Password Authenticated Key Exchange Using Hidden Smooth Subgroups

[Extended Abstract]

Craig Gentry
DoCoMo USA Labs
181 Metro Dr.
San Jose, CA USA
cgentry@docomolabs-usa.com

Philip Mackenzie
DoCoMo USA Labs
181 Metro Dr.
San Jose, CA USA
philmac@docomolabs-usa.com

Zulfikar Ramzan
DoCoMo USA Labs
181 Metro Dr.
San Jose, CA USA
ramzan@docomolabs-usa.com

ABSTRACT

Existing techniques for designing efficient password authenticated key exchange (PAKE) protocols all can be viewed as variations of a small number of fundamental paradigms, and all are based on either the Diffie-Hellman or RSA assumptions. In this paper we propose a new technique for the design of PAKE protocols that does not fall into any of those paradigms, and which is based on a different assumption. In our technique, the server uses the password to construct a multiplicative group with a (hidden) smooth order subgroup, where the group order depends on the password. The client uses its knowledge of the password to generate a root extraction problem instance in the server's group and a discrete logarithm problem instance in the (smooth order) subgroup. If the server constructed its group correctly based on the password, the server can use its knowledge of the group order to solve the root extraction problem, and can solve the discrete logarithm problem by leveraging the smoothness of the hidden subgroup.

The resulting scheme is provably secure (in the random oracle model) under the "decision subgroup assumption." The scheme can be efficiently instantiated using composite modulus groups, in which case the client and server each perform the equivalent of a small number of modular exponentiations, and the security reduces to a simple variant of the "Φ-hiding" assumption. We provide preliminary implementation results of this instantiation.

Categories and Subject Descriptors

C.2.2 [**Computer Communications Networks**]: Network Protocols; E.3 [**Data Encryption**]: Public key cryptosystems

CCS'05, November 7–11, 2005, Alexandria, Virginia, USA.

General Terms

Security

Keywords

cryptography, password, authentication, key exchange

1. INTRODUCTION

The Password Authentication Problem. Consider two parties, Alice and Bob, who wish to communicate securely over an insecure network, but whose only means of verifying each other's identity consists of a short secret password (e.g., a 4-digit PIN number). In particular, neither of them knows a public key corresponding to the other party, and neither has a certified public key (i.e., a public key whose certificate can be verified by the other party). Here, Alice should be concerned not only with eavesdroppers, but also with the party with whom she is communicating since a priori she cannot even be certain that it is Bob. Bob's situation is similar.

If Alice and Bob shared a high-strength cryptographic key (i.e., a *long* secret), then this problem could be solved using standard solutions for setting up a secure channel (e.g., [4]). However, since Alice and Bob only share a short secret password, they must also be concerned with *offline dictionary attacks.* An offline dictionary attack occurs when an attacker obtains some information that can be used to perform offline verification of password guesses. We will call this *password verification information.* For a specific example, consider the following. Say Alice and Bob share a password π, and say an attacker somehow obtained a hash of the password $h(\pi)$, where h is some common cryptographic hash function such as SHA-1 [40]. Then an attacker could go offline and run through a dictionary of possible passwords $\{\pi_1, \pi_2, \ldots\}$, testing whether $h(\pi_i) = h(\pi)$. In general, the password verification information obtained by the attacker may not be as simple as a hash of a password, and an attacker may not always be able to test all possible passwords against the password verification information, but if he can test a significant number of passwords, this is still considered an offline dictionary attack. See Wu [46] for a fairly recent demonstration of how effective an offline dictionary attack can be.

Common Approaches to Password Authentication. Unlike password authenticated key exchange, discussed be-

low, most common techniques for password authentication are *unilateral* authentication techniques – that is, only one party (a user or client) is authenticated to the other party (a server), but not vice-versa; they are also vulnerable to offline dictionary attacks or rely on certified (or otherwise authenticated) public keys.

The simplest password authentication technique is for the client to send a password to the server in the clear. This technique is used in some older Internet applications, as well as many web-based mail applications. Obviously this is completely insecure against an eavesdropper on the network, but is often considered acceptable on channels in which eavesdropping is relatively difficult.

