $Z^*_{N^2}$ that are not in the hashing domain X. Moreover, given the parameters N, g and an element $x \in Z^*_{N^2}$, there does not seem to be an easy way of deciding whether or not $x \in X$. This does not pose any problem for the application to oblivious transfer, but it potentially poses some problems in the case of chosen-ciphertext-secure encryption. Luckily, we show in the appendix that the proof of CCA security can be carried through even with this aspect missing.

2 Notations

For any positive integer X we denote the set $\{0, 1, \ldots, X-1\}$ by either $[X]$ or Z_X. We always denote the security parameter by n. A function $\nu : \mathbb{N} \to [0,1]$ is said to be negligible if for every polynomial $p(\cdot)$ and for every large enough n, $\nu(n) < 1/p(n)$. For an algorithm A, $y \leftarrow A(x)$ denotes running A on input x and assigning the result to y. If A is randomized then we denote $y \overset{\$}{\leftarrow} A(x)$ and y is a random variable. We also denote by $x \overset{\$}{\leftarrow} S$ the action of uniformly choosing an element from the set S.

For any two random variables X, Y, let $Dist(X,Y)$ denotes the statistical difference between X and Y, namely $Dist(X,Y) \triangleq \frac{1}{2}\sum_{s \in S} |\Pr[X = s] - \Pr[Y = s]|$, where S contains the support of both X and Y. We say that X and Y are ϵ-close, denoted $X \overset{\epsilon}{\approx} Y$, if $Dist(X,Y) \leq \epsilon$. We say that the ensembles $X = \{X_n\}_{n \in \mathbb{N}}$ and $Y = \{Y_n\}_{n \in \mathbb{N}}$ are *statistically indistinguishable*, denoted $X \overset{\mathrm{s}}{\equiv} Y$, if there exists a negligible function $\epsilon(\cdot)$ such that for every $n \in \mathbb{N}$, the random variables X_n and Y_n are $\epsilon(n)$-close.

We say that the ensembles $X = \{X_n\}_{n \in \mathbb{N}}$ and $Y = \{Y_n\}_{n \in \mathbb{N}}$ are *computationally indistinguishable*, denoted $X \overset{\mathrm{c}}{\equiv} Y$, if for every non-uniform probabilistic polynomial-time (PPT) distinguisher D there exists a negligible function $\epsilon(\cdot)$ such that for every $n \in \mathbb{N}$,

$$|\Pr[D(X_n) = 1] - \Pr[D(Y_n) = 1]| < \epsilon(n)$$

For simplicity, throughout this paper we say that two random variables X_n and Y_n are computationally (or statistically) indistinguishable, meaning that the corresponding distribution ensembles $\{X_n\}_{n \in \mathbb{N}}$ and $\{Y_n\}_{n \in \mathbb{N}}$ are computationally (or statistically) indistinguishable.

3 Security of Oblivious Transfer

Our definition of oblivious transfer is similar to the ones considered in previous works on oblivious transfer in the Bounded Storage Model [8, 2], and similar to the definition considered in [17] in the context of their DDH-based two-message oblivious transfer protocol. We remark that the definition below is specific for two-message protocols, as we only deal with such protocols in this work. In these protocols, the receiver R sends the first message, and the sender S sends the second message.

Definition 1 (Secure implementation of Oblivious Transfer) *A two-message, two-party protocol (S, R) is said to* securely implement oblivious transfer *for ℓ-bit strings (where $\ell : \mathbb{N} \to \mathbb{N}$) if it is a protocol in which both the sender and the receiver are probabilistic polynomial-time machines that* *get as input a security parameter n in unary representation. Moreover, the sender gets as input two strings $\gamma_0, \gamma_1 \in \{0,1\}^{\ell(n)}$, the receiver gets as input a choice bit $b \in \{0,1\}$, and the following conditions are satisfied:*

- Functionality: *If the sender and the receiver follow the protocol then for any security parameter n, any two input strings $\gamma_0, \gamma_1 \in \{0,1\}^{\ell(n)}$, and any bit b, the receiver outputs γ_b whereas the sender outputs nothing.*[3]

- Receiver's security: *Denote by $R(1^n, b)$ the message sent by the honest receiver with input $(1^n, b)$. Then the ensembles $\{R(1^n, 0)\}_{n\in\mathbb{N}}$ and $\{R(1^n, 1)\}_{n\in\mathbb{N}}$ are computationally indistinguishable; $\{R(1^n, 0)\}_{n\in\mathbb{N}} \overset{c}{\equiv} \{R(1^n, 1)\}_{n\in\mathbb{N}}$.*

- Sender's security: *Denote by $S(1^n, \gamma_0, \gamma_1, q)$ the response of the honest sender with input $(1^n, \gamma_0, \gamma_1)$ when the receiver's first message is q. Then there is a negligible function ν such that for any $n > 0$, any three messages $\gamma_0, \gamma_1, \gamma' \in \{0,1\}^{\ell(n)}$, and any message $q \in \{0,1\}^*$ (from a possibly cheating, not necessarily polynomial-time receiver), it holds that*

$$Dist(S(1^n, \gamma_0, \gamma_1, q),\ S(1^n, \gamma_0, \gamma', q)) \leq \nu(n) \quad or \quad Dist(S(1^n, \gamma_0, \gamma_1, q),\ S(1^n, \gamma', \gamma_1, q)) \leq \nu(n)$$

Note that similarly to the definitions in [8, 2, 17], Definition 1 is not a simulation-based definition but rather an indistinguishability-based one. Although it is a meaningful notion and is sufficient for some application, it is still weaker than the simulation-based notion: In particular it is cannot offer the same composability properties, and hence it is harder to use it as a building block in secure multi-party computation protocols.

The simulation-based definition compares the "real world," where the parties execute the protocol, to an "ideal world," where no message is exchanged between the two parties; rather, there is a trusted party that takes an input from both parties, computes the output of the Oblivious Transfer functionality on these inputs, and sends the corresponding output to each party. Loosely speaking, the simulation-based definition asserts that for every efficient adversary $\mathcal{A}$ (controlling either the sender or the receiver) in the "real world" there exists an efficient simulator $\mathcal{S}$, controlling the same party in the "ideal world," so that the outputs of the parties in the ideal world are computationally indistinguishable from their outputs in the real world. In particular, the simulator $\mathcal{S}$ needs to simulate the view of the adversary $\mathcal{A}$ in a computationally indistinguishable manner.

We note that Definition 1 does give a simulation-based guarantee in the case that the sender is corrupted (if you allow a resetting simulator). In this case, the simulator (who does not know the choice bit of the actual receiver) simulates the (honest) receiver first with choice bit $b = 0$, and then it resets the sender and simulates the honest receiver with choice bit $b = 1$. This way the simulator extracts both messages γ_0 and γ_1 from the corrupted sender $\mathcal{A}$. It then gives (γ_0, γ_1) to the trusted party. Then the simulator uses the view of the cheating sender $\mathcal{A}$ in the first execution (with the choice-bit $b = 0$). The fact that this view is indistinguishable from the "real world" view follows from the receiver's security which asserts that $\{R(1^n, 0)\}_{n\in\mathbb{N}} \overset{c}{\equiv} \{R(1^n, 1)\}_{n\in\mathbb{N}}$, and from the fact that sender does not receive any output from the trusted party.

On the other hand, Definition 1 does not give a simulation-based guarantee in the case that the receiver is corrupted. The reason is that a malicious receiver is not guaranteed to "know its own choice bit b", and therefore the simulator does not know which input bit to send to the trusted party

[3]This condition is also referred to as the completeness condition.

in order to obtain the desired output γ_b. However, Definition 1 does guarantee an exponential time simulation of the receiver's view of the interaction (similarly to the definition from [17]). Loosely speaking, the simulator can extract in exponential time a choice-bit b that is consistent with the receiver's first message q. Then the simulator gives this bit b to the trusted party, receives an output γ, and simulates the sender's message in the protocol by setting it to be $S(\gamma, 0^{\ell(n)}, q)$ if $b = 0$ and $S(0^{\ell(n)}, \gamma, q)$ if $b = 1$. This implies that the interaction gives no more information than what an *unbounded receiver* can derive from just the value of γ_b. (We note that if the receiver is "honest but curious" then the same simulation strategy can be implemented in polynomial time, and thus Definition 1 does guarantee simulation-based security for honest-but-curious adversaries.)

4 Smooth Projective Hash Functions

Our definition of smooth projective hashing differs in some ways from prior definitions [6, 10], mainly in that we add the requirement that it is possible to verify that at least one of two given elements is a non-member. We also depart from the presentation in previous work and define the notion of smooth projective hashing in terms of the procedures that are used to implement it rather than in terms of languages and sets. (This is merely a presentation issue, we believe that it makes the presentation clearer.) At the end of this section we briefly discuss the mapping between our presentation and the one used in previous work.

Syntax. A hash family $\mathcal{H}$ is defined by means of the following six polynomial-time algorithms, $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{IT}, \mathsf{HG}, \mathsf{Hash}, \mathsf{pHash})$:

- The *parameter-generator* PG is a randomized algorithm that takes as input the security parameter and outputs some parameters, $\Lambda \overset{\$}{\leftarrow} \mathsf{PG}(1^n)$. We assume that the security parameter can be inferred from Λ, and denote the security parameter corresponding to Λ by $n(\Lambda)$.

- The *instance-sampler* IS is a randomized algorithm that takes as input the parameters Λ and outputs a triple, $(w, x, x') \overset{\$}{\leftarrow} \mathsf{IS}(\Lambda)$. The intent is that x is a member of the special subset, x' is a non-member, and w is a witness for the membership of x in the special subset.

- The *instance-testing* algorithm IT tests the parameters Λ and two strings x_0, x_1, namely $\mathsf{IT}(\Lambda, x_0, x_1) \in \{0, 1\}$. The intent is to test that at least one of x_0, x_1 is not a member in the special subset.

- The *hash-key generator* HG is a randomized algorithm that takes as input the parameters Λ and outputs two keys (i.e., a primary hashing key and a projective key), $(k, pk) \overset{\$}{\leftarrow} \mathsf{HG}(\Lambda)$.

- The *primary hash algorithm* Hash takes the parameters Λ, a hash key k, and an element x, and outputs a string $y \leftarrow \mathsf{Hash}(\Lambda, k, x)$.

- The *secondary (projection) hash algorithm* pHash takes the parameters Λ, a projective key pk, and a pair (w, x), and outputs a string $y \leftarrow \mathsf{pHash}(\Lambda, pk, w, x)$.

For every string Λ, consider using Λ as the hashing parameters and let G_Λ denote the set of possible hash values with these parameters, namely

$$\begin{aligned} G_\Lambda &\stackrel{\text{def}}{=} \{\mathsf{Hash}(\Lambda, k, x) : (w, x, x') \in \mathsf{support}(\mathsf{IS}(\Lambda)),\ (k, pk) \in \mathsf{support}(\mathsf{HG}(\Lambda))\} \\ &\cup \ \{\mathsf{Hash}(\Lambda, k, x') : (w, x, x') \in \mathsf{support}(\mathsf{IS}(\Lambda)),\ (k, pk) \in \mathsf{support}(\mathsf{HG}(\Lambda))\} \end{aligned}$$

It helps to think of G_Λ as a group where we can efficiently compute the group operation and its inverse. In most of this paper, the group G_Λ will be the set of ℓ-bit strings (with the xor operation) where $\ell = \ell(n(\Lambda))$ for some polynomially-bound function $\ell(\cdot)$.

Definition 2 *Let $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{IT}, \mathsf{HG}, \mathsf{Hash}, \mathsf{pHash})$ and let Λ and x be two strings.*

Smoothness. *Let $\epsilon \geq 0$. We say that $\mathcal{H}$ is* ϵ-smooth *on (Λ, x) if the following two distributions are ϵ-close:*

$$\Big[(pk, \mathsf{Hash}(\Lambda, k, x))\Big]_{(k,pk) \xleftarrow{\$} \mathsf{HG}(\Lambda)} \quad \textit{and} \quad \Big[(pk, y)\Big]_{(k,pk) \xleftarrow{\$} \mathsf{HG}(\Lambda),\ y \xleftarrow{\$} G_\Lambda}$$

The first distribution is induced by choosing $(k, pk) \xleftarrow{\$} \mathsf{HG}(\Lambda)$ and outputting $(pk, \mathsf{Hash}(\Lambda, k, x))$, and the second is induced by choosing independently $y \xleftarrow{\$} G_\Lambda$ and $(k, pk) \xleftarrow{\$} \mathsf{HG}(\Lambda)$ and outputting (pk, y).

Projection. *We say that $\mathcal{H}$ is* projective *on (Λ, x) if for every pair of primary hashing keys k, k' for which there exist a projective key pk such that both (k, pk) and (k', pk) are in the support of $\mathsf{HG}(\Lambda)$, it holds that $\mathsf{Hash}(\Lambda, k, x) = \mathsf{Hash}(\Lambda, k', x)$.*

It is easy to verify that ϵ-smoothness on (Λ, x) and projection on (Λ, x) are contradictory requirements (assuming that $\epsilon < 1 - \frac{1}{|G_\Lambda|}$).

Definition 3 (Smooth Projective Hashing) *A family $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{IT}, \mathsf{HG}, \mathsf{Hash}, \mathsf{pHash})$ is a smooth projective hash family if there exists a negligible function $\epsilon : \mathbb{N} \to [0, 1]$ such that for every $\Lambda \in \mathsf{support}(\mathsf{PG})$, every $(w, x, x') \in \mathsf{support}(\mathsf{IS}(\Lambda))$, and every $(k, pk) \in \mathsf{support}(\mathsf{HG}(\Lambda))$, it holds that*

(a) $\mathsf{pHash}(\Lambda, pk, w, x) = \mathsf{Hash}(\Lambda, k, x)$.

(b) $\mathcal{H}$ is $\epsilon(n(\Lambda))$-smooth on (Λ, x').

(Clearly condition (a) above implies in particular that $\mathcal{H}$ is projective on (Λ, x), since the left-hand-side is independent of k except via the corresponding pk.)

