A Framework for Password-Based Authenticated Key Exchange*

Rosario Gennaro and Yehuda Lindell

IBM T.J.Watson Research Center
Yorktown Heights, NY 10528, USA.
{rosario,lindell}@us.ibm.com

Abstract. In this paper we present a general framework for password-based authenticated key exchange protocols, in the common reference string model. Our protocol is actually an abstraction of the key exchange protocol of Katz et al. and is based on the recently introduced notion of smooth projective hashing by Cramer and Shoup. We gain a number of benefits from this abstraction. First, we obtain a modular protocol that can be described using just three high-level cryptographic tools. This allows a simple and intuitive understanding of its security. Second, our proof of security is significantly simpler and more modular. Third, we are able to derive analogues to the Katz et al. protocol under additional cryptographic assumptions. Specifically, in addition to the DDH assumption used by Katz et al., we obtain protocols under both the Quadratic and N-Residuosity assumptions. In order to achieve this, we construct new smooth projective hash functions.

1 Introduction

A central problem in cryptography is that of enabling parties to communicate secretly and reliably in the presence of an adversary. This is often achieved by having the parties run a protocol for generating a mutual and secret session key. This session key can then be used for secure communication using known techniques (e.g., applying encryption and message authentication codes to all communication). Two important parameters to define regarding this problem relate to the strength of the adversary and the initial setup for the parties.

Adversarial power. The problem of session-key generation was initially studied by Diffie and Hellman [13] who considered a passive adversary that can eavesdrop on the communication of the parties, but cannot actively modify messages on the communication line. Thus, the parties are assumed to be connected by reliable, albeit non-private, channels. Many efficient and secure protocols are known for this scenario. In contrast, in this paper, we consider a far more powerful adversary who can modify and delete messages sent between the parties, as well as insert messages of its own choice. Such an adversarial attack could be carried out by the owner of a routing server on the Internet, for example.

* Extended Abstract. A full version of this paper is available from the *Cryptology ePrint Archive*, http://eprint.iacr.org/2003/032/

Setup assumptions. In order to achieve authenticated key exchange, the parties Alice and Bob must hold some secret information. Otherwise, there is nothing preventing an adversary from pretending to be Bob while communicating with Alice (and vice versa). Thus, some initial setup assumption is required. Known setup assumptions range from the case that the parties share high entropy secret keys to the case that all they share are low entropy *passwords* that can be remembered and typed in by human users. Although many secure and efficient protocols exist for the high entropy case, our understanding of the low entropy case is far from satisfactory. This is despite the fact that the most common setup assumption used today in practice is that of passwords.

This paper focuses on the question of password-based key exchange in the face of a powerful, active adversary. Before proceeding further, we describe this setting in some more detail.

Password-based authenticated key-exchange. We consider a multi-party scenario where all pairs of parties share a password that is chosen uniformly from some small dictionary (the assumption of uniformity is made for simplicity only). The parties interact in a network in which an active adversary has full control over the communication lines. Essentially, this means that the parties cannot communicate directly with each other; rather, all communication is carried out via the adversary. Nevertheless, the parties attempt to generate session keys that they can then use to secretly and reliably communicate with each other. An immediate observation is that in this scenario it is impossible to guarantee that the adversary's success is negligible (where success means, for example, that it succeeds in learning the session key). This is because it can guess the password and then impersonate Bob while communicating with Alice. If its password guess was correct, then it clearly obtains the session key. The important point is that because the password dictionary may be small (i.e., polynomial in the security parameter), the success by the adversary in this naive attack may be quite high. This type of attack is called an on-line guessing attack and is inherent whenever security depends on low entropy passwords. The aim of password-based authenticated key exchange is thus to limit the adversary to such an attack only.

Prior related work. The first (unbroken) protocol suggested for password-based session-key generation was by Bellovin and Merritt [4]. This work was very influential and became the basis for much future work in this area [5, 26, 19, 22, 25, 27]. However, these protocols have not been proven secure and their conjectured security is based on heuristic arguments. Despite the strong need for secure password-based protocols, the problem was not treated rigorously until quite recently.

A first rigorous treatment of this problem was provided by Halevi and Krawczyk [18]. They actually considered an *asymmetric* hybrid model in which one party (the server) may hold a high entropy key and the other party (the human) may only hold a password. The human is also assumed to have secure access to a corresponding public-key of the server. The protocol of [18] provides a password-based solution; however, it requires additional setup assumptions beyond that of human passwords. The first (and only currently known) protocol to achieve security without *any* additional setup is that of Goldreich and Lindell [17]. Their protocol is based on general assumptions (i.e., the existence of trapdoor permutations) and constitutes a proof that password-based authenticated key exchange can actually be obtained. Unfortunately, the [17] protocol is not efficient and thus cannot be used in practice.

Recently, Katz, Ostrovsky and Yung (KOY) [21] presented a highly efficient protocol for the problem of password-based authenticated key-exchange in the common reference string model. In this model, it is assumed that all parties have access to a set of public parameters, chosen by some trusted third party. Although this is a stronger setup assumption than where only human passwords are shared, it is still significantly weaker than other models that have been studied (like, for example, the Halevi–Krawczyk model). Furthermore, in practice there are settings in which such a setup can reasonably be implemented at little cost.[1] The KOY protocol is based on the Decisional Diffie-Hellman (DDH) assumption and has complexity that is only 5–8 times the complexity of unauthenticated key-exchange protocols. We remark that our work was borne out of an abstraction of the KOY protocol.

We note that password-based authenticated key-exchange protocols in the password only setting have been presented in the random oracle model [1, 6]. In this model, all parties are assumed to have oracle access to a totally random (universal) function [2]. The common interpretation of such results is that security is *likely* to hold even if the random oracle is replaced by a ("reasonable") concrete function known explicitly to all parties (e.g., SHA-1). However, it has been shown that it is impossible to replace the random oracle in a generic manner with any concrete function [7]. Thus, the proofs of security of these protocols are actually heuristic in nature.

1.1 Our Contributions

In this paper, we present a framework for password-based authenticated key-exchange protocols in the common reference string model. Our construction is an abstraction of the KOY protocol [21] and uses non-malleable commitments [14], one-time signature schemes and the smooth projective hash functions of Cramer and Shoup [10]. The advantages of this abstraction are as follows:

1. The security of the resulting protocol can be intuitively understood. Our work can thus also be seen as an "explanation" of the KOY protocol (in a similar way to the fact that [10] can be seen as an explanation of [9]).
2. The proof of our protocol is significantly simpler than that of [21], although there are definite similarities in the high-level overview of the proof (see [20]

[1] An example of where it is reasonable to assume a common reference string is when a large organization wishes to implement secure login for its employees. In this case, the organization is trusted to choose the common reference string properly, and this string can then be hardwired into the software code.

for a full proof of the protocol of [21]). Having abstracted out the building blocks of the protocol, the exact requirements on each element of the protocol also become clearer. One specific result of this is that by slightly modifying the protocol of [21], we are able to show that non-malleable commitments suffice (in contrast to [21] whose proof heavily relies on the fact that they use an encryption scheme that is secure against adaptive chosen-ciphertext attacks (CCA2)).