A more advanced technique is challenge-response, in which the server sends a challenge to the client, and the client responds with a message depending on the challenge and the password, for instance the hash of the challenge and password concatenated. This type of authentication is used in some operating systems to enable network access. It is vulnerable to an offline dictionary attack by an eavesdropper since the challenge and its corresponding response, together, make password verification information.

A more secure technique sends a password to the server over an *anonymous secure channel*, in which the server has been verified using a public key. This type of authentication is used in some remote terminal applications, as well as web-based applications, and it depends intrinsically on the ability of the client to verify the server's public key (otherwise, an attacker can impersonate the server). When used on the web, the public key of the server is certified by a certification authority that is presumably trusted by the client. For remote terminal applications, there typically is no trusted third party, and security relies on the client recognizing the public key, perhaps with a "fingerprint," or hash, of the public key.

Password Authenticated Key Exchange (PAKE). The purpose of PAKE is to provide mutual password authentication without pre-authenticated public keys and in such a way that the only feasible way to attack the protocol is to run a trivial *online dictionary attack* of simply iteratively guessing passwords and attempting to impersonate one of the parties. (Note that online attacks are easier to detect and thwart.) Using a PAKE protocol, the authenticating parties can "bootstrap" a short secret (the password) into a long secret (a cryptographic key) that thereafter can be used to provide a secure channel.

The problem of designing a secure PAKE protocol was proposed by Bellovin and Merritt [6] and by Gong *et al.* [25], and has since been studied extensively. Many PAKE protocols have been proposed, e.g., [7, 25, 24, 28, 29, 36, 44, 45, 33, 32], and many of these protocols have been shown to be insecure (see e.g., [41]). Recent protocols have proofs of security, based on certain well-known cryptographic assumptions, although some of these proofs assume the existence of ideal hash functions or ideal ciphers (i.e., black-box perfectly-random functions (random oracles) or keyed permutations, respectively). A few recent papers [2, 10, 1] present refinements of the EKE protocol of [7] and prove security based on the Diffie-Hellman (DH) assumption [19]. The first assumes both ideal ciphers and ideal hashes, while the others assume only ideal hashes. Other papers [37, 47] present refinements of the OKE protocol of [36] and prove security based on the RSA assumption [43]. These all assume ideal hashes. Another paper [31] presents a new protocol based on a variant of the Cramer-Shoup cryptosystem [16] and proves security based on the decisional DH assumption (see, e.g., [8]), assuming only a public random string (not an ideal hash function). Some variants of the [31] protocol are presented in [21, 30, 13]. Another password-authenticated key exchange protocol was developed in [23] and proven secure based on trapdoor permutations without any setup assumptions, but with a restriction that concurrent sessions with the same password are prohibited.

Overall, existing techniques for designing efficient PAKE protocols all can be viewed as variations of a small number of fundamental paradigms, and the most efficient of them are based on either the Diffie-Hellman or RSA assumptions. In particular, the existing techniques for designing efficient and provably secure PAKE protocols may be viewed as falling into one of the following two basic paradigms:

- the password is used to encrypt (or more generally, "entangle") some part of a message that is being used to perform key exchange, e.g., [1, 6, 10, 31, 37, 46, 47],[1] or
- the password is used to choose a parameter in a standard key exchange, e.g., [28, 32].

There is a third approach to achieving PAKE – namely, using oblivious polynomial evaluation (OPE), a primitive introduced by Naor and Pinkas [39]. OPE is a more general form of oblivious transfer (OT), first suggested by Rabin [42]. Goldreich and Lindell [23], following a suggestion of [39], showed that, by using OPE, one can achieve PAKE in the standard model using only trapdoor permutations. Although these are important theoretical contributions, the PAKE protocols based on OPE are not competitive with the most efficient PAKE protocols.

Our Contributions. We present the first practical and provably secure scheme that does not fall within the two basic paradigms mentioned above, but instead employs oblivious transfer (or at least, key aspects of oblivious transfer) to achieve PAKE.