Definition 4 (Verifiable Smoothness) *A smooth projective hash family $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{IT}, \mathsf{HG}, \mathsf{Hash}, \mathsf{pHash})$ is verifiably smooth if, in addition to the properties (a) and (b) in Definition 3, it holds that:*

(c) For every $\Lambda \in \mathsf{support}(\mathsf{PG})$ and every $(w, x, x') \in \mathsf{support}(\mathsf{IS}(\Lambda))$, it holds that $\mathsf{IT}(\Lambda, x, x') = \mathsf{IT}(\Lambda, x', x) = 1$.

(d) *For every* Λ, x_0, x_1 *such that* $\mathsf{IT}(\Lambda, x_0, x_1) = 1$, *it holds that either* $\mathcal{H}$ *is* $\epsilon(n(\Lambda))$*-smooth on* (Λ, x_0) *or it is* $\epsilon(n(\Lambda))$*-smooth on* (Λ, x_1) (*or both*).

Definition 5 (Hard Subset Membership) *A smooth projective hash family* $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{IT}, \mathsf{HG}, \mathsf{Hash}, \mathsf{pHash})$ *is said to have a* hard subset membership *property if the distribution ensembles* $\{A_n\}_{n\in\mathbb{N}}$, $\{B_n\}_{n\in\mathbb{N}}$ *defined below are computationally indistinguishable:*

Distribution A_n: *Choose at random* $\Lambda \xleftarrow{\$} \mathsf{PG}(1^n)$ *and* $(w, x, x') \xleftarrow{\$} \mathsf{IS}(\Lambda)$ *and output* (Λ, x, x').

Distribution B_n: *Choose at random* $\Lambda \xleftarrow{\$} \mathsf{PG}(1^n)$ *and* $(w, x, x') \xleftarrow{\$} \mathsf{IS}(\Lambda)$ *and output* (Λ, x', x).

Note that the difference between A_n and B_n is just in the order of x and x'. Note also that this condition is stronger than just requiring indistinguishability between members and non-members in the special subset (since x and x' may be chosen in a dependent manner). This stronger requirement is needed for the Oblivious Transfer application.

4.1 Comments

The definitions above are formulated in a way that is convenient for use in our application of Oblivious Transfer, but may make it harder to see the correspondence to the notions that were defined in previous work [6, 10]. We now briefly discuss this correspondence and provide some other clarifications.

Dealing with "bad inputs." We stress from the outset that many of the notations and definitions above do not depend on the inputs to the various algorithms being chosen "the right way." For example, the set G_Λ is well defined even when Λ is not in the support of the parameter-generation algorithm, and similarly the property of $\mathcal{H}$ being ϵ-smooth on (Λ, x) is well-defined for any two strings Λ and x.

In our application to Oblivious Transfer, we will use the hashing values only for instances that pass the instance-testing procedure, and will use that procedure to weed out nonsensical inputs. In particular, if we have some parameters Λ' that are malformed (in a recognizable way) we can have the instance-testing always rejecting them, and then the verifiable-smoothness requirement will be vacuous for such parameters.

Hash domain and the "special subset." Previous works presented the definitions in terms of some (parameter dependent) domain X_Λ for the hash function, and a "special subset" of that domain, which is an NP language $L_\Lambda \subset X_\Lambda$. They also required that it be possible to sample both members of L_Λ and non-members in $X_\Lambda \setminus L_\Lambda$. In our case, the special subset L_Λ is the support of the second element in the output of the instance-sampler, and the non-members $X_\Lambda \setminus L_\Lambda$ is the support of the third element, namely

$$\begin{aligned} L_\Lambda &= \{x : \exists\, w, x' \ s.t.\ (w, x, x') \in \mathsf{support}(\mathsf{IS}(\Lambda))\} \\ X_\Lambda \setminus L_\Lambda &= \{x' : \exists\, w, x \ s.t.\ (w, x, x') \in \mathsf{support}(\mathsf{IS}(\Lambda))\}. \end{aligned}$$

Since we require that the hash family is projective on members of the special subset and smooth on non-members, it follows that these two sets are indeed disjoint (assuming that $|G_\Lambda| > 1$).

The sampleable distributions on these sets are the ones induced by the instance-sampler. We comment that for our application we need to choose these elements in a dependent manner, since we need to sample pairs (x, x') that the instance-testing procedure accepts.

Projection and smoothness. The projection definition that we use is the usual one: for $x \in L_\Lambda$ (i.e., x that was output as the second element in the output of $\mathsf{IS}(\Lambda)$), the value of $\mathsf{Hash}(\Lambda, k, x)$ is determined by the corresponding projective key pk, and moreover it can be efficiently computed given a "witness" w using the algorithm $\mathsf{pHash}(\Lambda, pk, w, x)$. (This implies in particular that if there are a few "witnesses" for the same x, the value of $\mathsf{pHash}(\Lambda, pk, w, x)$ is the same for all of them.)

Our smoothness definition is the per-instance definition of Gennaro-Lindell rather than "random instance" definition of Cramer-Shoup. That is, we need $\mathcal{H}$ to be smooth *for every* non-member x rather than just for a random non-member.

Some redundancies. One can observe that the definitions above are somewhat redundant. For example, it is not hard to see that conditions (a), (c) and (d) of Definitions 3 and 4 together imply also condition (b) (assuming that $|G_\Lambda| > 1$). Also if $\mathcal{H}$ has the hard subset membership property then requiring $\mathsf{IT}(\Lambda, x, x') = 1$ in condition (c) of Definition 4 implies that also $\mathsf{IT}(\Lambda, x', x) = 1$ (except perhaps with a negligible probability).

5 Constructing 2-Message OT Protocols

We now show how to construct a two-message Oblivious Transfer protocol from smooth projective hash functions (defined in Section 4).

Let $\ell : \mathbb{N} \to \mathbb{N}$ be a (polynomially-bounded, efficiently computable) function, let $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{IT}, \mathsf{HG}, \mathsf{Hash}, \mathsf{pHash})$ be a verifiably-smooth projective hash family with the hard subset membership property (cf. Definitions 3-5), and assume for simplicity that for every setting of the parameters $\Lambda \in \{0,1\}^*$ it holds that $G_\Lambda = \{0,1\}^{\ell(n(\Lambda))}$. (At the end of this section we briefly discuss the (straightforward) modifications that are needed to deal with other domains.)[4]

Let n be the security parameter. Let (γ_0, γ_1) be the input of the sender, where γ_0 and γ_1 are $\ell(n)$-bit strings, and let $b \in \{0,1\}$ be the input of the receiver.

$R \to S$: The receiver chooses the hashing parameters $\Lambda \overset{\$}{\leftarrow} \mathsf{PG}(1^n)$, then samples random instances $(w, x, x') \overset{\$}{\leftarrow} \mathsf{IS}(\Lambda)$, sets $x_b \leftarrow x$ and $x_{1-b} \leftarrow x'$, and sends (Λ, x_0, x_1) to the sender.

$S \to R$: The sender invokes the testing algorithm $\mathsf{IT}(\Lambda, x_0, x_1)$ (to verify that the hashing is smooth on at least one of x_0, x_1). If the test fails then the sender aborts.

Otherwise the sender runs the hash-key generation algorithm twice independently to get $(k_0, pk_0) \overset{\$}{\leftarrow} \mathsf{HG}(\Lambda)$ and $(k_1, pk_1) \overset{\$}{\leftarrow} \mathsf{HG}(\Lambda)$, sets $y_0 \leftarrow \gamma_0 \oplus \mathsf{Hash}(\Lambda, k_0, x_0)$ and $y_1 \leftarrow \gamma_1 \oplus \mathsf{Hash}(\Lambda, k_1, x_1)$, and sends (pk_0, pk_1, y_0, y_1) to the receiver.

R: The receiver retrieves γ_b by computing $\gamma_b \leftarrow y_b \oplus \mathsf{pHash}(\Lambda, pk_b, w, x)$.

We next prove that the above protocol is secure according to Definition 1. The functionality follows from the fact that $\mathcal{H}$ is projective, which means that the value $\mathsf{Hash}(\Lambda, k, x_b)$ that the sender computes is equal to the value $\mathsf{pHash}(\Lambda, pk_b, w, x)$ that the receiver computes. The receiver's security follows from the hard subset membership property, which means that it is hard to distinguish between the pairs (x, x') and (x', x). The sender's security follows from verifiable smoothness,

[4]Recall that malformed Λ's can be handled using the instance-testing algorithm. See discussion in Section 4.1.

which means that for at least one of $b \in \{0,1\}$ the value of $\mathsf{Hash}(\Lambda, k_b, x_b)$ is (almost) random in $G_\Lambda = \{0,1\}^\ell$, even given the projective key pk_b.

Theorem 1 *The above 2-message OT protocol is secure according to Definition 1, assuming that $\mathcal{H}$ is a verifiably-smooth projective hash family that has the hard subset membership property.*

Proof The functionality trivially follows from $\mathcal{H}$ being projective. Similarly, the receiver's security trivially follows from $\mathcal{H}$ having the hard subset membership property, since $\{R(1^n,0)\}_{n\in\mathbb{N}} = \{A_n\}_{n\in\mathbb{N}}$ and $\{R(1^n,1)\}_{n\in\mathbb{N}} = \{B_n\}_{n\in\mathbb{N}}$, where $\{A_n\}_{n\in\mathbb{N}}$ and $\{B_n\}_{n\in\mathbb{N}}$ are from in Definition 5. Hence a probabilistic polynomial-time sender $\hat{S}$ that can predict with non-negligible advantage the choice bit b when interacting with $R(1^n, b)$ (on infinitely many auxiliary inputs $\{z_n\}_{n\in\mathbb{N}}$ with $|z_n| \le \mathsf{poly(n)}$) is by definition a distinguisher between $\{A_n\}_{n\in\mathbb{N}}$ and $\{B_n\}_{n\in\mathbb{N}}$.

It is left to prove the sender's security. Fix $n \in \mathbb{N}$ and $\gamma_0, \gamma_1, \gamma' \in \{0,1\}^{\ell(n)}$. Let $X = (\Lambda, x_0, x_1)$ be the first message sent by the receiver. If X is rejected by the testing algorithm, i.e. $\mathsf{IT}(\Lambda, x_0, x_1) = 0$, then the sender aborts regardless of its input (so the three random variables $S(1^n, \gamma_0, \gamma_1, X)$, $S(1^n, \gamma_0, \gamma', X)$, and $S(1^n, \gamma', \gamma_1, X)$ are identical). If $\mathsf{IT}(\Lambda, x_0, x_1) = 1$ then by verifiable smoothness we know that either $\mathcal{H}$ is ϵ-smooth on (Λ, x_0) or it is ϵ-smooth on (Λ, x_1), for some negligible ϵ. In the latter case we have

$$\begin{aligned}
S(1^n, \gamma_0, \gamma_1, X) &= (pk_0, pk_1, \gamma_0 \oplus \mathsf{Hash}(\Lambda, k_0, x_0),\ \gamma_1 \oplus \mathsf{Hash}(\Lambda, k_1, x_1)) \\
&\overset{\epsilon}{\approx} (pk_0, pk_1, \gamma_0 \oplus \mathsf{Hash}(\Lambda, k_0, x_0),\ \gamma_1 \oplus y)_{y \overset{\$}{\leftarrow} \{0,1\}^{\ell(n)}} \\
&= (pk_0, pk_1, \gamma_0 \oplus \mathsf{Hash}(\Lambda, k_0, x_0),\ \gamma' \oplus y)_{y \overset{\$}{\leftarrow} \{0,1\}^{\ell(n)}} \\
&\overset{\epsilon}{\approx} (pk_0, pk_1, \gamma_0 \oplus \mathsf{Hash}(\Lambda, k_0, x_0),\ \gamma' \oplus \mathsf{Hash}(\Lambda, k_1, x_1)) \\
&= S(1^n, \gamma_0, \gamma', X)
\end{aligned}$$

and in the former case we similarly get $S(1^n, \gamma_0, \gamma_1, X) \overset{2\epsilon}{\approx} S(1^n, \gamma', \gamma_1, X)$. This concludes the proof, since $\epsilon = \epsilon(n)$ is negligible in n. ■

Why two hashing keys? We note that in the protocol above the sender chooses two independent hashing key pairs (k_0, pk_0), (k_1, pk_1). One may wonder whether the proof would go through if the sender used only one hashing key pair. We claim that the answer is no, and that choosing two independent hash key pairs is essential, since otherwise the sender's security property of Definition 1 may be compromised. To see this, consider a modification where the sender uses only one key pair (k, pk), and consider a cheating receiver who sends a message $X = (\Lambda, x_0, x_1)$ where $\mathcal{H}$ is smooth on *both* (Λ, x_0) and (Λ, x_1). In this case, the definition of smooth projective hashing does not rule out the possibility that each of the values $\mathsf{Hash}(\Lambda, k, x_0)$, $\mathsf{Hash}(\Lambda, k, x_1)$ separately is enough to uniquely determines the hashing key k. In this case, the sender's message implies some relation between γ_0 and γ_1, violating Definition 1.

5.1 Working with different G_Λ

In the protocol above we assumed that for all Λ the group G_Λ is $\{0,1\}^{\ell(n(\Lambda))}$. Although it is always possible to use this setting (see discussion after Definition 6 in Section 6), one can sometimes gain efficiency by working with other groups. For example, for constructions based on DDH it is more natural to work with the underlying DDH group. The properties that we need from G_Λ for our construction to work are the following:

- G_Λ should be a quasi-group: Denoting the (quasi) group operation by '+', this means that for all $a, b \in G_\Lambda$ there exist unique $x, y \in G_\Lambda$ such that $a + x = y + a = b$.
- The group operation '+' and its inverse '−' can be computed efficiently. Namely, there should be a polynomial-time algorithm that on inputs Λ and $a, b \in G_\Lambda$ computes $c = a + b \in G_\Lambda$, and another one that on inputs Λ and $b, c \in G_\Lambda$ computes $a = c - b$ (i.e., an element $a \in G_\Lambda$ such that $a + b = c$).
- It should be possible to encode and decode the sender's inputs as elements in G_Λ. If the sender's inputs are ℓ-bit strings, there should be polynomial-time algorithms $\mathsf{Encode}, \mathsf{Decode}$ so that for all $s \in \{0,1\}^\ell$ it holds that $\mathsf{Encode}(\Lambda, s) \in G_\Lambda$ and $\mathsf{Decode}(\Lambda, \mathsf{Encode}(\Lambda, s)) = s$.