3. The KOY protocol assumes the DDH assumption. We demonstrate additional instantiations of the framework and obtain password-based authenticated key-exchange protocols under both the Quadratic Residuosity and N-Residuosity assumptions. The resulting protocol for N-Residuosity is also highly efficient. In contrast, the protocol based on Quadratic Residuosity is less efficient, but has the advantage of being based on a more standard assumption.

Before presenting our protocol, we briefly (and informally) describe its components (we stress that the descriptions below are very high-level and thus are not accurate):

Non-interactive non-malleable commitments [14]: A non-malleable commitment scheme has the property that given a commitment to a value x, it is hard to generate a commitment to a *related* value y (with probability that is greater than the a priori probability). Non-interactive non-malleable commitments are known to exist in the common reference string model [11, 12]. We actually need these commitments to be *perfectly binding*. In the common reference string model, such schemes can be obtained from any public-key encryption scheme that is non-malleable against chosen-plaintext attacks [12]. The common reference string for the non-malleable commitments is taken as the common reference string for our password protocol.

Smooth projective hashing [10]: Let X be a set and $L \subset X$ a language. Loosely speaking, a hash function H_k that maps X to some set is projective if there exists a projection key that defines the action of H_k over the subset L of the domain X. That is, there exists a projection function $\alpha(\cdot)$ that maps keys k into their projections $s = \alpha(k)$. The projection key s is such that for every $x \in L$ it holds that the value of $H_k(x)$ is uniquely determined by s and x. In contrast, nothing is guaranteed for $x \notin L$, and it may not be possible to compute $H_k(x)$ from s and x. A smooth projective hash function has the additional property that for $x \notin L$, the projection key s actually says *nothing* about the value of $H_k(x)$. More specifically, given x and $s = \alpha(k)$, the value $H_k(x)$ is uniformly distributed (or statistically close) to a random element in the range of H_k.

An interesting feature of smooth projective hashing is that if L is an NP-language, then for every $x \in L$ it is possible to compute $H_k(x)$ using the projection key $s = \alpha(k)$ and a witness of the fact that $x \in L$.

In this paper we prove another important property of smooth projective hash functions that holds when L is a hard-on-the-average NP-language. For a random $x \in_R L$, given x and $s = \alpha(k)$ the value $H_k(x)$ is *computationally indistinguishable* from a random value in the range of $H_k(x)$. Thus, even if $x \in L$, the value $H_k(x)$ is pseudorandom, unless a witness is known. (Of course, as described above, for $x \notin L$ the value of $H_k(x)$ is statistically close to a random element in the range of H_k.)

Our protocol uses a very simple combination of the above tools. In particular, we define a hard-on-the-average NP-language $L = \{(c, m)\}$, where c is a non-malleable commitment to m. (Notice that for a randomly generated commitment, it is hard to know whether or not c is a commitment to m, even given m.) The basic idea behind the protocol is to have the parties exchange non-malleable commitments of the joint password and compute the session key by applying smooth projective hash functions to these commitments. The smooth projective hash functions that they use are based on the hard language L described above. That is, let w be the parties' joint password. Then, for a commitment c we have that $(c, w) \in L$ if and only if c is a commitment of the password w. An informal (and incomplete) description of the protocol appears in Figure 1. We stress that this protocol description is incomplete. In particular, a one-time signature scheme and an additional exchange of a projection key and hash value should also be included.

Security of the protocol framework: We now explain why the protocol is secure. First, consider the case that the adversary passively eavesdrops on a protocol execution between two parties. The security of the session key in this case is based on the above-described property of smooth projective hash functions over hard languages. Specifically, the adversary sees (c, s, c', s') where c and c' are randomly generated commitments of w; in other words, $(c, w), (c', w) \in_R L$. Therefore, $H_k(c, w)$ and $H_{k'}(c', w)$ are pseudorandom and the generated session key is secure. Next, assume that the adversary sends one of the parties a commitment c that it generated itself. If c is a commitment to some value w' which is not the correct password, then the statement (c, w) is *not* in the language L. Therefore, by the smoothness of H_k, the part of the key $H_k(c, w)$ is statistically close to uniform with respect to the adversary (who only sees s and not k). Thus, as above, the generated key meets the requirements for a secret session key. (That is, the adversary has only a negligible advantage in distinguishing the session key output by the parties from a uniformly distributed key that is chosen independently of the protocol.)

The only "bad event" that can occur is if the adversary itself generates a commitment to the correct password w. In this case, we cannot say anything about the security of the session key. However, the adversary can succeed in generating such a commitment with only the same probability as guessing the password outright. This is due to the non-malleability of the commitment scheme which implies that all the "correct" commitments that the adversary sees throughout the executions do not help it in generating any new "correct" commitment. Thus, the adversary's success probability is essentially limited to its probability of guessing the password on-line.

Password-Based Session-Key Exchange

- **Common reference string:** a common reference string ρ for a non-malleable non-interactive commitment scheme.
- **Common input:** a shared (low-entropy) password w.
- **The protocol:**
  1. Party P_1 computes a commitment $c = C_\rho(w)$ using common reference string ρ and sends it to party P_2.
  2. Party P_2 chooses a key k for the smooth projective hash function (for the language L described above), and sends its projection $s = \alpha(k)$ to P_1.
     In addition, P_2 computes another commitment $c' = C_\rho(w)$ and sends it to party P_1.
  3. Party P_1 chooses another key k' for the smooth projective hash function, and sends its projection $s' = \alpha(k')$ to P_2.
- **Session key definition:** Both parties compute the session key to be $H_k(c, w) \oplus H_{k'}(c', w)$.
  1. P_1 computes $H_k(c, w)$ using the projection s and its knowledge of a witness for the fact that c is a commitment to the password w (it knows a witness because it generated c). In contrast, it computes $H_{k'}(c', w)$ using its knowledge of k' (and without a witness).
  2. P_2 computes $H_k(c, w)$ using its knowledge of k, and $H_{k'}(c', w)$ using s' and its knowledge of a witness.

Fig. 1. An informal and incomplete description of the protocol framework

Our constructions of smooth projective hash functions. One of the main contributions of this paper regards our use of the recently introduced notion of smooth projective hash functions [10]. First, we find a new and novel application of this notion to password-based key exchange. Second, we construct new smooth projective hash functions for languages not considered by [10]. Specifically, we construct smooth projective hash functions for the language of pairs (c, m) where c is an encryption of m by a CCA2-secure encryption scheme [14]. This suffices for our protocol framework since any CCA2-secure encryption scheme can be used as a non-interactive non-malleable commitment scheme (in the common reference string model). The KOY protocol implicitly contains one such construction for the encryption scheme of Cramer and Shoup based on the DDH assumption [9]. We prove this fact and then build new smooth projective hash functions for the recent CCA2-secure encryption schemes of Cramer and Shoup [10] that are based on the quadratic residuosity and N-residuosity assumptions. Our constructions of these smooth projective hash functions build on the work of [10]. However, it is important to notice the difference between their constructions and ours.