At a high level, our technique, based on oblivious transfer, works as follows. First, recall that in a 1-out-of-2 (string) OT protocol, the sender starts with two strings r_0 and r_1, and the chooser starts with a bit $b \in \{0, 1\}$; by the end of the protocol, the chooser learns r_b without learning any information about r_{1-b}, and without revealing any information about b to the sender. In our scheme, the client plays the role of "sender," while the server acts as a "chooser."

More specifically, to slightly oversimplify our scheme, the client generates a random "database" with n pairs of entries, each of which is an 2ℓ-bit string. Passwords are mapped (preferably injectively) to n-bit strings. After the client initiates the protocol, the server (roughly speaking) constructs n 1-out-of-2 OT queries to recover one database entry from

[1]Obviously, for this paradigm we are viewing "entanglement" and "key exchange" quite broadly. For example, the protocol of Katz *et al.* [31] can be viewed as an instance of key exchange using *smooth projective hashing* (see [21, 17]), in which the password is used to entangle an input to a projective hash function that is used to derive the shared key (where "entangling" is multiplication by a generator raised to a function of the password).

each of the n pairs; for each i, the server requests the entry that corresponds to the i^{th} bit in the n-bit string associated to the password. The client responds to the server's queries, and also sends a confirmation value – e.g., a hash that includes as input the n database entries corresponding to the password as well as the password itself. [2] The eventual shared cryptographic key is also generated as a function of the n entries corresponding to the password. Security against a malicious client follows from the chooser-privacy of the OT scheme. Roughly speaking, security against a malicious server follows from sender-privacy. The server can use the client's OT responses to test only one password – i.e., the password that the server used to parameterize its OT queries; if the server guessed wrong, sender-privacy ensures that the server will have essentially "no information" about one of the high-entropy (2ℓ-bit) strings that the client input to the hash function, and therefore will not be able to use the client's hash as password verification information.

While basing PAKE on OT in this way may itself be interesting theoretically, in this paper we focus on constructing a PAKE scheme that is usable in practice. In practice, we find that we can improve the performance of our scheme by using error-correcting codes – i.e., we map a password to a k-bit string for $k < n$, and then to an n-bit string using a code with minimum distance d. In this setting, the server can feasibly test only the password whose n-bit string is closest (in terms of Hamming distance) to the n-bit string associated with its n OT queries; for all other candidate passwords, it will (roughly speaking) lack at least $d\ell$ bits that the client input to the hash function. We find various performance "sweet spots" for (n, k, d, ℓ), subject to the constraints that $d\ell$ must be large enough (e.g., 80) to make offline guessing infeasible, $n\ell$ is reasonably small for efficiency, and (n, k, d) corresponds to an efficient error correcting code. In this setting, we then find that our high-level OT-based description above is actually stronger than what we need, since we do not need the 1-out-of-2 OT schemes to be secure individually, as long as they are, in some sense, secure in the aggregate. Moreover, our PAKE scheme only needs to work for some set of sufficiently random databases (rather than all databases). Our PAKE scheme also employs a number-theoretic method for efficiently "batching" the n OT queries and responses above into a single short query and response.

Briefly, we sketch the number-theoretic instantiation of our scheme. The scheme uses n (public) pairs of prime numbers $\{(p_{i,0}, p_{i,1}) : i \in [1, n]\}$, and employs two problems that are, in some sense, "duals" of each other – namely (for cyclic group G and prime p), it uses:

- (Discrete Log (Weakened Version)): Given generator $g \in G$ and $h = g^x$, compute $x \bmod p$;
- (Root Extraction): Given $j \in G$, compute $j^{1/p}$.

Why are these problems "duals" of each other? First, consider the case where p divides the group order $|G|$. Then, the above discrete log problem has a *unique* solution in $\{0, \ldots, p-1\}$. However, if j is a p-residue in G, the solution to the root extraction problem is *not* unique; there are p valid solutions. Next, consider the case where p does not divide $|G|$; here, the "solvability" situation of the two problems is reversed. The root extraction problem has a unique solution. (RSA decryption is based on this fact.) However, the discrete log problem has p valid solutions. Specifically, if $x' \in \mathbb{Z}$ satisfies $h = g^{x'}$, then $h = g^x$ for all $\{x = x' + r|G| : r \in \mathbb{Z}\}$; since $\gcd(p, |G|) = 1$, the values of $x \bmod p$ cover all of $\{0, \ldots, p-1\}$.