To work with such a (quasi) group, we modify the protocol so that instead of just computing $y_b \leftarrow \gamma_b \oplus \mathsf{Hash}(\Lambda, k_b, x_b)$, the sender encodes γ_b as an element $\Gamma_b \in G_\Lambda$ and then sends to the receiver $Y_b \leftarrow \Gamma_b + \mathsf{Hash}(\Lambda, k_b, x_b)$. Similarly, the receiver computes the inverse operation $\Gamma_b \leftarrow Y_b - \mathsf{pHash}(\Lambda, pk_b, w, x)$ and decode Γ_b to get γ_b.

The security of this protocol is proved similarly to Theorem 1. The arguments for functionality and receiver security are exactly as before. As for the sender security, we now replace $\gamma \oplus y$ for $y \overset{\$}{\leftarrow} \{0,1\}^{\ell(n)}$ by $\Gamma + Y$ for $y \overset{\$}{\leftarrow} G_\Lambda$. Since G_Λ is a quasi-group, this last distribution is the uniform distribution over G_Λ, regardless of what Γ is, and the proof follows.

6 Constructing Smooth Projective Hash Families

We next present two constructions of verifiably-smooth projective hash families with the hard subset membership property. In one construction the hard subset membership property is based on the Quadratic Residuosity Assumption, and in the other the hard subset membership property is based on the N'th Residuosity Assumption. A key vehicle in both constructions is the notion of a verifiably-ϵ-universal projective hash family.

Definition 6 (Universal Hashing) *Let $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{IT}, \mathsf{HG}, \mathsf{Hash}, \mathsf{pHash})$, let Λ and x be two strings, and let $\epsilon > 0$. We say that $\mathcal{H}$ is* ϵ-universal *on (Λ, x) if for all $y_0 \in G_\Lambda$ and all $pk_0 \in \{0,1\}^*$ it holds that*

$$\Pr_{(k,pk)}[pk = pk_0 \text{ and } \mathsf{Hash}(\Lambda, k, x) = y_0] \quad \le \quad \epsilon \cdot \Pr_{(k,pk)}[pk = pk_0]$$

where the probability is taken over a random choice $(k, pk) \overset{\$}{\leftarrow} \mathsf{HG}(\Lambda)$.

The definition of Verifiably-ϵ-universal Projective Hashing is similar to Definitions 3 and 4 of Verifiably-ϵ-smooth Projective Hashing. Specifically, conditions (a) and (c) below are identical to the ones in Definitions 3 and 4, and conditions (b′) and (d′) only differ in that we replaced ϵ-smooth by ϵ-universal (and ϵ need not be negligible).

Definition 7 (Verifiably-ϵ-universal Projective Hashing) *Let $\epsilon(\cdot)$ be a function. A family $\mathcal{H} = (\mathsf{PG}, \mathsf{IS}, \mathsf{IT}, \mathsf{HG}, \mathsf{Hash}, \mathsf{pHash})$ is an ϵ-universal projective hash family if for every Λ in the support of PG, every (w, x, x') in the support of $\mathsf{IS}(\Lambda)$, and every (k, pk) in the support of $\mathsf{HG}(\Lambda)$, it holds that*

(a) $\mathsf{pHash}(\Lambda, pk, w, x) = \mathsf{Hash}(\Lambda, k, x)$.

(b′) *$\mathcal{H}$ is $\epsilon(n(\Lambda))$-universal on (Λ, x').*

We say that $\mathcal{H}$ is verifiably-ϵ-universal projective hash family if in addition:

(c) *For every Λ in the support of PG and every (w, x, x') in the support of $\mathsf{IS}(\Lambda)$, it holds that $\mathsf{IT}(\Lambda, x, x') = \mathsf{IT}(\Lambda, x', x) = 1$.*

(d′) *For every Λ, x_0, x_1 such that $\mathsf{IT}(\Lambda, x_0, x_1) = 1$, it holds that either $\mathcal{H}$ is $\epsilon(n(\Lambda))$-universal on (Λ, x_0) or it is $\epsilon(n(\Lambda))$-universal on (Λ, x_1) (or both).*

Cramer and Shoup have shown in [6] how to transform an ϵ-universal projective hash family into a smooth projective hash family (for any $\epsilon < 1$), and the same transformation also works for transforming verifiably-ϵ-universal projective families into verifiably-smooth projective families. In a nutshell, one first reduces ϵ to ϵ^t by choosing t independent hashing key-pairs and hashing t times the same element. One then uses a "strong randomness extractor" [18] to extract a nearly uniform bit string from the t hash values. The new hash algorithms thus use t keys of the original algorithms and also the seed s for the extractor, setting

$$\begin{aligned}
\mathsf{Hash}'(\Lambda, (s, k_1, \ldots, k_t), x) &= \mathsf{Extract}(s; \mathsf{Hash}(\Lambda, k_1, x), \ldots, \mathsf{Hash}(\Lambda, k_t, x)) \\
\mathsf{pHash}'(\Lambda, (s, pk_1, \ldots, pk_t), w, x) &= \mathsf{Extract}(s; \mathsf{pHash}(\Lambda, pk_1, w, x), \ldots, \mathsf{pHash}(\Lambda, pk_t, w, x))
\end{aligned}$$

To extract ℓ bits, it is sufficient to choose t so that $t \cdot \log(1/\epsilon) \geq \ell + \omega(\log(n(\Lambda)))$ (see, e.g., Lemma 4 in the long version of [6]).[5] We comment that the resulting construction always has $G'_\Lambda = \{0,1\}^\ell$, regardless of the group G_Λ of the original construction. In the remainder of this paper we present two such constructions, one based on the Quadratic Residuosity Assumption and the other based on the N'th Residuosity Assumption. Both schemes are obtained by modifying the universal projective schemes of Cramer and Shoup to add the verifiable-universality property (and also to improve some parameters).

6.1 Some Algebraic Facts

We begin by recalling some algebraic facts that are used throughout the two constructions. For a finite algebraic group G (written multiplicatively), the order of G is the number of elements in it. The order of an element $x \in G$, denoted $\mathrm{ord}(x)$, is the smallest positive integer e such that $x^e = 1$, and it always divides the order of the group. Some other useful facts about the order of groups and elements are stated below.

1. For any $x \in G$, $\mathrm{ord}(1/x) = \mathrm{ord}(x)$. Also for any $x, y \in G$, $\mathrm{ord}(xy)$ divides $\mathrm{ord}(x) \cdot \mathrm{ord}(y)$.
2. For any integer m and element $x \in G$, $\mathrm{ord}(x^m) = \frac{\mathrm{ord}(x)}{GCD(m, \mathrm{ord}(x))}$. In particular, if $\mathrm{ord}(x)$ is co-prime with m then $\mathrm{ord}(x^m) = \mathrm{ord}(x)$, and if $\mathrm{ord}(x)$ *is not co-prime* with m then $\mathrm{ord}(x^m) \leq \mathrm{ord}(x)/2$.
3. If m is co-prime with the order of G, then the map $x \mapsto x^m$ is a permutation on G.

[5]Cramer and Shoup describe in [6] some optimizations to eliminate the extraction step in their factoring-based constructions, but these optimizations are not applicable when the hashing parameters are potentially malicious.

An easy corollary from the second fact above, is that if $\text{ord}(x) < 2^\ell$ then the order of x^{m^ℓ} must be co-prime with m. Another corollary of the facts above, which we use in Section 6.3, is the following:

Lemma 2 *Let $x, y \in G$ be two elements and let $m = \text{ord}(x/y)$. Then there exist two integers α, β, such that α divides $\text{ord}(x)$, β divides $\text{ord}(y)$, and $\alpha\beta = m$.*

Proof If we set $\alpha \leftarrow GCD(m, \text{ord}(x))$ then clearly α divides both m and $\text{ord}(x)$. It is left to show that $\beta \stackrel{\text{def}}{=} m/\alpha$ divides $\text{ord}(y)$. Denote $\rho \stackrel{\text{def}}{=} \text{ord}(x)/\alpha$ and observe that ρ and β are co-prime (since $\rho = \text{ord}(x)/GCD(m, \text{ord}(x))$ and $\beta = m/GCD(m, \text{ord}(x))$). Now $m = \text{ord}(x/y)$ must divide $\text{ord}(x)\cdot\text{ord}(1/y) = \text{ord}(x)\cdot\text{ord}(y)$. This means that also $\frac{m}{\alpha}$ divides $\frac{\text{ord}(x)\text{ord}(y)}{\alpha}$, namely $\beta \mid \rho\cdot\text{ord}(y)$. But since β and ρ are co-prime, then β must divide $\text{ord}(y)$. ■

6.2 A construction based on the Quadratic Residuosity Assumption

Let p, q be distinct odd primes, let $N \leftarrow pq$ and let QR_N denote the subgroup consisting of all squares of elements in the multiplicative group Z_N^*. Also, let J_N be the subgroup of Z_N^* consisting of all elements with Jacobi symbol $+1$. Then the order of Z_N^* is $\varphi(N)$, the order of J_N is $\varphi(N)/2$, and the order of QR_N is $\varphi(N)/4$. The Quadratic Residuosity Assumption asserts (informally) that given only N, it is hard to distinguish random elements in J_N from random elements in QR_N.

Assumption 8 (Quadratic Residuosity [12]) *The following ensembles are computationally indistinguishable*

$$\begin{aligned} & \{(N,x) \;:\; p,q \stackrel{\$}{\leftarrow} \mathsf{Primes}(n),\; N \leftarrow pq,\; x \stackrel{\$}{\leftarrow} J_N\}_{n\in\mathbb{N}} \\ \stackrel{\mathsf{c}}{\equiv}\; & \{(N,y) \;:\; p,q \stackrel{\$}{\leftarrow} \mathsf{Primes}(n),\; N \leftarrow pq,\; x \stackrel{\$}{\leftarrow} Z_N^*,\; y \leftarrow x^2 \bmod N\}_{n\in\mathbb{N}} \end{aligned}$$

where $\mathsf{Primes}(n)$ *denotes the set of prime numbers between 2^n and 2^{n+1} and J_N is the subgroup of Z_N^* of elements with Jacobi symbol +1.*

Blum integers. A Blum integer is a product of two primes, both of which are congruent to 3 modulo 4. It is not hard to see that if the Quadratic Residuosity Assumption holds, it must also hold when the moduli are chosen as Blum integers. This follows since (a) Blum integers are "dense" (i.e., when choosing $p, q \stackrel{\$}{\leftarrow} \mathsf{Primes}(n)$, there is a non-negligible probability to get $p \equiv q \equiv 3 \pmod 4$), and (b) Given a reliable oracle for Quadratic Residuosity, it is easy to check if a modulus $N = pq$ is a Blum integer (by checking that $N \equiv 1 \pmod 4$ and that $N - 1$ is *not* a quadratic residue modulo N). Below we will assume that determining quadratic residuosity is hard modulo Blum integers.

The Cramer-Shoup Scheme. Cramer and Shoup constructed in [6] an ϵ-universal projective hash family from the Quadratic Residuosity Assumption (in the special case where N is a product of two safe primes). Omitting some details, the hash parameters are the modulus $N \leftarrow pq$ and a quadratic residue $g \stackrel{\$}{\leftarrow} QR_N$, the hashing key is a random integer $k \stackrel{\$}{\leftarrow} \{1, 2, \ldots, N/2\}$, the projective key is $pk \leftarrow g^{2k} \bmod N$, the "special subset" is $L_{N,g} = \{g^{2w} : w < N/2\}$, the exponent w is a "witness" for $x \in L_{N,g}$, and the "non-members" are those elements $x \in J_N \setminus QR_N$ (i.e., the elements of even order). Given the witness w for the element x and the projective key pk, one can compute the hash value as

$$\mathsf{pHash}(N, g, pk, w, x) = pk^w = g^{2kw} = x^k \pmod{N}$$

Cramer and Shoup proved that when N is a product of two safe primes, this scheme is a (1/2)-universal projective family. Moreover, they also proved that when N is a product of two safe primes, distinguishing members from non-members can be reduced to the Quadratic Residuosity Assumption. These proofs make strong use of the fact that when N is a product of two safe primes, the subgroup of quadratic residues is cyclic and consists of exactly those elements in Z_N^* whose order is odd.

Our Modifications. In our case we must also consider a maliciously chosen modulus N that is not necessarily a product of two primes. To get verifiable smoothness (or verifiable universality) we need a way of checking that (a) the order of the element g is odd, and (b) the order of at least one of the two given elements x_0, x_1 is even. The latter test can be done by checking that $x_0 = N - x_1$ (which implies that $\mathrm{ord}(x_1/x_0) = 2$, and hence x_0, x_1 cannot both have odd order). For the former test, since we do not know how to test that g has odd order, we instead force it into the subgroup of odd-order elements by raising it to the power of $2^{\lceil \log N \rceil}$. Namely, instead of using the element g itself we use the element $g^{2^{\lceil \log N \rceil}}$. (We note that the trick of forcing g into the odd-order subgroup was used also in [13], as a way of eliminating the need for moduli of a special form.)

6.2.1 Detailed Construction

Using our notations from Section 4, we now describe the six algorithms that define the hash family $\mathcal{H}_{QR} = (\mathsf{PG}, \mathsf{IS}, \mathsf{IT}, \mathsf{HG}, \mathsf{Hash}, \mathsf{pHash})$.

Parameter-generator $\mathsf{PG}(1^n)$**.** Choose at random two n-bit prime numbers p, q, with $p < q < 2p - 1$ and $p \equiv q \equiv 3 \pmod{4}$. Set $N \leftarrow pq$, choose at random an element $g' \overset{\$}{\leftarrow} Z_N^*$, set $g \leftarrow (g')^2 \bmod N$, and output $\Lambda \leftarrow (N, g)$.

Instance-sampler $\mathsf{IS}(N, g)$**.** Choose at random $w \overset{\$}{\leftarrow} Z_N$, compute $T \leftarrow 2^{\lceil \log N \rceil}$, $x \leftarrow (g^T)^w \bmod N$ and $x' \leftarrow N - x$. Output (w, x, x').

Instance-testing algorithm $\mathsf{IT}(N, g, x, x')$**.** Check that $N > 2^{2n}$, $g, x \in Z_N^*$, and $x' = N - x$. Output '1' if all the tests pass and '0' otherwise.