Consider the case of quadratic residuosity, for example. The smooth projective hash function constructed by [10] is for the language of quadratic residues in Z_N^*. In contrast, our construction is for the far more involved language of pairs of plaintexts and ciphertexts for the entire encryption scheme of [10]. We remark that our constructions require a careful combination of ideas from both [10] and [21].

In addition to providing new constructions, we also make some modifications to the definition of smooth projective hash functions, as presented in [10]. We both strengthen and weaken their definition (in different ways) so that it suffices for our application. This different use may also open the door to further applications.

Editorial Remark: Due to lack of space in this abstract, a number of important sections have been omitted. Specifically, both the formal definition of password-based authenticated key-exchange and the proof of security for our password protocol have been omitted (we remark that we use the definition of [1]). Also, only one out of the three construction of smooth projective hash functions is presented. The full version of the paper contains all of the missing material.

2 Smooth Projective Hash Functions

A central element of our new framework for password-based key exchange is the recently introduced notion of *smooth projective hashing* of Cramer and Shoup [10]. However, their precise definition actually does not suffice for our application. Rather, we use a variant of their definition. In this abstract, we present only our variant of the definition, and briefly describe the differences between the two notions. We begin by defining hard subset membership problems which form the basis for the smooth projective hash functions that are of interest to us.

Notation. The security parameter is denoted by n. For a distribution D, $x \leftarrow D$ denotes the action of choosing x according to D. We denote by $x \in_R S$ the action of uniformly choosing an element from the set S. Finally, we denote statistical closeness of probability ensembles by $\stackrel{s}{\equiv}$, and computational indistinguishability by $\stackrel{c}{\equiv}$.

Subset membership problems. Intuitively, a hard subset membership problem is a problem for which "hard instances" can be efficiently sampled. More formally, a subset membership problem $\mathcal{I}$ specifies a collection $\{I_n\}_{n \in \mathsf{N}}$ such that for every n, I_n is a probability distribution over *problem instance descriptions* Λ. A problem instance description defines a set and a hard language for that set. Formally, each instance description Λ specifies the following:

1. Finite, non-empty sets $X_n, L_n \subseteq \{0,1\}^{\mathrm{poly}(n)}$ such that $L_n \subset X_n$, and distributions $D(L_n)$ over L_n and $D(X_n \backslash L_n)$ over $X_n \backslash L_n$.

2. A witness set $W_n \subseteq \{0,1\}^{\mathrm{poly}(n)}$ and an NP-relation $R_n \subseteq X_n \times W_n$. R_n and W_n must have the property that $x \in L_n$ if and only if there exists $w \in W_n$ such that $(x, w) \in R_n$.

We are interested in subset membership problems $\mathcal{I}$ which are efficiently samplable. That is, the following algorithms must exist:

1. *Problem instance samplability:* a probabilistic polynomial-time algorithm that upon input 1^n, samples $\Lambda = (X_n, D(X_n \backslash L_n), L_n, D(L_n), W_n, R_n)$ from the distribution of problem instances I_n.
2. *Instance member samplability:* a probabilistic polynomial-time algorithm that upon input 1^n and an instance $(X_n, D(X_n \backslash L_n), L_n, D(L_n), W_n, R_n)$, samples $x \in L_n$ according to distribution $D(L_n)$, together with a witness w for which $(x, w) \in R_n$.
3. *Instance non-member samplability:* a probabilistic polynomial-time algorithm that upon input 1^n and an instance $(X_n, D(X_n \backslash L_n), L_n, D(L_n), W_n, R_n)$, samples $x \in X_n \backslash L_n$ according to distribution $D(X_n \backslash L_n)$.

We are now ready to define hard subset membership problems:

Definition 1. (hard subset membership problems): *Let $V(L_n)$ be the following random variable: Choose a problem instance Λ according to I_n, a value $x \in L_n$ according to $D(L_n)$ (as specified in Λ), and then output (Λ, x). Similarly, define $V(X_n \backslash L_n)$ as follows: Choose a problem instance Λ according to I_n, a value $x \in X_n \backslash L_n$ according to $D(X_n \backslash L_n)$ (as specified in Λ) and then output (Λ, x). Then, we say that a subset membership problem $\mathcal{I}$ is* hard *if*

$$\{V(L_n)\}_{n \in \mathsf{N}} \stackrel{\mathsf{c}}{\equiv} \{V(X_n \backslash L_n)\}_{n \in \mathsf{N}}$$

In other words, $\mathcal{I}$ is hard if random members of L_n cannot be distinguished from random non-members (according to the specified distributions). In order to simplify notation, from here on we drop the subscript of n from all sets. However, all mention of sets X and L etc., should be understood as having being sampled according to the security parameter n.

Smooth projective hash functions. We present the notion of smooth projective hash functions in the context of hard subset membership problems. Thus, the sets X and L mentioned below should be thought of those derived from such a problem. Loosely speaking a smooth projective hash function is a function with two keys. The first key maps the entire set X to some set G. The second key (called the projection key) is such that it can be used to correctly compute the mapping of L to G. However, it gives no information about the mapping of $X \backslash L$ to G. In fact, given the projection key, the distribution over the mapping of $X \backslash L$ to G is statistically close to uniform (or "smooth"). We now present the formal definition.

Let X and G be finite, non-empty sets and let $\mathcal{H} = \{H_k\}_{k\in K}$ be a collection of hash functions from X to G. We call K the key space of the family. Now, let L be a non-empty, proper subset of X (i.e., L is a language). Then, we define a *key projection* function $\alpha : K \times X \to S$, where S is the space of key projections. Informally, the above system defines a projective hash system if for $x \in L$, the projection key $s_x = \alpha(k, x)$ uniquely determines $H_k(x)$. (Ignoring issues of efficiency, this means that $H_k(x)$ can be computed given only s_x and $x \in L$.) We stress that the projection key $s = \alpha(k, x)$ is only guaranteed to determine $H_k(x)$ when $x \in L$, and nothing is guaranteed otherwise. Formally,

Definition 2. (projective hash functions): *The family $(\mathcal{H}, K, X, L, G, S, \alpha)$ is a* projective hash family *if for all $k \in K$ and $x \in L$, it holds that the value of $H_k(x)$ is* uniquely determined *by $\alpha(k, x)$ and x.*

Of course, projective hash functions can always be defined by taking $\alpha(k, x) = k$. However, we will be interested in *smooth* projective hash functions which have the property that for every $x \in X\backslash L$, the projection key $s_x = \alpha(k, x)$, reveals (almost) nothing about the value $H_k(x)$. More exactly, for $x \notin L$ chosen according to $D(X\backslash L)$, the distribution of $H_k(x)$ given $s_x = \alpha(k, x)$ should be statistically close to uniform. Formally,