Keeping this in mind, the server constructs the abovementioned "OT queries" by mapping the password to its n-bit string $b_1 \cdots b_n$, and then ensuring that its group G has order divisible by all of the primes p_{i,b_i} but none of the primes $p_{i,1-b_i}$; it sends this group and a generator to the client. The client uses G to generate "challenges" to the server to solve instances of the discrete log problem for the p_{i,b_i}'s and instances of the root extraction problem for the $p_{i,1-b_i}$'s; the set of answers will be unique only if the order of G is divisible (and not divisible) by the correct primes. In our scheme, these problems are batched, so that the server performs a single discrete logarithm problem in a subgroup of G with order $\prod_{i=1}^{n} p_{i,b_i}$ and computes a single $(\prod_{i=1}^{n} p_{i,1-b_i})^{th}$ root of a group element. Since we choose the $p_{i,j}$'s to be small primes, the server can actually solve the discrete logarithm problem in the $(\prod_{i=1}^{n} p_{i,b_i})$-order subgroup quite efficiently using, e.g., Pohlig-Hellman and the baby-step/giant-step algorithm. Security against malicious clients rests on a variant of the Φ-hiding assumption of Cachin, Micali, and Stadler [11] – namely, roughly that given an appropriately-generated composite modulus N, it is hard to distinguish which of two numbers P_1 and P_2 divides $\phi(N)$, under the promise that exactly one of them does, where P_1 and P_2 will each be products of n of the primes, one from each pair.

While the "conventional wisdom" about schemes that use OT is that they tend to be inefficient, our scheme is actually very competitive with, though still slightly less efficient than, the most efficient PAKE protocols that fall within the two "basic" paradigms mentioned above. We implemented the scheme and ran experiments over a variety of processors. For 14-bit passwords (e.g., PIN numbers) mapped to 32-bit codewords, our scheme requires 26 (resp. 23) milliseconds of client (resp. server) computation on a 3.20GHz Xeon processor; when used with an arbitrary dictionary of passwords mapped to 64-bit codewords, the scheme requires about twice as much computation for both the client and the server.

Organization. After giving some preliminaries in Section 2, we describe our computational assumption in Section 3. Section 4 describes our scheme. Section 5 gives details of our implementation. Section 6 describes our security model and provides a formal theorem statement regarding security. Finally, Section 7 makes concluding remarks.

2. PRELIMINARIES

Notation Relating to the Set of Passwords. First, we build some notation that will be useful in the description of our scheme. Let Π be the set of passwords in the dictionary, and let $f_\Pi : \Pi \to \{0,1\}^k$ be a function that maps passwords to strings of length k. For example, for PIN numbers, this may be the identity mapping, or for arbitrary dictionaries,

[2] Intuitively, the client must send the confirmation value because after the n OTs, the client knows all $2n$ database entries, and has not yet committed to the password in any way. The confirmation value includes the password as input to the hash because without it the client would be able to complete the protocol without knowing the password by choosing its database entries so that the strings in each pair are identical. In this case, the client would send (and the server would accept) the same confirmation value regardless of the password.

this could be a collision-resistant hash function. Now, let $\alpha : \{0,1\}^k \to \{0,1\}^n$ be an error correcting code with distance d. Thus, $\alpha(f_\Pi(\pi))$ is the n-bit codeword corresponding to password π. We will denote this n-bit string by s_π.