Hash-key generator $\mathsf{HG}(N, g)$**.** Choose $k \overset{\$}{\leftarrow} Z_N$, set $T \leftarrow 2^{\lceil \log N \rceil}$ and $pk \leftarrow (g^T)^k \bmod N$. Output (k, pk).

Primary hashing algorithm $\mathsf{Hash}(N, g, k, x)$**.** Output $x^k \bmod N$.

Projective hash algorithm $\mathsf{pHash}(N, g, pk, w, x)$**.** Output $pk^w \bmod N$.

Remark. Since we only use the value g^T for hashing (never g itself), one can be tempted to drop the squaring in the Parameter-generator and simply choose at random $g \overset{\$}{\leftarrow} Z_N^*$. We note, however, that our proof of the hard subset membership property (Lemma 3) relies on this extra squaring operation.

6.2.2 Proof of Security

We now show that the construction above has a hard subset membership domain under the Quadratic Residuosity Assumption, and that it is a verifiable-ϵ-universal projective hash family with $\epsilon \approx \frac{1}{2}$.

Lemma 3 *Under the Quadratic Residuosity Assumption, the construction of $\mathcal{H}$ from Section 6.2.1 has the hard subset membership property.*

Proof We need to prove indistinguishability between the ensembles

$$A_n = \left\langle N, g, x, x' \right\rangle_n \text{ and } B_n = \left\langle N, g, x', x \right\rangle$$

where arithmetic is modulo N, and both ensembles are taken over choosing

$$p, q \overset{\$}{\leftarrow} \mathsf{Primes}(n) \text{ s.t. } p \equiv q \equiv 3 \pmod 4, \quad N \leftarrow pq, \quad T \leftarrow 2^{\lceil \log N \rceil}, \quad g' \overset{\$}{\leftarrow} Z_N^*,$$
$$g \leftarrow (g')^2, \quad w \overset{\$}{\leftarrow} Z_N, \quad x \leftarrow (g^T)^w, \quad x' \leftarrow N - x \pmod N$$

Recall that when N is a Blum integer, then the quadratic residues modulo N are exactly these elements that have odd order in Z_N^* (since the order of QR_N is $\varphi(N)/4$, which is odd). Moreover, every quadratic residue has exactly four square roots modulo N. If (r_1, r_2, r_3, r_4) are the four square roots of some $x \in QR_N$ then exactly one of them is itself a quadratic residue modulo N (call it r_1), and exactly two have Jacobi symbol $+1$ modulo N, namely r_1 and $N - r_1$.

Assume for the sake of contradiction that there exists a PPT algorithm $\mathcal{A}$ that distinguishes between the ensembles A_n and B_n with non-negligible probability. Let

$$p_1(n) \overset{\text{def}}{=} \Pr[\mathcal{A}(N, g, x, x') = 1], \quad p_2(n) \overset{\text{def}}{=} \Pr[\mathcal{A}(N, g, x', x) = 1], \quad \text{and } \epsilon(n) \overset{\text{def}}{=} |p_1(n) - p_2(n)|$$

We describe a PPT distinguisher $\mathcal{D}$ for Quadratic Residuosity modulo Blum integers with advantage (close to) $\epsilon(n)$. Since the Quadratic Residuosity Assumption implies in particular that it is hard to decide quadratic residuosity modulo Blum integers, then $\mathcal{D}$ violates the Quadratic Residuosity Assumption. The distinguisher $\mathcal{D}(N, z)$ in our reduction works as follows:

1. Choose $w \overset{\$}{\leftarrow} Z_N$ and set $T \leftarrow 2^{\lceil \log N \rceil}$.
2. Set $g \leftarrow z^2 \bmod N$, $x_0 \leftarrow z \cdot (g^T)^w \bmod N$ and $x_1 \leftarrow N - x_0$.
3. Run $\mathcal{A}$ to get $b \leftarrow \mathcal{A}(N, g, x_0, x_1)$, and output b.

Below we analyze the reduction under the simplifying assumption that the exponent w is chosen uniformly in $Z_{\varphi(N)}$ rather than in Z_N, both in the scheme and in the reduction. It is well known that this modification changes the various distributions by only $O(2^{-n})$ (where n is the bit-length of p, q).

Under this assumption, we show that when z is a quadratic residue the input to $\mathcal{A}$ is distributed according to A_n, and when z is a quadratic non-residue the input to $\mathcal{A}$ is distributed according to B_n. It thus follows that when z is a random element in QR_N then $\mathcal{D}$ outputs 1 with probability $p_1(n)$ and when z is a random element in J_N then $\mathcal{D}$ outputs 1 with probability $p_2(n)$, so the advantage of $\mathcal{D}$ is $p_1(n) - p_2(n) = \epsilon(n)$ (minus the negligible deviation caused by our simplifying assumption).

In both cases, the modulus N is chosen just like in the scheme $\mathcal{H}_{QR}$ and the element g is a random quadratic residue modulo N. It is left to show that (x_0, x_1) in the reduction are distributed like (x, x') in the scheme when z is a quadratic residue, and like (x', x) in the scheme when z is a quadratic non-residue.

Case 1: $z \in QR_N$**.** In this case we know that $\beta \overset{\text{def}}{=} \text{ord}(z)$ is an odd integer, which implies that the order of $g = z^2 \bmod N$ is the same as the order of z, namely $\text{ord}(g) = \text{ord}(z) = \beta$. Also, since β is odd then $\mu \overset{\text{def}}{=} 2^{-1} \bmod \beta$ and $\tau \overset{\text{def}}{=} T^{-1} \bmod \beta$ exist, and we have $z = g^{\mu} \pmod N$. Hence in this case we can write

$$x_0 = z \cdot (g^T)^w = g^{\mu + Tw} = g^{T(\tau\mu + w)} \pmod N$$

Now notice that since w is random in $Z_{\varphi(N)}$ and β divides $\varphi(N)$, the random variables $\tau\mu + w \bmod \beta$ and $w \bmod \beta$ are identically distributed, so $x_0 \leftarrow g^{T(w+\mu\tau)} \bmod N$ in the reduction is distributed identically to $x \leftarrow (g^T)^w \bmod N$ in the protocol.

Case 2: $z \in J_N \setminus QR_N$**.** In this case we know that $z' \overset{\text{def}}{=} N - z$ is a quadratic residue modulo N. By the same reasoning as above, we know that $\beta \overset{\text{def}}{=} \text{ord}(z')$ is an odd integer, which implies that the order of $g \leftarrow z^2 = (z')^2 \bmod N$ is also β, that $\mu \overset{\text{def}}{=} 2^{-1} \bmod \beta$ and $\tau \overset{\text{def}}{=} T^{-1} \bmod \beta$ exist, and that $z' \leftarrow g^{\mu} \bmod N$. Hence in this case we can write

$$x_1 = -(z \cdot (g^T)^w) = z' \cdot (g^T)^w = g^{\mu + Tw} = g^{T(\tau\mu + w)} \pmod N$$

and by the same arguments as above we have that $x_1 \leftarrow g^{T(w+\mu\tau)} \bmod N$ in the reduction is distributed identically to $x \leftarrow (g^T)^w \bmod N$ in the protocol. ■

Lemma 4 *The construction of $\mathcal{H}_{QR}$ from Section 6.2.1 is a verifiable-ϵ-universal projective hash family, where $\epsilon < \frac{1}{2} + O(2^{-n})$.*

Proof The projectiveness and the completeness of the instance-testing (conditions (a) and (c) in Definition 7) are easy to check: For any RSA modulus N and any $g \in QR_N$, $w \in Z_N$, and $k \in Z_N$, setting $x \leftarrow (g^T)^w \bmod N$ and $pk \leftarrow (g^T)^k \bmod N$ (where $T = 2^{\lceil \log N \rceil}$) we get for Condition (a):

$$\mathsf{pHash}(N, g, pk, w, x) = pk^w = (g^{Tk})^w = (g^{Tw})^k = x^k = \mathsf{Hash}(N, g, k, x) \pmod N$$

Also Condition (c) holds trivially.

The more interesting property is the verifiable-ϵ-universality (property (d')), which we prove next. (Note that properties (a), (c), and (d') together imply also property (b').) Let (N, g, x, x') be any four elements such that $\mathsf{IT}(N, g, x, x') = 1$. Namely, $N > 2^{2n}$, $g, x \in Z_N^*$, and $x' = N - x$. Note that since $x/x' = -1 \pmod N$ and $\text{ord}(-1) = 2$, then at least one of x, x' has an even order modulo N. We next show that for any element z of even order, $\mathcal{H}_{QR}$ is ϵ-universal on (N, g, z), which implies that it must be ϵ-universal on at least one of x, x'.

Fix an odd modulus $N > 2$ and some $g, z \in Z_N^*$ such that $e \overset{\text{def}}{=} \text{ord}(z)$ is even. Also denote $\tau \overset{\text{def}}{=} \text{ord}(g^T)$ (where $T = 2^{\lceil \log N \rceil}$) and observe that τ must be odd and must divide $\varphi(N)$. We again make the simplifying assumption that the hashing key k is chosen from $Z_{\varphi(N)}$ instead of from Z_N, thus introducing an error of $O(1/\sqrt{N}) = O(2^{-n})$ into the analysis. Under this assumption we show that for any pk and any y, it holds that

$$\Pr_{k \overset{\$}{\leftarrow} Z_{\varphi(N)}} [g^{Tk} = pk,\ y = z^k] \ \leq\ \frac{1}{2} \cdot \Pr_{k \overset{\$}{\leftarrow} Z_{\varphi(N)}} [g^{Tk} = pk]$$

We consider the following procedure for choosing $k \xleftarrow{\$} Z_{\varphi(N)}$: First choose $k_0 \xleftarrow{\$} \{0, \dots, \tau - 1\}$, then $k_1 \xleftarrow{\$} \{0, 1, \dots, \frac{\varphi(N)}{\tau} - 1\}$, and then set $k \leftarrow k_0 + \tau \cdot k_1$. Note that $pk = (g^T)^k$ depends only on the choice of k_0. It is therefore sufficient to show that for any k_0 (that determines pk) and any y, it holds that $\Pr_{k_1}[z^{k_0+\tau \cdot k_1} = y] \leq \frac{1}{2}$. In other words, let K_y be the set of all values that yield y,

$$K_y \stackrel{\text{def}}{=} \left\{ k_1 \in \left[0, \frac{\varphi(N)}{\tau} - 1\right] \ : \ z^{k_0+\tau \cdot k_1} = y \pmod{N} \right\},$$

and we need to show that K_y contains no more than $\frac{\varphi(N)}{2\tau}$ values. We observe that e (the order of z) does not divide τ (the order of g^T) since e is even and τ is odd. Therefore $z^\tau \neq 1 \pmod{N}$, which means that for any value of k_1, $z^{k_0+\tau \cdot k_1} \neq z^{k_0+\tau \cdot (k_1+1)} \pmod{N}$. Hence the set K_y cannot contain two consecutive integers, and since $\frac{\varphi(N)}{\tau}$ is even then K_y cannot contain more than half the values in $[0, \frac{\varphi(N)}{\tau} - 1]$. This concludes the proof. ■

6.3 A construction based on the N'th Residuosity Assumption

Let N be an odd positive integer and consider the multiplicative group $Z^*_{N^2}$ and the two subsets

$$R_N \stackrel{\text{def}}{=} \{x^N : x \in Z^*_{N^2}\} \quad \text{and} \quad G_N \stackrel{\text{def}}{=} \{1 + vN : v \in Z_N\}$$

R_N is the set of N'th residues, which is a subgroup of $Z^*_{N^2}$ of order $\varphi(N)$. G_N is also a subgroup of $Z^*_{N^2}$, which is homomorphic to the additive group Z_N (and thus has order N). Moreover, the order of an element $(1 + vN)$ in $Z^*_{N^2}$ equals the order of v in the additive group Z_N. The group $Z^*_{N^2}$ is homomorphic to the direct product of G_N and R_N, $Z^*_{N^2} \approx G_N \times R_N$. In particular, $Z^*_{N^2}$ has order $N \cdot \varphi(N)$. The N'th Residuosity Assumption, originally introduced by Paillier [19], asserts (informally) that for an RSA modulus N, it is hard to distinguish random elements of $Z^*_{N^2}$ from random elements of R_N.

Assumption 9 (N'th Residuosity [19]) *The following ensembles are computationally indistinguishable*

$$\begin{aligned} &\{(N, x) \ : \ p, q \xleftarrow{\$} \mathsf{Primes}(n), \ N \leftarrow pq, \ x \xleftarrow{\$} Z^*_{N^2}\}_{n \in \mathbb{N}} \\ \stackrel{\mathsf{c}}{\equiv} \ &\{(N, y) \ : \ p, q \xleftarrow{\$} \mathsf{Primes}(n), \ N \leftarrow pq, \ x \xleftarrow{\$} Z^*_{N^2}, \ y \leftarrow x^N \bmod N^2\}_{n \in \mathbb{N}} \end{aligned}$$

where $\mathsf{Primes}(n)$ *denotes the set of prime numbers between* 2^n *and* 2^{n+1}.

The Cramer-Shoup Scheme. Cramer and Shoup constructed in [6] an ϵ-universal projective hash family from the N'th Residuosity Assumption (in the special case where N is a product of two safe primes). This construction is very similar to the quadratic Residuosity construction, with the group of quadratic residues modulo N replaced with the group of N'th residues modulo N^2. Omitting some details again, the hash parameters Λ are the modulus $N \leftarrow pq$ and an N'th residue $g \in R_N$ (with Jacobi symbol $+1$), the hashing key is a random integer $k \xleftarrow{\$} \{1, 2, \dots, \lfloor N^2/2 \rfloor\}$, the projective key is $pk \leftarrow g^k \bmod N^2$, and the hash is computed as $\mathsf{Hash}(N, g, k, x) = x^k \bmod N^2$. The "special subset" is $L_{N,g} = \{g^w : w < N/2\}$, the exponent w is a "witness" for $x \in L_{N,g}$, and the "non-members" are those elements $x \in Z^*_{N^2}$ (with Jacobi symbol $+1$) whose order is divisible by p or q. Given the witness w for the element x and the projective key pk, one can compute the hash value as

$$\mathsf{pHash}(N, g, pk, w, x) = pk^w = g^{kw} = x^k \pmod{N^2}$$

Cramer and Shoup proved that when N is a product of two safe primes, this scheme is an ϵ-universal projective family with $\epsilon \approx \max\{1/p,\ 1/q\}$). Moreover, for that case they reduced distinguishing members from non-members to the N'th Residuosity Assumption. Similarly to the Quadratic Residuosity construction, here too the proofs make strong use of the fact that when N is a product of two safe primes, the N'th residues are exactly those elements of $Z^*_{N^2}$ whose order is co-prime to N, and moreover the subgroup of N'th residues with Jacobi symbol $+1$ is cyclic.