Definition 3. (smooth projective hash functions: *Let $(\mathcal{H}, K, X, L, G, S, \alpha)$ be a projective hash family. Then, for every $x \in X\backslash L$ let $V(x, \alpha(k, x), H_k(x))$ be the following random variable: choose $k \in_R K$ and output $(x, \alpha(k, x), H_k(x))$. Similarly, define $V(x, \alpha(k, x), g)$ as follows: choose $k \in_R K$ and $g \in_R G$ and output $(x, \alpha(k, x), g)$. Then, the projective hash family $(\mathcal{H}, K, X, L, G, S, \alpha)$ is* smooth *if for all $x \in X\backslash L$*

$$\Big\{V(x, \alpha(k), H_k(x))\Big\}_{n\in\mathsf{N}} \stackrel{s}{\equiv} \Big\{V(x, \alpha(k), g)\Big\}_{n\in\mathsf{N}}$$

To summarize, a smooth projective hash function has the property that a projection of a key may be computed which enables the computation of $H_k(x)$ for $x \in L$, but gives almost no information about the value of $H_k(x)$ for $x \notin L$.

Efficient smooth projective hash functions. We say that a smooth projective hash family is efficient if the following algorithms exist:

1. *Key sampling:* a probabilistic polynomial-time algorithm that upon input 1^n samples $k \in K$ uniformly at random.
2. *Projection computation:* a deterministic polynomial-time algorithm that upon input 1^n, $k \in K$ and $x \in X$, outputs $s_x = \alpha(k, x)$.
3. *Efficient hashing from key:* a deterministic polynomial-time algorithm that upon input 1^n, $k \in K$ and $x \in X$, outputs $H_k(x)$.
4. *Efficient hashing from projection key and witness:* a deterministic polynomial-time algorithm that upon input 1^n, $x \in L$, a witness w such that $(x, w) \in R$, and $s_x = \alpha(k, x)$ (for some $k \in K$), computes $H_k(x)$.

We note an interesting and important property of such hash functions. For $x \in L$, it is possible to compute $H_k(x)$ in two ways: either by knowing the key k (as in item 3 above) or by knowing the projection s_x of the key, and a witness for x (as in item 4 above). This property plays a central role in our password-based protocol.

A new property of smooth projective hash functions that we prove in this paper is that for a randomly chosen $x \in_R L$ the value $H_k(x)$ is *computationally indistinguishable* from a random element in the range of H_k, even when given $s_x = \alpha(k, x)$. This essentially implies that the two ways described above to compute $H_k(x)$ are the *only* two efficient ways.

2.1 Differences from the Original Definition

As we have mentioned, our definition of smooth hashing differs from the one presented in [10]. Here we describe the main differences between the two notions:

1. The definition of hard subset membership problems in [10] relates only to the uniform distributions over X and L, whereas we allow any samplable distributions over L and $X \setminus L$.
2. The definition in [10] requires the projection to be a function of the key only; i.e., $\alpha : K \to S$. The value $s = \alpha(k)$ then uniquely defines the value of $H_k(x)$ for *every* $x \in L$. In contrast, we allow the projection to be element dependent. Notice that this is a weaker requirement because smooth hashing of the original definition always satisfies this condition (i.e., define $\alpha(k, x) = \alpha(k)$ for all x).
3. The original smoothness property in [10] relates to the case that the element x is randomly chosen in $X \setminus L$ (according to the distribution $D(X \setminus L)$). However, for our application, we need to deal with the case that x is *adversarially chosen*. We therefore require smoothness with respect to *every* $x \in X \setminus L$. This is clearly a strengthening of the requirement. (We note that the notion of ϵ-universal hashing in [10] does relate to the case of an arbitrary $x \in X \setminus L$; however, their notion of smooth hashing does not.)
   We remark that the weakening of the projection requirement actually makes it harder to satisfy the stronger smoothness property. This is because we must rule out the case that the adversary finds a "bad" x such that s_x reveals more than it should. Indeed, this technical difficulty arises in our constructions of smooth hash functions under the N-Residuosity and Quadratic Residuosity assumptions. Fortunately, it can be shown that these "bad" x's are hard to find and this suffices.

2.2 Our Usage of Smooth Projective Hashing

As we have mentioned, for our password protocol, we construct smooth projective hash functions for a specific family of hard subset membership problems. In this section we describe this family. Let $\mathcal{C}$ be a non-interactive non-malleable perfectly-binding commitment scheme; such schemes are known to exist in the common reference string model. We denote by $C_\rho(m;r)$ a commitment to m using random-coins r and common reference string ρ. Such a commitment scheme is the basis for our hard problem. Let C_ρ denote the space of all strings that may be output by the commitment scheme $\mathcal{C}$ when the CRS is ρ, and let M denote the message space. We note that actually, C_ρ and M must be supersets of these spaces that are *efficiently recognizable*; the actual definition of the supersets depends on the specific commitment scheme used; see Section 4. Next, define the following sets:

- $X_\rho = C_\rho \times M$.
- $L_\rho = \{(c,m) \mid \exists r\ c = C_\rho(m;r)\}$

Having defined the sets X_ρ and L_ρ, we now proceed to define the distributions $D(L_\rho)$ and $D(X_\rho \backslash L_\rho)$. The distribution $D(L_\rho)$ is defined by choosing a random r and a message m (according to some other distribution) and outputting $(C_\rho(m;r),m)$. In contrast, the distribution $D(X_\rho \backslash L_\rho)$ is defined by choosing a random r and outputting $(C_\rho(0^{|m|};r),m)$.[2] Clearly, by the hiding property of C, it holds that for every m, random elements chosen from $D(L_\rho)$ are computationally indistinguishable from random elements chosen from $D(X_\rho \backslash L_\rho)$. This therefore constitutes a hard subset membership problem. (The witness set W_ρ and NP-relation R_ρ are defined in the natural way.)

To summarize, the problem instance sampler for this problem uniformly chooses a common reference string ρ from CRS. This then defines the sets X_ρ, L_ρ, W_ρ and the NP-relation R_ρ. Taking the distributions $D(L_\rho)$ and $D(X_\rho \backslash L_\rho)$ as defined above, we have that

$$\Lambda = (X_\rho, D(X_\rho \backslash L_\rho), L_\rho, D(L_\rho), W_\rho, R_\rho)$$

is a hard subset membership problem.

Our password-based key exchange protocol assumes the existence of a smooth projective hash family for the problem Λ. This hash family $\mathcal{H}$ is indexed by the key space K and it holds that for every $k \in K$,

$$H_k : C_\rho \times M \to G$$

where G is a group of *super-polynomial* size. Let α be the key projection function. We require that α be a function of the hash key and the commitment *only*. That is,

$$\alpha : K \times C_\rho \to S$$

In Section 2.1, we allowed α to depend on the element x, which in this case is a pair (c,m). In our application, m will contain the secret password and we require that α can be computed without knowledge of the password. Therefore, α is a function of k and c, rather than a function of k and (c,m).