NOTATION RELATING TO THE SET OF PRIMES. Let let $\mathcal{P} = \{p_{1,0}, p_{1,1}, p_{2,0}, p_{2,1}, \ldots, p_{n,0}, p_{n,1}\}$ be a set of $2n$ $(\ell+1)$-bit prime numbers (conceptually) divided into pairs. Let $P = \prod_{p\in\mathcal{P}} p$. For string $s \in \{0,1\}^n$, let $\theta(s)$ equal $\{p_{i,s_i} : 1 \le i \le n\}$, and let $P_s = \prod_{p\in\theta(s)} p$. For each password π, let $P_\pi = P_{s_\pi}$, and let $P_{\bar\pi} = P_{\bar s_\pi}$, where $\bar s$ is the bit-wise complement of s – i.e., $\bar s_i = 1 - s_i$ for $1 \le i \le n$. For a modulus N, define the *prime set* of N, denoted $\mathcal{P}_N$, to be the prime factors of $\phi(N)$ that are in $\mathcal{P}$. We call an element $x \in \mathbb{Z}_N^\star$ a quasi-generator with respect to $\mathcal{P}$ if $\gcd(\phi(N)/|\langle x\rangle|, P) = 1$.

ERROR CORRECTING CODES. To implement our PAKE protocol, we use a binary error correcting code (ECC), and in particular, a BCH code [9, 26] (see [35]). Here we define some terminology related to such codes.

An (n,k,d)-error correcting code maps k-bit strings into n-bit codewords, such that the Hamming distance[3] between any two codewords is at least d. Such a code can correct $\lfloor \frac{d}{2} \rfloor$ errors. In our constructions, however, we will not actually be correcting errors, and in fact, we will not be *decoding* codewords at all. We simply need to *encode* bit strings, and use the fact that any two codewords will have Hamming distance at least d. In Section 4.1, we will discuss the specific parameters used to make our system secure and efficient.

HASH FUNCTIONS. We will use a hash function H for key generation and for producing verification values. For our security proofs, we assume the hash functions behave like black-box perfectly random functions, i.e., random oracles [3]. Although it has been shown that protocols secure in the random oracle model are not necessarily secure when the random oracle is instantiated by a real hash function [12], a proof of security in the random oracle model can provide much greater confidence in the security of a protocol.

SYSTEM MODEL. We assume there is a set of clients *Clients* and a set of servers *Servers*. Our PAKE protocol will be run between a client $C \in$ *Clients* and a server $S \in$ *Servers*. We assume that C does not store any per-server information (such as a server certificate) nor any password data. C simply stores the description of a protocol, and any public parameters of that protocol. C receives, as input, the password π_C and the server S with whom it will perform the PAKE protocol. S stores a password file consisting of a record $\pi_S[C]$ for each C. This record is not restricted to storing the password. It may, for instance, store a function of the password, or some other auxiliary data used for authenticating C in the PAKE protocol. (This assumption is actually for efficiency only. One could always implement the server by computing these values on-the-fly given simply the password.)

We assume that clients and servers may execute the PAKE protocol multiple times with different partners, and we model this by allowing an unlimited number of (possibly concurrent) *instances* of the protocol for each participant. Instance i of client C is denoted Π_i^C, and instance j of server S is denoted Π_j^S.

[3]The Hamming distance between two bit strings a and b is simply the number of 1's in $a \oplus b$.

3. OUR COMPLEXITY ASSUMPTION

THE DECISION SUBGROUP ASSUMPTION. First, we state the complexity assumption on which the security of our PAKE scheme is based: the assumed hardness of the "decision subgroup problem." Very roughly, our assumption is that if N_{π_0} and N_{π_1} are two composite (RSA-type) moduli generated in a "suitable way" such that for $i \in \{0,1\}$, $\theta(s_{\pi_i})$ is the prime set of N_{π_i}, it is hard to decide the value of b from $(\pi_0, \pi_1, \mathcal{P}, N_{\pi_b})$ if b is drawn uniformly at random from $\{0,1\}$. This assumption is a stronger version of the Φ-hiding assumption of Cachin, Micali, and Stadler [11].