Our Modifications. In our case we must also consider a maliciously chosen modulus N that is not necessarily a product of two primes. To get verifiable smoothness (or verifiable universality) we need a way of checking that (a) the order of the element g is co-prime with N, and (b) the order of at least one of the two given elements x_0, x_1 is *not co-prime* with N. For the former, since we do not know how to test that g belongs to the subgroup of elements whose order is co-prime with N, we instead force it into that subgroup by raising it to the power of $N^{\lceil 2 \log N \rceil}$. Namely, instead of using the element g itself we use the element $g^{N^{\lceil 2 \log N \rceil}} \bmod N^2$. (Again, the use of this trick is similar to [13].)

The latter verification can in principle be done by checking that $x_1 \neq x_0$ and that $(x_1/x_0)^N = 1 \pmod{N^2}$. This implies that the order of x_1/x_0 in $Z^*_{N^2}$ divides N and is greater than one, and as $\mathrm{ord}(x_1/x_0)$ divides $\mathrm{ord}(x_1) \cdot \mathrm{ord}(1/x_0) = \mathrm{ord}(x_1) \cdot \mathrm{ord}(x_0)$, it means that $\mathrm{ord}(x_1)$ and $\mathrm{ord}(x_0)$ cannot both be co-prime with N. In our scheme, however, we use a slightly more elaborate test that yields better universality bound ϵ. Specifically, we test that x_1/x_0 is of the form $(1 + vN)$ where v and N are co-prime. With these modifications (which are introduced to get verifiability), we also observe that we no longer need the RSA modulus to be a product of safe primes, and also we do not restrict our attention to elements with Jacobi symbol $+1$.

6.3.1 Detailed Construction

Using our notations from Section 4, we now describe the six algorithms that define the hash family $\mathcal{H}_{NR} = (\mathsf{PG}, \mathsf{IS}, \mathsf{IT}, \mathsf{HG}, \mathsf{Hash}, \mathsf{pHash})$.

Parameter-generator $\mathsf{PG}(1^n)$**.** Choose two random n-bit prime numbers p, q (with $p < q < 2p$). Set $N \leftarrow pq$, choose an element $g' \xleftarrow{\$} Z^*_{N^2}$, set $g \leftarrow (g')^N \bmod N^2$, and output $\Lambda \leftarrow (N, g)$.

Instance-sampler $\mathsf{IS}(N, g)$**.** Choose $v, w \xleftarrow{\$} Z^*_N$, compute $T \leftarrow N^{\lceil 2 \log N \rceil}$, $x \leftarrow (g^T)^w \bmod N^2$ and $x' \leftarrow x \cdot (1 + vN) \bmod N^2$. Output (w, x, x').

Instance-testing algorithm $\mathsf{IT}(N, g, x, x')$**.** Check that $N > 2^{2n}$ and that $g, x \in Z^*_{N^2}$. Then set $d \leftarrow x'/x \bmod N^2$, verify that $(d-1)$ is divisible by N (over the integers) and set $v \leftarrow (d-1)/N$. Finally, verify that v and N are co-prime. Output '1' if all the tests pass and '0' otherwise.

Hash-key generator $\mathsf{HG}(N, g)$**.** Choose $k \xleftarrow{\$} Z_{N^2}$, set $T \leftarrow N^{\lceil 2 \log N \rceil}$ and $pk \leftarrow (g^T)^k \bmod N^2$. Output (k, pk). (Note that the order of g^T must be co-prime with N.)

Primary hashing algorithm $\mathsf{Hash}(N, g, k, x)$**.** Output $x^k \bmod N^2$.

Projective hash algorithm $\mathsf{pHash}(N, g, pk, w, x)$**.** Output $pk^w \bmod N^2$.

Remark. Since we only use the value g^T for hashing (never g itself), one can be tempted to drop the exponentiation in the Parameter-generator and simply choose at random $g \xleftarrow{\$} Z^*_{N^2}$. We note, however, that our proof of the hard subset membership property (Lemma 5) relies on this extra exponentiation.

6.3.2 Proof of Security

Lemma 5 *Under the N'th Residuosity Assumption, the construction of $\mathcal{H}_{NR}$ from Section 6.3.1 has the hard subset membership property.*

Proof We need to prove indistinguishability between the ensembles

$$A_n = \langle N, g, x, x' \rangle_n \text{ and } B_n = \langle N, g, x', x \rangle$$

where the arithmetic is modulo N^2, and both ensembles are taken over choosing

$$\begin{array}{l} p, q \xleftarrow{\$} \mathsf{Primes}(n), \quad N \leftarrow pq, \quad T \leftarrow N^{\lceil 2 \log N \rceil}, \quad g' \xleftarrow{\$} Z^*_{N^2}, \\ g \leftarrow (g')^N, \quad w, v \xleftarrow{\$} Z^*_N, \quad x \leftarrow (g^T)^w, \quad x' \leftarrow x(1 + vN) \pmod{N^2} \end{array}$$

Assume for the sake of contradiction that there exists a PPT algorithm $\mathcal{A}$ that distinguishes between the ensembles A_n and B_n with non-negligible probability. Let

$$p_1(n) \stackrel{\text{def}}{=} \Pr[\mathcal{A}(N, g, x, x') = 1], \quad p_2(n) \stackrel{\text{def}}{=} \Pr[\mathcal{A}(N, g, x', x) = 1], \quad \text{and } \epsilon(n) \stackrel{\text{def}}{=} |p_1(n) - p_2(n)|$$

Assume without loss of generality that there is an infinite set $S \subseteq \mathbb{N}$ such that for every $n \in S$ it holds that $p_1(n) \geq p_2(n)$, and moreover $\epsilon(n) = p_1(n) - p_2(n) \geq \frac{1}{\mathsf{poly(n)}}$. We now describe a PPT distinguisher $\mathcal{D}$ for the N'th Residuosity Assumption with advantage (close to) $\epsilon(n)/2$ for every $n \in S$, using $\mathcal{A}$ as a subroutine. $\mathcal{D}(N, z)$ works as follows:

1. Choose a random bit $b \xleftarrow{\$} \{0, 1\}$ and $v, w \xleftarrow{\$} Z^*_N$.
2. Set $g \leftarrow z^N \bmod N^2$, $x \leftarrow z^w \bmod N^2$, and $x' \leftarrow x(1 + vN) \bmod N^2$.
3. Set $x_b \leftarrow x$ and $x_{1-b} \leftarrow x'$, run $\mathcal{A}$ to get $b' \leftarrow \mathcal{A}(N, g, x_0, x_1)$, and output $b' \oplus b$.

We first claim that when the input z is chosen as a random N'th residue modulo N^2 then the distribution on (N, g, x, x') is nearly identical to the distribution in the actual scheme. To see this, first note that since N was chosen as $N \leftarrow pq$ where p, q are primes and $p < q < 2p - 1$, then N and $\varphi(N)$ are co-prime. Next, we make the simplifying assumption that the exponent w is chosen at random in $Z_{\varphi(N)}$ instead of from Z^*_N (in both the reduction and the instant-sampling algorithm of the scheme itself). It is well known that this modification changes the various distributions by only $O(2^{-n})$ (where n is the bit-length of p, q).

Since R_N has order $\varphi(N)$, which is co-prime with N, it follows that the maps $x \mapsto x^N$, $x \mapsto x^T$ and $x \mapsto x^{N^{-1} \bmod \varphi(N)}$ are all permutations on R_N. Hence choosing $g \xleftarrow{\$} R_N$ (as done in the scheme) induces the same distribution as choosing $z \xleftarrow{\$} R_N$ and setting $g \leftarrow z^N$ (as in the reduction). Moreover, choosing $w \xleftarrow{\$} Z_{\varphi(N)}$ and setting $x \leftarrow (g^T)^w$ (as in the scheme) induces the same distribution as choosing $w \xleftarrow{\$} Z_{\varphi(N)}$ and setting $x \leftarrow z^w = g^{N^{-1}w \pmod{\varphi(N)}}$ (as in the reduction).

Finally, computing x' from x is done in exactly the same way in the scheme and in the reduction. We thus get that for every $n \in S$,

$$\begin{aligned}
\Pr_{N,z \xleftarrow{\$} R_N}[\mathcal{D}(N,z)=1] &\geq \Pr[b=0]\cdot\Pr[\mathcal{A}(N,g,x,x')=1] + \Pr[b=1]\cdot\Pr[\mathcal{A}(N,g,x',x)=0] - O(2^{-n}) \\
&= \frac{1}{2}\cdot\big(\Pr[\mathcal{A}(N,g,x,x')=1] + (1-\Pr[\mathcal{A}(N,g,x',x)=1])\big) - O(2^{-n}) \\
&= \frac{1+\epsilon(n)}{2} - O(2^{-n})
\end{aligned}$$

We next show that when z is a random element in $Z_{N^2}^*$ then the input to $\mathcal{A}$ is independent of the bit b. We first observe that for any fixed $x \in Z_{N^2}^*$ and $w \in Z_N^*$, choosing at random $v \xleftarrow{\$} Z_N^*$ and setting $x' \leftarrow x(1+vN)$ induces the same distribution as setting $x' \leftarrow x(1+vwN)$ (since both v and vw are uniform in Z_N^*). In the rest of the analysis, we therefore consider this alternate setting of x' in Step 2 of the reduction. Namely, we consider the modified reduction where Step 2 is replaced by

2′. Set $g \leftarrow z^N \bmod N^2$, $x \leftarrow z^w \bmod N^2$, and $x' \leftarrow x(1+vwN) \bmod N^2$.

Note that with this alternate step, we have $x' = z^w(1+vN)^w = (z(1+vN))^w \pmod{N^2}$.

Next notice that for any fixed $v \in Z_N^*$ and a randomly chosen $z \xleftarrow{\$} Z_{N^2}^*$, the two distributions $(z, z(1+vN))$ and $(z(1+vN), z)$ are identical. This implies that also the two distributions $(z^w,\ (z(1+vN))^w)$ and $((z(1+vN))^w,\ z^w)$ are identical, for every fixed $v, w \in Z_N^*$.

Finally, since for all z, v is holds that $z^N = (z(1+vN))^N \pmod{N^2}$, the distribution of the random variable $(N, z^N, z^w, (z(1+vN))^w) = (N,g,x,x')$ is identical to the distribution of the random variable $(N, z^N, (z(1+vN))^w, z^w) = (N,g,x',x)$. Hence the input of $\mathcal{A}$ (and therefore also its output) is independent of the bit b that $\mathcal{D}$ chooses in Step 1, and thus $\Pr[\mathcal{D}(N,z)=1] = \frac{1}{2}$.

Combining the analysis for the two cases, we get that for every $n \in S$,

$$\Pr_{N,z \xleftarrow{\$} R_N}[D'(N,z)=1] - \Pr_{N,z \xleftarrow{\$} Z_{N^2}^*}[D'(N,z)=1] \geq \frac{1+\epsilon(n)}{2} - O(2^{-n}) - \frac{1}{2} = \frac{\epsilon(n)}{2} - O(2^{-n})$$

■

Lemma 6 *The construction of $\mathcal{H}_{NR}$ from Section 6.3.1 is a verifiable-ϵ-universal projective hash family where $\epsilon(n) \leq 2^{-n} + 2^{-2n}$.*

Proof The projection and the completeness of the scheme (conditions (a) and (c) in Definition 7) are easy to check: For any RSA modulus N and any $g \in R_N$, $w \in Z_N^*$, and $k \in Z_{N^2}$, setting $x \leftarrow (g^T)^w \bmod N^2$ and $pk \leftarrow (g^T)^k \bmod N^2$ (where $T = N^{\lceil 2\log N \rceil}$) we get for Condition (a):

$$\mathsf{pHash}(N,g,pk,w,x) = pk^w = (g^{Tk})^w = (g^{Tw})^k = x^k = \mathsf{Hash}(N,g,k,x) \pmod{N^2}$$

As for Condition (c), the instance-sampler sets $x' \leftarrow x(1+vN) \bmod N^2$ for some $v \in Z_N^*$. Condition (c) follows from the fact that $\frac{(x'/x \bmod N^2)-1}{N} = v$, $\frac{(x/x' \bmod N^2)-1}{N} = N-v$, and both v and $N-v$ are co-prime with N.

The more interesting property is the verifiable-ϵ-universality (property (d')), which we prove next. (Note that properties (a), (c), and (d') together imply also property (b').) Fix any (N,g,x,x')

such that $\mathsf{IT}(N, g, x, x') = 1$. Namely, $N > 2^{2n}$, $g, x \in Z^*_{N^2}$, and $x' = x(1+vN) \pmod{N^2}$ for some $v \in Z^*_N$ (which means that also $x' \in Z^*_{N^2}$). We show that either $\mathcal{H}_{NR}$ is ϵ-universal on (N, g, x) or it is ϵ-universal on (N, g, x').

Denote $d \stackrel{\text{def}}{=} (x'/x) = 1 + vN \pmod{N^2}$, and recall that the order of d in $Z^*_{N^2}$ is the same as the order of v in the additive group Z_N, which is exactly N (since v is co-prime with N). By Lemma 2, there exist two integers α, β such that $\alpha\beta = N$, α divides $\text{ord}(x)$ and β divides $\text{ord}(x')$. We now show that $\mathcal{H}$ is $(\frac{1}{\beta} + \frac{1}{N-1})$-universal on (N, g, x'), and a similar argument shows that it is also $(\frac{1}{\alpha} + \frac{1}{N-1})$-universal on (N, g, x). Observing that $N > 2^{2n}$ and thus either α or β must be larger than 2^n completes the proof.