[2] In actuality, these distributions need to be defined in a slightly different way, due to the dependence on the message m. However, due to lack of space in this abstract, we slightly simplified the presentation. See the full version for details.

3 The Protocol

Our protocol below uses a non-interactive and non-malleable perfectly-binding commitment scheme. The only known such schemes are in the common reference string (CRS) model, and we therefore write the protocol accordingly. Let ρ denote the CRS and let $C_\rho(m; r)$ denote a commitment to the message m using random coin tosses r and the CRS ρ. We also denote by $C_\rho(m)$ a commitment to m using unspecified random coins. As described in Section 2.2, we use a family of smooth projective functions $\mathcal{H} = \{H_k\}$ such that for every k in the key space K, $H_k : C_\rho \times M \to G$, where M is the message space, C_ρ is an efficiently recognizable superset of $\{C_\rho(m; r) \mid m \in M \ \& \ r \in \{0, 1\}^*\}$, and G is a group of super-polynomial size. Recall that the key projection function α is defined as a function of K and C_ρ. See Section 2.2 for more details.

We assume that there is a mechanism that enables the parties to differentiate between different concurrent executions and to identify who they are interacting with. This can easily be implemented by having P_i choose a sufficiently long random string r and send the pair (i, r) to P_j along with its first message. P_i and P_j will then include r in any future messages of the protocol. We stress that the security of the protocol does not rest on the fact that these values are not modified by the adversary. Rather, this just ensures correct communication for protocols that are not under attack. The protocol appears in Figure 2 on the next page.

Motivation for the protocol. First notice that both P_i and P_j can compute the session key as instructed. Specifically, P_i can compute $H_k(c, VK \circ w \circ i \circ j)$ because it has the projection key s and the witness r for c. Furthermore, it can compute $H_{k'}(c', w)$ because it has the key k' (and therefore does not need the witness r' for c'). Likewise, P_j can also correctly compute both the hash values (and thus the session key). Second, when both parties P_i and P_j see the same messages (c, s, c', s'), the session keys that they compute are the same. This is because the same hash value is obtained when using the hash keys (k and k') and when using the projection keys (s and s'). This implies that partnered parties conclude with the same session key.

We now proceed to motivate why the adversary cannot distinguish a session key from a random key with probability greater that $Q_{\text{send}}/|\mathcal{D}|$, where Q_{send} equals the number of Send oracle calls made by the adversary to different protocol instances and $\mathcal{D}$ is the password dictionary. In order to see this, notice that if P_i, for example, receives c' that is not a commitment to w by CRS ρ, then P_i's session key will be statistically close to uniform. This is because P_i computes $H_{k'}(c', w)$, and for $c' \notin C_\rho(w)$ we have that the statement (c', w) is not in the language L_ρ defined for $\mathcal{H}$ (see Section 2.2). Therefore, by the definition of smooth projective hashing, $\{c', w, \alpha(k, c'), H_k(c', w)\}$ is statistically close to $\{c', w, \alpha(k, c'), g\}$, where $g \in_R G$ is a random element. The same argument holds if P_j receives c that is not a commitment to $VK \circ w \circ i \circ j$. It therefore follows that if the adversary is to distinguish the session key from a random element, it must hand the parties commitments of the valid messages (and in particular

F-PaKE

Common reference string: A common reference string ρ for a non-interactive and non-malleable perfectly-binding commitment scheme C.

Common private input: A password $w = w_{i,j}$.

Messages:

1. Party P_i chooses a key-pair (VK, SK) from a one-time signature scheme, generates a commitment $c = C_\rho(VK \circ w \circ i \circ j; r)$, and sends (VK, c) to P_j.
2. Party P_j receives (VK, c), and checks that c is of the proper format (i.e., $c \in C_\rho$ as defined in Section 2.2). If yes, then, P_j does the following:
   (a) Choose two keys k and k_{test} for the smooth projective hash function family $\mathcal{H}$ defined in Section 2.2 and compute the projections $s = \alpha(k, c)$ and $s_{\text{test}} = \alpha(k_{\text{test}}, c)$.
   (b) Generate a commitment $c' = C_\rho(w; r')$.
   P_j sends (s, c', s_{test}) to P_i.
3. P_i receives (s, c', s_{test}), checks that $c' \in C_\rho$, and does the following:
   (a) Choose a key k' for $\mathcal{H}$ and compute the projection $s' = \alpha(k', c')$.
   (b) Compute the signature $\sigma = \mathsf{Sign}_{SK}(s, s')$.
   (c) Compute the hash $h_{\text{test}} = H_{k_{\text{test}}}(c, VK \circ w \circ i \circ j)$ using the projection key s_{test} and the witness r.
   P_i then sends $(s', \sigma, h_{\text{test}})$ to P_j.

Session-Key Definition:

– P_i computes $sk = H_k(c, VK \circ w \circ i \circ j) + H_{k'}(c', w)$, where addition is in the group G. (Note that P_i computes the first element using the projection key s and the witness r.)
– P_j checks that the following two conditions hold:
  1. $h_{\text{test}} = H_{k_{\text{test}}}(c, VK \circ w \circ i \circ j)$
  2. $\mathsf{Verify}_{VK}((s, s'), \sigma) = 1$.

  If at least one of the above conditions does not hold, then P_j aborts (setting $\mathsf{acc} = 0$). Otherwise, P_j computes $sk = H_k(c, VK \circ w \circ i \circ j) + H_{k'}(c', w)$. (Note that the second element is computed using the projection key s' and the witness r').

Session-Identifier Definition: Both parties take the series of messages (c, s, c', s') to be their session identifiers.

Fig. 2. A framework for Password Based Key Exchange

containing the correct passwords). One way for the adversary to do this is to copy (valid) commitments that are sent by the honest parties in the protocol executions. However, in this case, the adversary does not know the random coins used in generating the commitment, and once again the result of the hash function is pseudorandom in G. This means that the only option left to the adversary is to come up with valid commitments that were not previously sent by honest parties. However, by the non-malleability of the commitment scheme, the adversary cannot succeed in doing this with probability non-negligibly greater than just a priori guessing the password. Since it effectively has Q_{send} password guesses, its success probability is limited to $Q_{\text{send}}/|\mathcal{D}| + \mathsf{negl}(n)$. We remark that the value h_{test} constitutes a test that c is a valid commitment; this is included to ensure that P_j rejects in case it receives an invalid commitment c. This is needed in the proof.