We make this assumption more formal as follows. As formalized, this assumption can apply not only to groups based on composite moduli, but to any group with a hidden smooth-order subgroup. Let κ be the security parameter, $\mathcal{P}$ be a set of primes and $\mathcal{P}'$ be a nonempty subset of $\mathcal{P}$. Let $\mathsf{Gen}(1^\kappa, \mathcal{P}, \mathcal{P}')$ be a group generation function with associated functions $f()$ and $f'()$ that, when $\prod_{p\in\mathcal{P}'} p \le 2^{f'(\kappa)}$, randomly generates a group whose order is (1) in the range $[2^{f(\kappa)-1}, 2^{f(\kappa)}]$, (2) divisible by $\prod_{p\in\mathcal{P}'} p$, and (3) not divisible by any $p \in \mathcal{P} \setminus \mathcal{P}'$. (In other words, Gen induces a distribution on such groups.) Then for an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1(1^\kappa, \mathcal{P})$ constructs two non-empty subsets $\mathcal{P}_0, \mathcal{P}_1 \in \mathcal{P}$, $\prod_{p\in\mathcal{P}_0} p \le 2^{f'(\kappa)}$ and $\prod_{p\in\mathcal{P}_1} p \le 2^{f'(\kappa)}$, along with some state s, we say

$$\begin{aligned}&\mathsf{Succ}^{\mathsf{dsga}}_{\mathcal{A}}(\kappa, \mathsf{Gen}, \mathcal{P})\\ &\quad= \Pr((\mathcal{P}_0, \mathcal{P}_1, s) \leftarrow \mathcal{A}_1(1^\kappa, \mathcal{P}); b \xleftarrow{R} \{0,1\};\\ &\qquad G \leftarrow \mathsf{Gen}(1^\kappa, \mathcal{P}, \mathcal{P}_b); \mathcal{A}_2(s, G, \mathcal{P}, \mathcal{P}_0, \mathcal{P}_1) = b),\end{aligned}$$

and

$$\mathsf{Adv}^{\mathsf{dsga}}_{\mathcal{A}}(\kappa, \mathsf{Gen}, \mathcal{P}) = 2\mathsf{Succ}^{\mathsf{dsga}}_{\mathcal{A}}(\kappa, \mathsf{Gen}, \mathcal{P}) - 1.$$

We say $\mathsf{Adv}^{\mathsf{dsga}}(\kappa, \mathsf{Gen}, \mathcal{P}, t)$ is the maximum over all adversaries $\mathcal{A}$ that run in time at most t of $\mathsf{Adv}^{\mathsf{dsga}}_{\mathcal{A}}(\kappa, \mathsf{Gen}, \mathcal{P})$. (Here we assume t is a function of κ.) The $(\mathsf{Gen}, \mathcal{P})$-decision subgroup assumption states that $\mathsf{Adv}^{\mathsf{dsga}}(\kappa, \mathsf{Gen}, \mathcal{P}, t)$ is negligible. For convenience, we will use the notation $\mathsf{Adv}^{\mathsf{dsga}}(t)$ instead of $\mathsf{Adv}^{\mathsf{dsga}}(\kappa, \mathsf{Gen}, \mathcal{P}, t)$, with Gen as specified in Section 4 and $\mathcal{P}$ implicit from the context. For the following discussion, let η denote $f(\kappa)$ and ρ denote $f'(\kappa)$.

One specific case where the $(\mathsf{Gen}, \mathcal{P})$-decision subgroup assumption is thought to be true is when $\mathsf{Gen}(1^\kappa, \mathcal{P}, \mathcal{P}')$ generates multiplicative groups modulo RSA moduli constructed in a specific way, where η is the RSA modulus size corresponding to a given security parameter κ (see [34]), and $\rho < \eta/4$.

For example, consider the following method of generating N, where we will set $\rho + 2\ell < \eta/4$. The server generates a prime Q_1 of the form $2P_\pi R_1 u_1 + 1$ where u_1 is a small integer ($|u_1| = \ell$) and R_1 is a prime of the appropriate size to ensure that $|Q_1| = \eta/2$, which we assume to be greater than the size of any $p \in \mathcal{P}$. The server generates a prime Q_2 of the form $2R_2u_2 + 1$ where, like above, u_2 is a small integer ($|u_1| = \ell$) and R_2 is a prime of the appropriate size to ensure that $|Q_2| = \eta/2$. (We could set $u_1 = u_2 = 1$, in which case Q_2 would be a safe prime – i.e., a prime of the form $2R_2 + 1$ for prime R_2 – but generating Q_1 and Q_2 as above is substantially faster.) The server sets $N = Q_1Q_2$. However, if $N \equiv 1 \bmod p$ for any $p \in \mathcal{P}$, the server rejects N and starts over.