Recall that the hashing key is chosen as $k \xleftarrow{\$} Z_{N^2}$, the projective key is then computed as $pk \leftarrow g^{Tk} \bmod N^2$ where $T = N^{\lceil 2 \log N \rceil}$, and the hash function is computed as $\mathsf{Hash}(N, g, k, x) = x^k \bmod N^2$. We make the simplifying assumption that the hashing key k is chosen from $Z_{\varphi(N^2)}$ instead of from Z_{N^2}, thus introducing an error of $O(1/\sqrt{N}) = O(2^{-n})$ into the analysis.

For the rest of the proof denote $\tau \stackrel{\text{def}}{=} \text{ord}(g^T)$ and observe that τ is co-prime with N and must divide $\varphi(N)$. Consider now the following procedure for choosing the hashing key k, that implies the same distribution as choosing $k \xleftarrow{\$} Z_{\varphi(N^2)}$: Choose $k_0 \xleftarrow{\$} \{0, \ldots, \tau - 1\}$, then $k_1 \xleftarrow{\$} \{0, 1, \ldots, \frac{\varphi(N^2)}{\tau} - 1\}$, and then set $k \leftarrow k_0 + \tau k_1$. Observe that the projective key pk depends only on the choice of k_0, since $g^{T(k_0 + \tau \cdot k_1)} = g^{Tk_0} \pmod{N^2}$. Below we prove, however, that the hash value on at least one of x, x' must depend also on the choice of k_1. Since k is chosen as $k = k_0 + \tau k_1$, we have

$$\mathsf{Hash}(N, g, k, x') = (x')^k = (x')^{k_0} \cdot (x')^{\tau k_1} \pmod{N^2}.$$

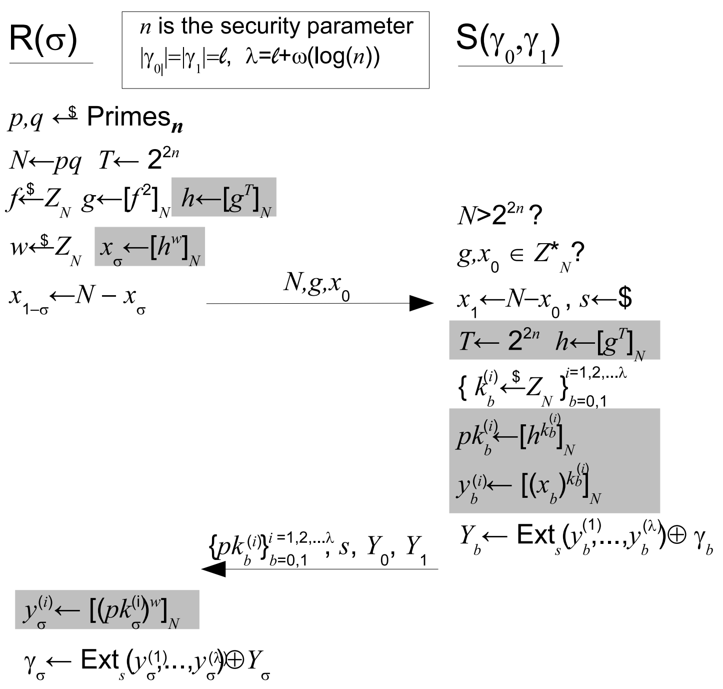
Also, since τ is co-prime with N, then it is also co-prime with β (as β divides N). Hence for two different values $k_1 \neq k_1' \pmod{\beta}$ we also have $\tau k_1 \neq \tau k_1' \pmod{\beta}$. As β also divides $\text{ord}(x')$, this means that also $\tau k_1 \neq \tau k_1' \pmod{\text{ord}(x')}$, and therefore $(x')^{\tau k_1} \neq (x')^{\tau k_1'}$.

Next notice that $\tau | \varphi(N)$ implies $\tau \leq \varphi(N)$ and so $\frac{\varphi(N^2)}{\tau} - 1 \geq N - 1 \geq \beta - 1$. It follows that when choosing $k_1 \xleftarrow{\$} \{0, 1, \ldots, \frac{\varphi(N^2)}{\tau} - 1\}$, the random variable $(k_1 \bmod \beta)$ can assume all the values between 0 and $\beta - 1$, so $(x')^{\tau \cdot k_1}$ can assume at least β different values modulo N^2. Moreover, each value of $(k_1 \bmod \beta)$ occurs either $\left\lfloor (\frac{\varphi(N^2)}{\tau} - 1)/\beta \right\rfloor$ or $\left\lceil (\frac{\varphi(N^2)}{\tau} - 1)/\beta \right\rceil$ times, and with $\frac{\varphi(N^2)}{\tau} - 1 \geq N - 1$ it means that no value has probability of more than $\frac{1}{\beta} + \frac{1}{N-1}$. Hence $\mathcal{H}$ is $(\frac{1}{\beta} + \frac{1}{N-1})$-universal on (N, g, x').

A symmetric argument shows that $\mathcal{H}$ is $(\frac{1}{\alpha} + \frac{1}{N-1})$-universal on (N, g, x). Since $\alpha\beta = N > 2^{2n}$ then at least one of α, β must be larger than 2^n, hence $\mathcal{H}$ is $(2^{-n} + 2^{-2n})$-universal on at least one of (N, g, x) or (N, g, x'). ■

7 Performance

We analyze the performance of our two oblivious transfer schemes: the one based on the Quadratic Residuosity Assumption from Section 6.2 (cf. Figure 1), and the one based on the N'th Residuosity Assumption from Section 6.3.



$R(\sigma)$
n is the security parameter
$|\gamma_0|=|\gamma_1|=\ell$, $\lambda=\ell+\omega(\log(n))$
$S(\gamma_0,\gamma_1)$
$p,q \xleftarrow{\$}$ Primes$_n$
$N\leftarrow pq$ $T\leftarrow 2^{2n}$
$f\xleftarrow{\$}Z_N$ $g\leftarrow[f^2]_N$ $h\leftarrow[g^T]_N$
$w\xleftarrow{\$}Z_N$ $x_\sigma\leftarrow[h^w]_N$
$x_{1-\sigma}\leftarrow N-x_\sigma$
N,g,x_0
$N>2^{2n}$?
$g,x_0\in Z^*_N$?
$x_1\leftarrow N-x_0$, $s\leftarrow\$$
$T\leftarrow 2^{2n}$ $h\leftarrow[g^T]_N$
$\{k_b^{(i)}\xleftarrow{\$}Z_N\}_{b=0,1}^{i=1,2,\ldots\lambda}$
$pk_b^{(i)}\leftarrow[h^{k_b^{(i)}}]_N$
$y_b^{(i)}\leftarrow[(x_b)^{k_b^{(i)}}]_N$
$Y_b\leftarrow \mathrm{Ext}_s(y_b^{(1)},\ldots,y_b^{(\lambda)})\oplus\gamma_b$
$\{pk_b^{(i)}\}_{b=0,1}^{i=1,2,\ldots\lambda}$, s, Y_0, Y_1
$y_\sigma^{(i)}\leftarrow[(pk_\sigma^{(i)})^w]_N$
$\gamma_\sigma\leftarrow \mathrm{Ext}_s(y_\sigma^{(1)},\ldots,y_\sigma^{(\lambda)})\oplus Y_\sigma$

Figure 1: Two-flow OT protocol using the QR-based smooth projective hashing. The modular exponentiations are highlighted.

7.1 The QR-based scheme

We start with the performance of the sender, whose input consists of two ℓ-bit input strings γ_0, γ_1 (and also the security parameter n). The sender gets (N, g, x_0) from the receiver and it first verifies that $2^{2n+2} > N > 2^{2n}$ and that g, x_0 are co-prime with N, and computes $x_1 \leftarrow N - x_0$ and $h \leftarrow g^{2^{2n}} \bmod N$. Denoting $\lambda = \ell + \omega(n)$, the sender then chooses 2λ random elements $k_b^{(i)} \xleftarrow{\$} Z_N$ ($b \in \{0, 1\}$, $i \in \{1, \ldots, \lambda\}$), and computes $pk_b^{(i)} \leftarrow h^{k_b^{(i)}} \bmod N$ and $y_b^{(i)} \leftarrow (x_b)^{k_b^{(i)}} \bmod N$. The sender also chooses a random extractor seed s, uses it to compute $Y_b \leftarrow \mathsf{Extract}(s; y_b^{(1)}, \ldots, y_b^{(\lambda)}) \oplus \gamma_b$, and sends back to the receiver all the $pk_b^{(i)}$'s and also s, Y_0 and Y_1. Hence the total work for the sender includes two GCD computations (to verify that $g, x \in Z_N^*$), $4\ell + \omega(\log n)$ modular exponentiations (to compute h and all the $pk_b^{(i)}$'s and $y_b^{(i)}$'s), and a few other faster operations (including one extraction operation).

The modular exponentiations can be sped up using multi-exponentiation techniques (see [16] for a survey).[6] For example, to compute the $pk_b^{(i)}$'s, we can first compute all the powers $h^1, h^2, h^4, \ldots, h^{2^{2n+1}}$, and then each $pk_b^{(i)}$ can be obtained as a subset product of these $2n + 2$ elements (with the subset-size being roughly n on the average). Pre-computing more powers of h can be used to speed-up the computation further: For a "window-size" parameter w, one can pre-compute $\approx 2^w \cdot 2n/w$ powers, and then each $pk_b^{(i)}$ can be computed as a multiple of at most $2n/w$ of these values. The

[6]Multi-exponentiation is typically used to speed up the computation of a product $\prod g_i^{e_i}$. But the same techniques can be used when computing all the individual terms, if the base of the exponent is the same in all of them.

same optimizations can be applied to the computation of the $y_b^{(i)}$'s. Using these techniques with window-size $w \approx \log \ell$, the complexity of the sender can be made as low as $O(\ell / \log \ell)$ modular exponentiations.

Considering next the receiver, it has the choice bit σ (and the security parameter n) as input. The receiver generates a $2n$-bit Blum integer N, then chooses $g \overset{\$}{\leftarrow} QR_N$, $w \overset{\$}{\leftarrow} Z_N$, computes $x \leftarrow g^{2^{2n} \cdot w} \bmod N$ and $x' \leftarrow N - x$, and sends to the sender N, g and one of x, x'. Upon receipt of the reply $(\{pk_b^{(i)}\}_{b=0,1}^{i=1,\ldots,\lambda}, s, Y_0, Y_1)$ from the sender, the receiver computes $y_\sigma^{(i)} \leftarrow (pk_\sigma^{(i)})^w \bmod N$ for $i = 1, \ldots, \lambda$ and then recovers $\gamma_\sigma \leftarrow \mathsf{Extract}(s; y_\sigma^{(1)}, \ldots, y_\sigma^{(\lambda)}) \oplus Y_\sigma$. The total work is therefore one key generation, $\ell + \omega(\log n)$ modular exponentiations, and a few other faster operations (including one extraction operation). We cannot use multi-exponentiation techniques for the receiver, since these exponentiations all have different bases. However, since the receiver knows the factorization of N then it can use Chinese-remaindering to speed up the exponentiation by a constant factor.

7.2 The NR-based scheme

Below we denote $\lambda \overset{\text{def}}{=} \left\lceil \frac{\ell + \omega(\log n)}{n} \right\rceil$. Since the constructions from Section 6.3 is 2^{-n}-universal, we need to repeat it λ times to get smoothness with output length of ℓ bits. We begin again with the sender. On input γ_0, γ_1 (and n), the sender gets from the receiver (N, g, x_0, x_1) and it first verifies that $2^{2n+2} > N > 2^{2n}$ and that g, x_0 are co-prime with N, then it computes $d \leftarrow x_1 / x_0 \pmod{N^2}$ and checks that $d-1$ is divisible by N and that $(d-1)/N$ is co-prime with N. If all the tests pass then the sender computes $h \leftarrow g^{N^{4n}} \bmod N^2$, then chooses 2λ random elements $k_b^{(i)} \overset{\$}{\leftarrow} Z_{N^2}$ ($b \in \{0,1\}$, $i \in \{1, \ldots, \lambda\}$), and computes $pk_b^{(i)} \leftarrow h^{k_b^{(i)}} \bmod N^2$ and also $y_b^{(i)} \leftarrow (x_b)^{k_b^{(i)}} \bmod N^2$. The sender also chooses a random extractor seed s, uses it to compute $Y_b \leftarrow \mathsf{Extract}(s; y_b^{(1)}, \ldots, y_b^{(\lambda)}) \oplus \gamma_b$, and sends back to the receiver all the $pk_b^{(i)}$'s and also s, Y_0 and Y_1.

The complexity of the sender's procedure is dominated by the computation of $h \leftarrow g^{N^{4n}}$ (roughly $2n$ full exponentiations), which cannot be sped-up with techniques of multi-exponentiation. (On the other hand, the computation of h is independent of the inputs $\sigma, \gamma_0, \gamma_1$, so in some applications it can perhaps be done off line.) For long input strings, $\ell > n^2$, the computation of the $pk_b^{(i)}$'s and $y_b^{(i)}$'s may also become significant (4λ full exponentiations), and this can be sped-up with multi-exponentiation techniques.

Considering the receiver, on input σ (and n), it generates a $2n$-bit RSA modulus N, then chooses $g \overset{\$}{\leftarrow} R_N$, $w, v \overset{\$}{\leftarrow} Z_N$, computes $h \leftarrow g^{N^{4n}}$, $x \leftarrow h^w$, and $x' \leftarrow (1 + vN)x$ (all modulo N^2), and sends to the sender N, g and the pair $\{x, x'\}$ (in order that depends on the choice bit σ). Upon receipt of the reply $(\{pk_b^{(i)}\}_{b=0,1}^{i=1,\ldots,\lambda}, s, Y_0, Y_1)$, the receiver computes $y_\sigma^{(i)} \leftarrow (pk_\sigma^{(i)})^w \bmod N^2$ for $i = 1, \ldots, \lambda$ and then recovers $\gamma_\sigma \leftarrow \mathsf{Extract}(s; y_\sigma^{(1)}, \ldots, y_\sigma^{(\lambda)}) \oplus Y_\sigma$. Since the receiver knows the factorization of N, the computation of $x \leftarrow g^{N^{4n} \cdot w}$ takes only two full exponentiations (one to evaluate $u \leftarrow N^{4n} w \bmod \phi(N^2)$ and the other to evaluate $x \leftarrow g^u \bmod N^2$). Hence the total work is one key generation and $\lambda + O(1)$ exponentiations.