We note one more important point regarding the protocol. Based on the above explanation, one may wonder why it does not suffice to exchange c and s only, without the second exchange of c' and s'. The need for this additional exchange is best demonstrated with the following "attack". Assume that the parties only exchange c and s. Then, the adversary $\mathcal{A}$ can interact with P_i and obtain the commitment c. Next, $\mathcal{A}$ chooses k and returns $s = \alpha(k, c)$ to P_i. Now, if the session key was defined to be $H_k(c, w)$, then $\mathcal{A}$ can distinguish this from random as follows. By traversing the dictionary $\mathcal{D}$ with all possible w's, $\mathcal{A}$ can compile a list of all $|\mathcal{D}|$ possible candidates for the session key ($\mathcal{A}$ can do this because it knows k and so can compute $H_k(c, w)$ without a witness for c). Since $\mathcal{D}$ may be small, this enables $\mathcal{A}$ to easily distinguish the session key from random. This problem is overcome, however, by having the parties repeat the commitment/projection exchange in both directions. We have the following theorem:

Theorem 1. *Assume that C is a non-interactive and non-malleable perfectly-binding commitment scheme that is secure under multiple commitments, and assume that $\mathcal{H}$ is a family of smooth projective hash functions. Then, Protocol* F-PaKE *in Figure 2 is a secure password-based session-key generation protocol.*

The proof of Theorem 1 appears in the full version of the paper. Our proof of security differs from the proof of [21] in a number of ways. First, our proof is based on more generic primitives and is therefore more modular. Second, our proof holds for the case that the parties commit to their passwords etc. using a non-malleable commitment scheme. In contrast, the proof of [21] requires that the parties "commit" using a CCA2-secure encryption scheme. In particular, their proof uses the capability of decryption in an essential way, whereas ours does not.

Using CCA2-encryption instead of non-malleable commitments. If a CCA2-encryption scheme is used to implement the non-malleable commitments, then the s_{test} and h_{test} values can be removed from the protocol, improving the efficiency. The proof of the security of the protocol with this modification is actually very different from our proof of Theorem 1, and is quite similar to the proof of [21]. This version of the protocol has the advantage of being slightly more efficient.

Protocol F-PaKE *and the protocol of* KOY [21]. Consider the above-mentioned modification of the protocol that relies on CCA2 encryption. Then, a protocol almost identical to that of [21] is obtained by using the CCA2-encryption scheme of Cramer and Shoup [9] that is based on the DDH assumption. Indeed, as we have mentioned, our protocol framework was obtained through an abstraction of [21].

4 Encryption Schemes With Smooth Projective Hashing

As we discussed in Section 2.2, the underlying language for the hard subset membership problem that we refer to is the language of pairs (c, m) where $c = C_\rho(m)$. However, in all our specific instantiations of the protocol, we use a CCA2-secure encryption scheme for the non-malleable commitment. Thus, in the context of our protocol, the common reference string ρ is defined to be the public-key pk of the encryption scheme, and a commitment to m is defined as an encryption $E_{pk}(m)$.

In the full paper, we construct smooth projective hash functions for all three CCA2-secure encryption schemes proposed by Cramer and Shoup in [9, 10]. The smooth hashing for the DDH-based scheme from [9] was already implicitly described in the KOY protocol [21]. We explicitly define the construction and prove that it constitutes a smooth projective hash family. Furthermore, we construct new smooth projective hash functions for the CCA2-schemes based on the Quadratic Residuosity and N-Residuosity Assumptions presented in [10]. In this abstract we only present the construction based on the N-residuosity assumption.

4.1 A Relaxed Notion of Smooth Projective Hashing

The smooth projective hash functions we present do not satisfy the definition we presented in Section 2 but rather a relaxed definition.[3] This weaker notion is still sufficient to prove the security of our password-based key exchange protocol.

The relaxation here involves modifying the stronger smoothness condition as defined in Section 2. Informally speaking, we no longer require that the smoothness condition hold for *all* values $x \in X \setminus L$. Rather, we define a subset $\Delta \subset X \setminus L$ in which the smoothness property may not hold. However, we require that it is computationally hard to find any element in Δ. This suffices because the result is that a computationally bound adversary cannot produce any "bad" elements for which the smoothness property will not hold. Formally, we say that the family

[3] We remark that the smooth projection hash functions for the encryption scheme based on the DDH assumption do satisfy the stricter original definition.

$\mathcal{H}$ is a weak smooth projective hash family if the stronger smoothness condition is replaced by the following two conditions:

1. There exists a subset $\Delta \subset X \backslash L$ which is hard to sample, i.e. for all probabilistic polynomial-time Turing Machines A

$$\Pr[A(1^n, X, L) \in \Delta] < \mathsf{negl}(n)$$

Furthermore, it is easy to verify membership in Δ. That is, there exists a probabilistic polynomial-time Turing Machine T such that for every $x \in \Delta$, $\Pr[T(X, L, x) = 1] > 1 - \mathsf{negl}(|x|)$ and for every $x \notin \Delta$, $\Pr[T(X, L, x) = 1] < \mathsf{negl}(|x|)$;

2. For *every* $x \in X \setminus (\Delta \cup L)$:

$$\Big\{V(x, \alpha(k, x), H_k(x))\Big\}_{n \in \mathsf{N}} \overset{\mathrm{s}}{\equiv} \Big\{V(x, \alpha(k, x), g)\Big\}_{n \in \mathsf{N}}$$

The random variable V is defined in Section 2.

In other words, the strong smoothness property only holds for values $x \notin \Delta$. In particular, it may be possible to distinguish $H_k(x')$ from a random g for $x' \in \Delta$. This would be worrisome were it not for the first condition above that tells us that the probability that the adversary will find such an $x \in \Delta$ is negligible.

4.2 The Cramer-Shoup scheme based on N-residuosity

This construction is for the Cramer-Shoup CCA2-secure encryption scheme based on the hardness of deciding N-residuosity in $Z^*_{N^2}$ [10]. This assumption was originally introduced by Paillier in [24]. We describe a small variation of the original Cramer-Shoup scheme, which can easily be proven secure using the same techniques used in [10].

Let $N = pq$ be the product of two safe primes, i.e. $p = 2p' + 1$ and $q = 2q' + 1$, with p', q' prime as well. Let $N' = p'q'$. Assume w.lo.g. that $q' < p'$ and that $|q'| = |p'| = \mathsf{poly}(n) > 2n$ where n is the security parameter.

Consider the group $Z^*_{N^2}$; its order is $4NN'$. Let's consider the subgroup J_N of $Z^*_{N^2}$ which contains all the elements whose Jacobi symbol with respect to N is 1. It is not hard to see that J_N is cyclic, has order $2NN'$ and can be written as the direct product of three cyclic subgroup $J_N = G \cdot G_1 \cdot G_2$ where G is the subgroup of J_N which contains all the $(2N)$-residues. Clearly G has order N'. On the other hand G_1 is a group of order N and G_2 is the group generated by (-1). Denote $G' = G \cdot G_2$. See [10] for details.

A generator g for G' can be found by selecting $\mu \in_R Z^*_{N^2}$ and setting $g = -\mu^{2N}$. It is not hard to see that this results in a generator with overwhelming probability, and that the distribution is statistically close to uniform over all generators of G'. Clearly g^2 will then be a generator for G.

The N-residuosity assumption says that it's hard to distinguish between a random element of $Z^*_{N^2}$ and a random N-residue mod N^2. The following encryption scheme is CCA2-secure under this assumption.

A preliminary Lemma. Before describing the scheme we prove a technical Lemma that will be useful later in the construction of the smooth projective hash function for it. The Lemma states that, if factoring is hard, it is computationally infeasible to find elements of "low" order in the group G.

Lemma 1. *Let $N = pq$, with $p = 2p' + 1$, $q = 2q' + 1$ and p, q, p', q' all distinct primes. Let g' be a generator of G as above and let $h = (g')^z \bmod N$, with $h \neq \pm 1$. If $GCD(z, N') > 1$ then $GCD(h \pm 1, N) > 1$.*

We now describe the scheme.

Key Generation. Randomly choose the secret key $z, \tilde{z}, \hat{z} \in [0..N^2/2]$ and publish the public key $h = g^z$, $\tilde{h} = g^{\tilde{z}}$ and $\hat{h} = g^{\hat{z}}$. We assume w.l.o.g that $h \neq 1$ and $GCD(\tilde{z}, N') = 1$ since the opposite happens only with negligible probability. The public key also includes a universal one-way hash function [23] H which maps inputs to Z_N.

Encryption. To encrypt a message $m \in Z_N$, choose $r \in_R [0..N/4]$ and compute $u = g^r$, $e = (1 + mN)h^r$ and $v = \|(\tilde{h}\hat{h}^\theta)^r\|$, where $\theta = H(u, e)$ and the function $\| \cdot \|$ is defined as $\|v\| = v$ if $v \leq N^2/2$ and $\|v\| = N^2 - v$ otherwise[4].

Decryption. We describe the decryption mechanism for completeness. Given a ciphertext (u, e, v) the receiver checks that $v \leq N^2/2$, then computes $\theta = H(u, e)$ and checks if $v^2 = u^{2(\tilde{z}+\theta\hat{z})}$. If either test fails it outputs "?", otherwise let $\tilde{m} = eu^{-z}$. If $\tilde{m} = (1 + mN)$ for some m, output m, otherwise output "?".

A smooth projective hash function. We define a smooth projective hashing for this encryption scheme, by specifying the sets X, L, K, the projection function α, and the hash function H_k. In order to define the set C (that must be an efficiently recognizable superset of all possible ciphertexts; see Section 2.2), we first define the notion of a proper ciphertext: A ciphertext $c = (u, e, v)$ is called proper if u, e, v all have Jacobi symbol equal to 1, with respect to N. Also given $\theta = H(u, e)$, we require that if $\tilde{h}\hat{h}^\theta = 1$ then $v = 1$ as well. We now define $X = \{(c, m)\}$, where c is any proper ciphertext and m is any message. Observe that proper ciphertexts can be easily recognized, as required. As defined in Section 2.2, the language L is the subset of X where c is an encryption of m with public-key pk.

The key space is $[0..2NN']^3$ i.e. the key for a hash function is $k = (a_1, a_2, a_3)$ such that $a_i \in_R [0..2NN']$. However this space is not efficiently samplable so we replace it with $[0..N^2/2]$. The uniform distribution over $[0..N^2/2]$ is statistically close to the uniform one over $[0..2NN']$. The description of the family also includes a hash function UH randomly chosen from a 2-universal family. $UH : J_N \longrightarrow \{0, 1\}^n$.

[4] The full version explains why the *absolute value function* $\| \cdot \|$ is needed.

Given an input $x = (c, m) = ((u, e, v), m)$ the projection function is defined by

$$s_x = \alpha(k, x) = g^{2a_1} h^{2a_2} (\tilde{h}\hat{h}^\theta)^{2a_3}$$

where $(g, h, \tilde{h}, \hat{h})$ constitutes the public key and $\theta = H(u, e)$.

Given an input $x = (c, m) = ((u, e, v), m)$ the hash function is defined as $H_k(x) = UH[f_k(x)]$ where

$$f_k(x) = u^{2a_1} \left(\frac{e}{1 + mN} \right)^{2a_2} v^{2a_3}$$

Notice that if c is a correct encryption of m under key pk, then $f_k(x) = s_x^r$, where r is the randomness used to construct c. Thus, it is possible to compute $H_k(x)$ given only the projection and the witness, as required.

We need to prove the smoothness property. Consider $x = (c, m) \in X \backslash L$; i.e., $c = (u, e, v)$ is a *proper* ciphertext, but is *not* a correct encryption of m using the public key $(g, h, \tilde{h}, \hat{h})$. Let $\theta = H(u, e)$ and consider $\lambda = \tilde{z} + \theta\hat{z} \bmod N'$. Notice that we can write $(\tilde{h}\hat{h}^\theta)^2 = g^{2\lambda}$.

We are going to prove that H_k is a *weak* smooth projective hash. Thus we need to define a set Δ of commitments for which the smoothness property may not hold, but such that finding an element in Δ is infeasible. We say that a commitment is in Δ if $GCD(\lambda, N') \neq 1, N'$. Lemma 1 shows that producing a commitment with this property is equivalent to factoring N. It also shows how to easily recognize such elements (just test if $GCD((\tilde{h}\hat{h}^\theta)^2 \pm 1, N)$ is a non-trivial factor of N). From now on, we assume to be outside of Δ.

By assumption, if $(\tilde{h}\hat{h}^\theta)^2$ is different than 1, then it is also a generator for G. Notice that g^2 and h^2 are also generators of G, thus we can write the commitment as $u = (-1)^{b_1}\gamma_1(g^2)^{r_1}$, $e = (-1)^{b_2}\gamma_2(h^2)^{r_2}(1 + mN)$ and $v = (-1)^{b_3}\gamma_3[(\tilde{h}\hat{h}^\theta)^2]^{r_3}$ where $\gamma_i \in G_1$. Notice that in the computation of $H_k(x)$ the (-1) components are irrelevant since we raise each term to the power $2a$. So we ignore them from now on.

Consider now the equation in the a_i's defined by the projection s_x. Notice that $s_x \in G$ since we are squaring each term. Therefore:

$$\log_{g^2} s_x = a_1 + za_2 + \lambda a_3 \mod N' \tag{1}$$

Let us distinguish two cases:

- There exists $\gamma_i \neq 1$. Then we have that $f_k(x) = (\gamma_i^2)^{a_i} \cdot \sigma$ for some $\sigma \in J_N$. Let $[a_1, a_2, a_3]$ be a solution of Equation (1), mod N'. Consider the set of keys obtained as

$$A_{a_1,a_2,a_3} = [a_1 + d_1N', a_2 + d_2N', a_3 + d_3N']$$

  for $d_i \in [0..2N - 1]$. Given s_x the key k is uniformly distributed in the union over all $[a_1, a_2, a_3]$ which are solutions of Eq. (1) of the sets A_{a_1,a_2,a_3}. Let's focus on one specific set A_{a_1,a_2,a_3}. Recall that $\gamma_i \in G_1$ and $\gamma_i \neq 1$ so its order is either p, q, N. Since 2 is co-prime with these values, the order of γ_i^2 is the same as the one of γ_i. And since $GCD(N, N') = 1$ we have that keys of the form $a_i + d_i N$ map $(\gamma_i^2)^{a_i+d_iN}$ uniformly over the group generated by γ_i.