A small optimization. We observe that we can slightly improve the sender's complexity by having the sender check that N is not divisible by any number smaller than n. Then both the sender and the receiver can use $h \leftarrow g^T$ for $T = N^{\lceil 4n/\log n \rceil}$ (instead of $T = N^{4n}$ as above), which would reduce the complexity of the sender roughly by a $\log n$ factor. The reason that this works is that we use the exponentiation to T only to ensure that the order of g^T is co-prime with N. If N does not have divisors smaller than n, then for every x whose order is not co-prime with N we have that $\mathrm{ord}(x^N) \leq \mathrm{ord}(x)/n$, and therefore taking $T = N^{\lceil \log_n N^2 \rceil}$, x^T must have order co-prime with N.

8 Conclusions

We presented in this work a general framework for constructing two-message oblivious transfer protocols using *verifiable* smooth projective function families. This is a modification of Cramer and Shoup's notion of smooth projective hashing: Compared to the original notion, we add the ability to generate pairs of inputs, so that one can verify that the function is not projective on both, without revealing which is projective and which is not. Using this framework, we gave two new oblivious transfer protocols; the security of one is based on the Quadratic Residuosity Assumption and the security of the other is based on the N'th Residuosity Assumption.

In contrast to previous work on factoring-based smooth projective hashing, we were able to prove security of our schemes without requiring the RSA modulus to be a product of safe primes. Moreover, we observe that the safe-prime requirement is unnecessary for several previous works; in particular, for the CCA secure encryption schemes of Cramer-Shoup [6] and of Camenisch-Shoup/Gennaro-Lindell [3, 10]. One open problem that remains is to check if the safe-prime requirement can be eliminated from other factoring-based smooth projective hashing works. For example, the password authentication schemes of Gennaro-Lindell [10] and of Canetti et al. [4].

Acknowledgments. We would like to greatly thank Alon Rosen, whose comments, suggestions and involvement played an essential part in the creation of this work. We also thank Rosario Gennaro for many discussions on smooth projective hashing.

References

[1] W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In *Advances in Cryptology - EUROCRYPT'01*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135. Springer, 2001.

[2] C. Cachin, C. Crépeau, and J. Marcil. Oblivious transfer with a memory-bounded receiver. In *FOCS'98*, pages 493–502. IEEE, 1998.

[3] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology – CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003. Long version available on-line from the Cryptology ePrint Archive, report 2002/161 (version 20030825:120805), `http://eprint.iacr.org/2002/161`.

[4] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. Universally composable password-based key exchange. In *Advances in Cryptology - EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421. Springer, 2005.

[5] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 1998.

[6] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology – EUROCRYPT'02*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2002. Long version available as ECCC TR01-072 (Revision 2), `http://www.eccc.uni-trier.de/report/2001/072/`.

[7] C. Crépeau. Equivalence between two flavours of oblivious transfers. In *Advances in Cryptology - CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 350–354. Springer, 1987.

[8] Y. Z. Ding, D. Harnik, A. Rosen, and R. Shaltiel. Constant-round oblivious transfer in the bounded storage model. In *Theory of Cryptography - TCC'04*, volume 2951 of *Lecture Notes in Computer Science*, pages 446–472. Springer, 2004.

[9] S. Even, O. Goldreich, and A. Lempel. A Randomized Protocol for Signing Contracts. *Communications of the ACM*, 28(6):637–647, June 1985.

[10] R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In *Advances in Cryptology – EUROCRYPT '03*, volume 2656 of *LNCS*, pages 524–543. Springer, 2003. Full version available on the ePrint archive, `http://eprint.iacr.org/2003/032/`.

[11] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing – STOC '87*, pages 218–229. ACM, 1987.

[12] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[13] S. Halevi. Efficient commitment schemes with bounded sender and unbounded receiver. *Journal of Cryptology*, 12(2):77–90, 1999.

[14] J. Katz, R. Ostrovsky, and M. Yung. Efficient and secure authenticated key exchange using weak passwords. *JACM*, 57(1):78–116, 2009.

[15] J. Kilian. Founding Cryptography on Oblivious Transfer. In *STOC'88*, pages 30–31. ACM, 1988.

[16] B. Möller. Algorithms for multi-exponentiation. In *SAC'01*, volume 2259 of *Lecture Notes in Computer Science*, pages 165–180. Springer-Verlag, 2001.

[17] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA'01*, pages 448–457. ACM, 2001.

[18] N. Nisan and D. Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[19] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT'99*, volume 1562 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 1999.

[20] M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard, 1981.

[21] A. C.-C. Yao. How to generate and exchange secrets. In *FOCS'86*, pages 162–167. IEEE, 1986.

A Factoring-Based CCA-Secure Encryption without Safe Primes

In this section we show that the factoring-based CCA secure encryption schemes from [6, 10, 3] — all of which were only proved secure when used with RSA moduli that are product of safe primes — remain secure even when used with "non-safe" RSA modulus. (We stress that we make no modification to the encryption schemes, only the proofs are modified somewhat.) These schemes are all very similar and all are proved in more-or-less the same way. Below we first give a very high level explanation of how to modify these security proofs to prove security for "non-safe" RSA moduli, and then exemplify this by providing detailed description for the proof from [3].

A.1 Modifying the Proofs: High Level Description

We begin with a very high-level description of the Cramer-Shoup construction of a CCA-secure encryption scheme from a smooth projective hash family. (This description is over-simplified and contains only the details that are important to convey our ideas.)

The encryption scheme. Very loosely speaking, the key generation algorithm first runs the parameter-generator algorithm PG to obtain parameters $\Lambda \overset{\$}{\leftarrow} \mathsf{PG}(1^n)$. Then it runs the hash-key generator algorithm HG to obtain a primary hashing key and a projective key $(k, pk) \overset{\$}{\leftarrow} \mathsf{HG}(\Lambda)$. It outputs (Λ, pk) as the public key, and k as the secret key. The parameters Λ define a hashing domain X and a "special subset" L. The encryption algorithm chooses an element x in the special subset L (together with a corresponding witness), and computes a tag $\hat{\pi}$ which is (essentially) the hash of x, computed using the projective key pk included in the public key. The ciphertext is a triple $(x, \hat{\pi}, e)$, where e is the element that actually carries information about the encrypted message (but we mostly ignore the element e in this high-level description). The decryption algorithm uses the corresponding primary hashing key k included in the secret key to re-compute the hash of x. It rejects the ciphertext if the computed hash differs from $\hat{\pi}$, and otherwise it extracts the message from e.

Recall the standard CCA-security game: The attacker gets the public key, and can interact with a decryption oracle. At some point the attacker sends two messages (called the *test messages*, and gets back the encryption of one of them (called the *target ciphertext*). The attacker can make more calls to the decryption oracle, as long as it does not query it on the target ciphertext, and its goal is to guess which of the two test messages is encrypted in the target ciphertext. The proof of CCA-security then uses the following arguments:

1. The answers to decryption queries, where the x component belongs to the special subset L, do not give any more information about the primary hashing key beyond what's implied

by the public key itself. (This is an information-theoretic argument that follows from the projectiveness of the hashing scheme.)

2. Decryption queries where x is not in the special subset L (i.e., where $x \in X \setminus L$), have incorrect tags w.h.p. (This is an information-theoretic argument that follows from the smoothness of the hashing scheme.)

3. The adversary cannot distinguish whether the x component of the challenge ciphertext is in the special subset L or is in $X \setminus L$ (because of the hard subset membership property).

4. If the x component of the target ciphertext is in $X \setminus L$, then the target ciphertext contains no information about the plaintext, assuming no additional information about the primary hashing key k is given beyond its corresponding projective key *pk*. (This is again an information-theoretic argument, similar to the argument about the smoothness of the hashing scheme.)

Roughly speaking, the last two arguments imply that barring some additional information from the decryption queries, the target ciphertext cannot be distinguished from an entity that carries no information about the plaintext. The first two arguments, on the other hand, imply that the attacker cannot use the decryption queries to get any information beyond what's already implied by the public key. Combining these arguments, CCA security follows.

The difficulty in removing the "safe prime requirement." Trying to use the constructions from Section 6 to get CCA security as above we run into some issues. Consider for example the construction of projective hashing based on the N'th Residuosity Assumption. In this case, the parameters are $\Lambda \leftarrow (N, g)$ where N is an RSA modulus and g is an N'th residue modulo N^2, the "special subset" is $L = \langle g \rangle$, and the witness for membership in L is the exponent r such that $g^r = x \bmod N^2$. When N is not a product of safe primes, the set of N'th residues in $Z^*_{N^2}$ is not a cyclic group (even when restricted to elements with Jacobi symbol $+1$), and in particular the "special subset" $L = \langle g \rangle$ does not contain all the N'th residues in $Z^*_{N^2}$. Since the hash function is only smooth on non-N'th-residues, we can no longer claim that it is smooth on any $x \notin L$.

In the Oblivious-Transfer application we "solved" this problem by defining $X = \langle g \rangle \cdot \langle 1 + N \rangle$, which ensures that every element in $X \setminus L$ is a non-N'th-residue. However, in a CCA attack the attacker may choose to supply ciphertext elements that are not in this set X, and there does not seem to be an easy way of deciding whether or not $x \in X$. In essence, our problem is that we cannot rule out the possibility that the attacker may find ciphertext elements which are neither in the smooth domain nor in the projective domain.

Our solution. We overcome this problem by considering four sets rather than two: Namely, we have the "big domain" $X = Z^*_{N^2}$, the "small domain" $X^* = \langle g \rangle \cdot \langle 1 + N \rangle$, the "projective subset" $L = R_N$ (of all the N'th residues modulo N^2), and the "special subset" $L^* = \langle g \rangle = X^* \cap L$. The encryption scheme itself would produce only elements that belong to the "special subset" L^*, but the attacker can submit decryption queries with arbitrary elements from the "big domain" X.

Then we use the facts that (a) the hashing scheme is projective on the "projective subset" L,[7] (b) the hashing scheme is smooth (or at least universal) on $X \setminus L$, and (c) the uniform distributions

[7]This statement is not precise; see remark below.

on L^* and X^* are indistinguishable (this is Lemma 7 below). The four arguments from above are refined as follows:

1′. The answers to decryption queries, where the x component belongs to the projective subset L, do not give any more information about the primary hashing key beyond what's implied by the public key itself.[7] (This is an information-theoretic argument that follows from the protectiveness of the hashing scheme.)

2′. Decryption queries where x is in $X \setminus L$ have incorrect tags w.h.p. (This is an information-theoretic argument that follows from the smoothness of the hashing scheme.)

3′. The adversary cannot distinguish whether the x component of the challenge ciphertext is in the special subset L^* or is in $X^* \setminus L^*$ (because of the hard subset membership property).

4′. If the x component of the target ciphertext is in $X^* \setminus L^*$ then the target ciphertext contains no information about the plaintext, assuming no additional information about the primary hashing key k is given beyond its corresponding projective key pk. (This is again an information-theoretic argument, similar to the argument about the smoothness of the hashing scheme.)

As before, arguments $3'$ and $4'$ imply that barring some additional information from the decryption queries, the target ciphertext cannot be distinguished from an entity that carries no information about the plaintext. Arguments $1'$ and $2'$, on the other hand, imply that the attacker cannot use the decryption queries to get information that will help him distinguish the target ciphertext from an entity that carries no information about the plaintext.

Remark. Argument $1'$ from above is slightly incorrect, in that the hashing schemes from Section 6 may fail to be projective on the "projective subset" L. Specifically, if the order of the element g (which is a random N'th residue) is not maximal (among the N'th residues), then the projective key g^k does not contain enough information on the hashing key k to fully determine the value of x^k for every N'th residue x.

An easy solution here is to consider a mental experiment in which the hashing scheme is augmented by including a maximal-order N'th residue h as a parameter and h^k as part of the projective key (so both h, h^k are included in the public key of the encryption scheme). These elements are never used by the encryption or decryption algorithms; their sole purpose is to leak to the adversary more information on the secret key, so as to make the Argument $1'$ from above correct. The arguments above are then applied to prove that this modified scheme is CCA secure, which implies that also the original scheme is CCA secure.

A.2 Reconstructing the Camenisch-Shoup Proof

We next reconstruct the proof from the long version of [3], augmenting it to show that the N'th-Residuosity-based encryption scheme of Gennaro-Lindell/Camenisch-Shoup [10, 3] is CCA secure even when the modulus is not a product of safe primes.[8] This modification is based on the same

[8]We chose to reconstruct the proof from [3] since the presentation of that proof is more "from first principles" than that of the proof from [10].

approach as in Section A.1, but is explained in greater detail. The encryption scheme consists of the following three algorithms.

Key generation. On security parameter n, choose a seed s for a collision-resistant hash function H_s, choose $p, q \overset{\$}{\leftarrow} \mathsf{Primes}(n)$, and set $N \leftarrow pq$. Next choose at random $g' \overset{\$}{\leftarrow} Z^*_{N^2}$ and three integers $x_1, x_2, x_3 \overset{\$}{\leftarrow} [N^2/4]$, and set $g \leftarrow (g')^{2N}$, $y_1 \leftarrow g^{x_1}$, $y_2 \leftarrow g^{x_2}$, and $y_3 \leftarrow g^{x_3}$ (all modulo N^2). Output (s, N, g, y_1, y_2, y_3) as the public key, and (x_1, x_2, x_3) as the secret key.

Encryption. To encrypt $m \in [N]$ (with label $L \in \{0,1\}^*$) choose at random $r \overset{\$}{\leftarrow} [N/4]$ and set

$$e \leftarrow y_1^r(1+N)^m, \quad u \leftarrow g^r, \quad \alpha \leftarrow H_s(u, e, L), \quad v \leftarrow \mathsf{abs}\left((y_2 y_3^\alpha)^r\right),$$

where all the calculation are modulo N^2, and $\mathsf{abs}(x) \overset{\text{def}}{=} \min\{x, N^2 - x\}$. The ciphertext is (e, u, v).

Decryption. To decrypt the ciphertext (e, u, v) with label L, set $\alpha \leftarrow H_s(u, e, L)$, and check that $v = \mathsf{abs}(v)$ and $u^{2(x_2+\alpha x_3)} = v^2 \pmod{N^2}$. If so, let $z \leftarrow (e/u^{x_1})^{N+1} \pmod{N^2}$, and if $z-1$ is divisible by N (over the integers) then compute $m \leftarrow \frac{z-1}{N}$ and output m.

Remark. Some aspects of this construction are not really relevant for our discussion: These include the squaring of g' during key-setup, the use of the "absolute values", the squaring of u and v and exponentiation to the $N+1$ power during decryption, and also the inclusion of the attached labels. The reason that we keep these aspects here is because they are present in the scheme and proof from [3], and we want to stress that we are proving the exact same scheme (using an almost identical proof), except that we omit the requirement of using safe primes. Clearly, the same proof as below can be applied also to the simplified scheme without these components.

A.2.1 Proof of security

To prove security for moduli that are not product of safe primes, we use the following lemma.

Lemma 7 *Under the N'th Residuosity Assumption, given a random RSA modulus N and a random N'th residue $g^N \in R_N$ the uniform distribution on $\langle g^N \rangle \subseteq R_N$ is indistinguishable from the uniform distribution on $\langle g^N \rangle \cdot \langle 1+N \rangle \subseteq Z^*_{N^2}$. More precisely, the following two ensembles are computationally indistinguishable:*

$$\left\langle N, g^N, g^{Nr} \right\rangle \overset{c}{\equiv} \left\langle N, g^N, g^{Nr}(1+N)^s \right\rangle$$

where the arithmetic is modulo N^2 and both ensembles are taken over choosing

$$p, q \overset{\$}{\leftarrow} \mathsf{Primes}(n), \quad N \leftarrow pq, \quad g \overset{\$}{\leftarrow} Z^*_{N^2}, \quad r \overset{\$}{\leftarrow} Z_{\varphi(N)}, \; s \overset{\$}{\leftarrow} Z_N$$

Proof Assume that we have an algorithm $\mathcal{A}$ that can distinguish the two distributions above with advantage ϵ, and we show a distinguisher $\mathcal{D}$ for N'th Residuosity with advantage at least $\epsilon - O(2^{-n})$. On input (N, z), $\mathcal{D}$ chooses $t \overset{\$}{\leftarrow} [N]$, runs $\mathcal{A}$ on the tuple (N, z^N, z^t) and outputs whatever $\mathcal{A}$ does.

Since N is co-prime with $\varphi(N)$, it follows that if z is a random N'th residue then so is z^N, and moreover $\langle z \rangle = \langle z^N \rangle$. Also, since $(t \mod \varphi(N))$ is statistically close upto 2^{-n} to the uniform distribution over $[\varphi(N)]$ then z^t is statistically close to the uniform distribution over $\langle z \rangle = \langle z^N \rangle$.

We now show that for a random $z \xleftarrow{\$} Z^*_{N^2}$, the distribution (z^N, z^t) is statistically close to $(x, x^r(1+N)^s)$ for a random $x \xleftarrow{\$} R_N$ and $r \xleftarrow{\$} [\varphi(N)]$, $s \xleftarrow{\$} [N]$. Recall that $Z^*_{N_2} \approx G_N \times R_N$, so for every $z \in Z^*_{N^2}$ there is a unique representation $z = z_1 \cdot z_2 \pmod{N^2}$ with $z_1 \in G_N$ and $z_2 \in R_N$. Hence

$$(z^N, z^t) \;=\; ((z_1z_2)^N, (z_1z_2)^t) \;=\; (z_2^N, z_2^t \cdot z_1^t)$$

Moreover for a uniform $z \xleftarrow{\$} Z^*_{N^2}$ we have that z_1, z_2 are independent and uniform in G_N, R_N, respectively. The same argument from above says that the distribution (z_2^N, z_2^t) is statistically close upto 2^{-n} to (x, x^r) for $x \xleftarrow{\$} R_N$ and $r \xleftarrow{\$} [\varphi(N)]$. Finally, recall that t is co-prime with N except with probability $O(2^{-n})$, and z_1 is uniform in G_N and independent from z_2, t. It hence follows that the statistical distance between the two distributions

$$\left\{(z^N, z^t) : z \xleftarrow{\$} Z^*_{N^2}, t \xleftarrow{\$} [N]\right\} \text{ and } \left\{(x, x^r \cdot y) : x \xleftarrow{\$} R_N, r \xleftarrow{\$} [\varphi(N)], y \xleftarrow{\$} G_N\right\}$$

is bounded by $O(2^{-n})$. ■

Lemma 8 *Under the N'th Residuosity Assumption, the encryption scheme from Section A.2 is CCA secure.*

Proof The proof mimics closely the proof of Theorem 1 from the long version of [3]: We describe almost the exact same nine games that were considered in the proof of Theorem 1 there, in fact the only steps in which our proof differs from the one in [3] are Games 4-5 (see below).

Game 0. This is the standard CCA-game. Namely, the key-generation algorithm is run on security parameter n, resulting in public key (s, N, g, y_1, y_2, y_3) and secret key (x_1, x_2, x_3) as above. The attacker gets the public key as input, and it may issue decryption queries (e_i, u_i, v_i, L_i), $i = 1, 2, \ldots$ For each query, the attacker gets the result of running the decryption algorithm on these ciphertexts and labels. (This is called "Probing phase I" in [3]).

Next the attacker outputs two messages $m_0, m_1 \in [N]$ and label L^*. Then a random bit $\sigma \xleftarrow{\$} \{0,1\}$ is chosen and the message m_σ is encrypted (with label L^*). This is done by choosing $r^* \xleftarrow{\$} [N^2/4]$ and setting

$$u^* \leftarrow g^{r^*}, \quad e^* \leftarrow y_1^{r^*}(1+N)^{m_\sigma}, \quad \alpha^* \leftarrow H_s(e^*, u^*, L^*), \text{ and } v^* \leftarrow \mathsf{abs}\left((y_2 y_3^{\alpha^*})^{r^*}\right).$$

Below we call m_0, m_1 the "target messages" and (e^*, u^*, v^*) the "target ciphertext". The target ciphertext is returned to the attacker, and then the attacker can keep making decryption queries as before, under the condition that $(e_i, u_i, v_i, L_i) \neq (e^*, u^*, v^*, L^*)$. (This is called "Probing phase II".) Finally the attacker outputs a bit σ', and it is considered successful if and only if $\sigma' = \sigma$.

The goal of the analysis is to prove that under the N'th Residuosity Assumption, the attacker cannot succeed with probability noticeably better than 1/2. The analysis proceeds by making successive small changes to the way some variables are computed in this game, each time proving that the change can have at most a negligible effect on the success probability of the attacker, until arriving at a game where the attacker's view is independent of the bit σ (and therefore its success probability is exactly 1/2).

Game 1. The only difference between this game and the previous one is that the decryption oracle rejects any ciphertext query (e_i, u_i, v_i, L_i) during "Probing phase II" such that $(e_i, u_i, L_i) \neq (e^*, u^*, L^*)$ but $H_s(e_i, u_i, L_i) = H_s(e^*, u^*, L^*)$. Clearly this only happens if the attacker finds a collision in the hash function H_s, so the success probability in this game is at most negligibly different than in Game 0.

Game 2. Next we also reject ciphertext queries (e_i, u_i, v_i, L_i) during "Probing phase II" such that $v_i \neq v^*$ but $v_i^2 = (v^*)^2$. Observe that since $v_i, v^* < N^2/2$ then the condition above implies finding a nontrivial square root of unity, and hence factoring N. It follows that this modification too can only change the success probability by a negligible amount.

Game 3. We now change the way the target ciphertext is computed. Specifically, we now compute

$$e^* \leftarrow (u^*)^{x_1}(1+N)^{m_\sigma}, \text{ and } v^* \leftarrow \mathsf{abs}\left((u^*)^{x_2+\alpha^* x_3}\right)$$

(where $\alpha^* \leftarrow H_s(u^*, e^*, L^*)$). As these values coincide with the values of e^*, v^* that were computed before, this modification has no effect on the success probability.

Games 4-5. *This is the only difference between our proof the one from [3].* In the proof from [3], Game 4 modifies the choice of u^*, choosing it as a random square in the set of N'th residues modulo N^2 (instead of setting $u^* \leftarrow g^{r^*}$ for a random exponent $r^* \xleftarrow{\$} [N^2/4]$), and Game 5 modified this choice again, choosing u^* as a random square in $Z^*_{N^2}$.

In our proof, we skip Game 4 altogether and in Game 5 we choose u^* as a random element in $\langle g\rangle \cdot \langle 1+N\rangle$. Here we appeal to Lemma 7, which tells us that for an N'th residue g, a random element in $\langle g\rangle$ is indistinguishable from a random element in $\langle g\rangle \cdot \langle 1+N\rangle$. Hence the difference in success probability between Game 3 and Game 5 must be negligible.[9]

Game 6. As done in the proof from [3], we now choose u^* not as a completely random element in $\langle g\rangle \cdot \langle 1+N\rangle$, but rather a random element in that group subject to the restriction that its order is divisible by N. Since a random element in that group satisfies this restriction with all but exponentially small probability, then this has almost no effect on the success probability.

Game 7. We now modify the key-generation algorithm, choosing $x_1, x_2, x_3 \xleftarrow{\$} [N \cdot \varphi(N)/4]$ (instead of choosing them from $[N^2/4]$). This only has an exponentially small effect on the success probability.

Game 8. The last game modifies again the decryption oracle, this time rejecting any ciphertext query (e_i, u_i, v_i, L_i) for which u_i is not an N'th residue modulo N^2.

Denote $\lambda(N) \overset{\text{def}}{=} \varphi(N)/4$ and let $x_1' \leftarrow x_1 \bmod \lambda(N)$ and $x_1'' \leftarrow x_1 \bmod N$. It is fairly easy to see that in this game, the public key and the answers of the decryption oracle are independent of x_1'' and depend on x_1 only through x_1' (recall that g is a $2N$'th residue, so its order divides $\lambda(N)$). Moreover we know that $x_1'' \xleftarrow{\$} [N]$, and we know that the order of u^* is divisible by N (and u^* is a square), which means that u^* can be written as $u^* = w(1+N)^t$ with w a $2N$'th residue and t co-prime with N. Since the element e^* is computed as

$$e^* \leftarrow (u^*)^{x_1}(1+N)^{m_\sigma} = w^{x_1'}(1+N)^{t \cdot x_1'' + m_\sigma},$$

[9] The element g in Lemma 7 was chosen as a random N'th residue, whereas here g is chosen as a random $2N$'th residue. However, it is clear that if $\langle \mu^N \rangle$ is indistinguishable from $\langle \mu^N \rangle \cdot \langle 1+N\rangle$ then also $\langle \mu^{2N} \rangle$ is indistinguishable from $\langle \mu^{2N} \rangle \cdot \langle 1+N\rangle$.

then the distribution of e^* is independent of m_σ (and thus also of σ). It follows that the view of the attacker is independent of σ, and therefore its success probability in this game is exactly 1/2.

It remains to bound the difference between the success probability of the attacker in games 7 and 8. Namely, we need to bound the probability that there exists some decryption query (e_i, u_i, v_i, L_i) in Game 8 such that $v_i = \mathsf{abs}(v_i)$, $u_i^{2(x_2+\alpha_i x_3)} = v_i^2$, the two conditions from Games 1 and 2 do not hold, and yet u_i is not an N'th residue modulo N^2.

Consider a particular decryption query (e_i, u_i, v_i, L_i) for which u_i is not an N'th residue modulo N^2, and denote by o_i the order of u_i in $Z^*_{N^2}$. We first observe that o_i *is not co-prime with* N. Indeed, if o_i were co-prime with N then there would exist integers a, b such that $aN + bo_i = 1$, and therefore

$$(u_i^a)^N = (u_i^a)^N (u_i^{o_i})^b = u_i^{aN+bo_i} = u_i \pmod{N^2},$$

contradicting our working assumption that u_i is not an N'th residue modulo N^2. Since $N = pq$ with p, q primes, it follows that the order of u_i is divisible by either p or q (or both).

The rest of the argument follows the exact same line as in the proof of [3] (but our presentation is slightly different). We observe that the view of the attacker is completely determined by the following values:

- N, g and $x_i \bmod \lambda(N)$ (which completely determine the answers of all the decryption queries),
- u^*, σ and $x_1 \bmod N$ (which together with the values above determine the value of the element e^* of the target ciphertext), and
- $x_2 + \alpha^* x_3 \bmod N$ (which together with the values above determines the value of the element v^* of the target ciphertext.

We therefore consider the alternative view of Game 8 where the values N, g, $x_i \bmod \lambda(N)$, u^*, σ, and $x_1 \bmod N$ are chosen at the outset, and the values of x_2, x_3 are chosen as follows:

- If the i'th decryption query was made during "Probe phase I" then we choose $x_2, x_3 \bmod N$ after the attacker makes this query. Since both x_2, x_3 are uniform in $[N]$ and since u_i, v_i are fixed and the order of u_i is divisible by p or q (and therefore so is the order of u_i^2), then the probability of getting $u_i^{2(x_2+\alpha_i x_3)} = v_i^2$ is at most $1/\min(p, q)$.
- If the i'th decryption query was made during "Probe phase II" then we choose the value of $x_2 + \alpha^* x_3 \bmod N$ after the attacker determines the target messages m_0, m_1, and we choose $x_2 + \alpha_i x_3 \bmod N$ after the attacker makes the i'th decryption query. Since $\alpha_i \neq \alpha^*$ (and they are both smaller than N) then the value of $x_2 + \alpha_i x_3 \bmod N$ is still uniform in $[N]$ even after $x_2 + \alpha^* x_3 \bmod N$ is fixed. Again, this implies that the probability of getting $u_i^{2(x_2+\alpha_i x_3)} = v_i^2$ is at most $1/\min(p, q)$.

We therefore determine that the probability that any decryption query i induces a difference between Game 7 and Game 8 is at most $1/\min(p, q)$, and therefore the difference in the success probability between these two games is at most $\kappa/\min(p, q)$ where κ is the number of decryption queries. This completes the security proof. ■