In conclusion, given s_x the value $(\gamma_i^2)^{a_i}$ and consequently $f_k(x)$, can be guessed with probability at most $1/q$.

- $\gamma_i = 1$ for all $i = 1, 2, 3$. Then $f_k(x)$ is an element of G and thus its discrete log with respect to g^2 defines another equation in the a_i's (recall that $GCD(2, N') = 1$ thus 2^{-1} is well defined mod N'):

$$2^{-1} \log_{g^2} f_k(x) = r_1 a_1 + r_2 z a_3 + r_3 \lambda a_3 \mod N' \tag{2}$$

We want to prove that Equations (1) and (2) are "linearly independent", i.e. the 2×3 matrix of the coefficients of these equations in the a_i's has at least one 2×2 minor whose determinant is non-zero mod N'.

Since c is not a correct commitment to m then we must have that either $r_2 \neq r_1$ or $r_3 \neq r_1$. Then the corresponding minors have determinant $z(r_2 - r_1)$ and $\lambda(r_3 - r_2)$. Since z and λ are co-prime with N' (recall that we are outside of Δ), we have that at least one of these minors has non-zero determinant.

If this determinant is invertible mod N', then it is easy to see that $f_k(x)$ is uniformly distributed over G given s_x.

Assume now that both determinants are non-invertible. Then it must be that both $r_2 - r_1$ and $r_3 - r_1$ are multiples of either p' or q'. Assume w.l.o.g. that $r_2 = r_1 + \gamma_2 p'$ and $r_3 = r_1 + \gamma_3 q'$ (the argument is the same if you switch p' and q'). Then the corresponding minor has determinant $z\lambda(r_3 - r_2)$ which is invertible mod N'. Thus again $f_k(x)$ is uniformly distributed over G given s_x.

Finally we are left with the case in which $r_2 = r_1 + \gamma_2 p'$ and $r_3 = r_1 + \gamma_3 p'$ (again w.l.o.g. since the argument is the same for q'). This means that Equations (1) and (2) are linearly dependent mod p' but they must be linearly independent mod q' (recall that either $r_2 \neq r_1$ or $r_3 \neq r_1$ mod N'). Thus $\log_g f_k(x)$ is uniformly distributed mod q'.

In any case we can bound the probability of guessing $H_k(x)$ given s_x, with $1/q'$.

In conclusion we have that given s_x the value $f_k(x)$ can be guessed with probability at most $1/q' < 2^{-2n}$ by choice of the security parameter. Applying the properties of 2-universal hash functions we have that the distribution of $H_k(x)$ is $2^{-n/3}$ close to uniform over $\{0, 1\}^n$.

Acknowledgements: We are grateful to both Jonathan Katz and Victor Shoup for answering our questions about their respective papers [21] and [10]. We also thank Jonathan for some helpful comments on the presentation.

References

1. M. Bellare, D. Pointcheval and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In *Eurocrypt 2000*, Springer-Verlag (LNCS 1807), pages 139–155, 2000.

2. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *1st Conf. on Computer and Communications Security*, ACM, pages 62–73, 1993.
3. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *CRYPTO'93*, Springer-Verlag (LNCS 773), pages 232–249, 1994.
4. S. M. Bellovin and M. Merritt. Encrypted Key Exchange: Password based protocols secure against dictionary attacks. In *Proceedings 1992 IEEE Symposium on Research in Security and Privacy*, pages 72–84. IEEE Computer Society, 1992.
5. S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *Proceedings of the 1st ACM Conference on Computer and Communication Security*, pages 244–250, 1993.
6. V. Boyko, P. MacKenzie and S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In *Eurocrypt 2000*, Springer-Verlag (LNCS 1807), pages 156–171, 2000.
7. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. In 30*th STOC*, pages 209–218, 1998.
8. R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Eurocrypt 2001*, Springer-Verlag (LNCS 2045), pages 453–474, 2001.
9. R. Cramer and V.Shoup. A practical public-key cryptosystem secure against adaptive chosen ciphertexts attacks. In *CRYPTO'98*, Springer-Verlag (LNCS 1462), pages 13–25, 1998.
10. R. Cramer and V.Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *Eurocrypt 2002*, Springer-Verlag (LNCS 2332), pages 45–64, 2002.
11. G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-Interactive and Non-Malleable Commitment. In 30*th STOC,* pages 141–150, 1998.
12. G. Di Crescenzo, J. Katz, R. Ostrovsky and A. Smith. Efficient and Non-interactive Non-malleable Commitment. In *Eurocrypt 2001*, Springer-Verlag (LNCS 2045), pages 40–59, 2001.
13. W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Trans. on Inf. Theory*, IT-22, pp.644–654, Nov. 1976.
14. D. Dolev, C. Dwork and M. Naor. Non-malleable Cryptography. *SIAM Journal of Computing*, 30(2):391–437.
15. C. Dwork. *The non-malleability lectures.* Course notes for CS 359, Stanford University, Spring 1999. Available at: `theory.stanford.edu/~gdurf/cs359-s99`.
16. O. Goldreich. *Foundations of Cryptography – Basic Tools.* Cambridge University Press. 2001.
17. O. Goldreich and Y. Lindell. Session Key Generation using Human Passwords Only. In *CRYPTO 2001*, Springer-Verlag (LNCS 2139), pages 408–432, 2001.
18. S. Halevi and H. Krawczyk. Public-Key Cryptography and Password Protocols. In *ACM Conference on Computer and Communications Security*, 1998.
19. D.P. Jablon. Strong password-only authenticated key exchange. *SIGCOMM Computer Communication Review*, 26(5):5–26, 1996.
20. J. Katz. *Efficient Cryptographic Protocols Preventing "Man-in-the-Middle" Attacks.* Ph.D. Thesis, Columbia University, 2002.
21. J. Katz, R. Ostrovsky and M. Yung. Practical Password-Authenticated Key Exchange Provably Secure under Standard Assumptions. In *Eurocrypt 2001*, Springer-Verlag (LNCS 2045), pp.475–494, 2001.

22. S. Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Proceedings of the Workshop on Security Protocols*, Ecole Normale Superieure, 1997.
23. M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Applications. In 21*st STOC*, pages 33–43, 1989.
24. P. Paillier. Public-Key Cryptosystems based on Composite Degree Residue Classes. In *EUROCRYPT'99*, Springer-Verlag (LNCS 1592), pages 223–228, 1999.
25. S. Patel. Number theoretic attacks on secure password schemes. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 236–247, 1997.
26. M. Steiner, G. Tsudik and M. Waidner. Refinement and extension of encrypted key exchange. *ACM SIGOPS Oper. Syst. Rev.*, 29(3):22–30, 1995.
27. T. Wu. The secure remote password protocol. In *1998 Internet Society Symposium on Network and Distributed System Security*, pp.97–111, 1998.