This construction of N guarantees that P_π divides $\phi(N)$,

since it divides $\phi(Q_1)$, and that $\gcd(P_{\bar{\pi}}, \phi(N)) = 1$, since $P_{\bar{\pi}}$ is relatively prime to P_π (by definition), R_1 and R_2 (because they are primes larger than any $p \in \mathcal{P}$), and u_1 and u_2 (because they are both ℓ bits, i.e., smaller than any $p \in \mathcal{P}$). By choosing $\rho + 2\ell < \eta/4$, and choosing Q_1 and Q_2 in this way, we guarantee that there is no obvious way to find a factor of $\phi(N)$ of size at least $\eta/4$. In particular, for any u_1 and u_2 of size ℓ, the size of the factor $P_\pi u_1 u_2$ is less than $\eta/4$. Finally, we reject any $N \equiv 1 \bmod p$ for any $p \in \mathcal{P}$ to avoid any obvious distinguishing attacks, since it will always be the case that $N \not\equiv 1 \bmod p$ for $p \in \theta(s_\pi)$, but not necessarily for $p \in \mathcal{P} \setminus \theta(s_\pi)$.

The constraint that $\rho' = \rho + 2\ell < \eta/4$ is made to prevent attacks based on Coppersmith's method (see [14], [15], [38]), by which one can factor N if one knows a factor $P \geq N^{1/4}$ of $Q_1 - 1$ or $Q_2 - 1$. One security issue regarding this assumption is how much smaller ρ' should be than $\eta/4$ – i.e., how much degradation there is in the performance of Coppersmith's method (which is a latticed-based attack) when η is made bigger in relation to ρ'. Since Coppersmith's method works well (in polynomial time) when $\rho' \geq \eta/4$, one might expect that the algorithm's performance declines only gradually – e.g., so that for $\eta/4 > \rho' \geq \eta/5$ the algorithm (while not polynomial-time) would be only slightly super-polynomial, perhaps because of the inefficiency of lattice reduction. As pointed out in [22], however, this is not true; when η/ρ' is larger than 4, the target vector used in Coppersmith's method (i.e., the one that would help us factor the modulus) is not even the shortest vector in the lattice; thus, even *perfect* lattice reduction algorithms would not, by themselves, make the attack work. These considerations give us confidence that, as long as $\rho' < \eta/4$, and there is no way to easily guess (using $\mathcal{P}'$) the existence of a subgroup of size $\eta/4$ or larger, we can safely instantiate the decision subgroup problem using RSA moduli as outlined above.

To gain further confidence in the decision subgroup assumption, we can consider its vulnerability to generic attacks. Generic attacks have been previously considered in groups of unknown order [18]. The decision subgroup assumption was also considered in this model by [22]. They show, roughly, that as the two distributions associated with the two sets of groups (based respectively on $\mathcal{P}_0, \mathcal{P}_1$) each tend to output a group G whose order is divisible by a large evenly-distributed prime, the decision subgroup problem is hard against generic attacks. In other words, the security of the decision subgroup problem against generic attacks depends less on the order of the subgroup H hidden in G than it does on the distribution of $|G : H|$.

4. OUR PASSWORD AUTHENTICATED KEY EXCHANGE SCHEME

In the Introduction, we presented a high-level description of our PAKE scheme. Here we describe the scheme in detail.

PARAMETERS. We consider Π, f_Π, α and $\mathcal{P}$ to be publicly available "parameters" of our scheme. The scheme also employs two cryptographic hash functions H_0 and H_1, which are modeled in the security proof as independent random oracles.

THE PROTOCOL. Figure 1 gives the protocol. Here we give more details.

Server Initialization. Basically, in this step, the server generates a modulus N with prime set $\theta(s_\pi)$, along with a quasi-generator $x \in \mathbb{Z}_N^\star$, according to the method described in Section 3. Afterwards, it can store these values so that they can be re-used in subsequent sessions with the same client.

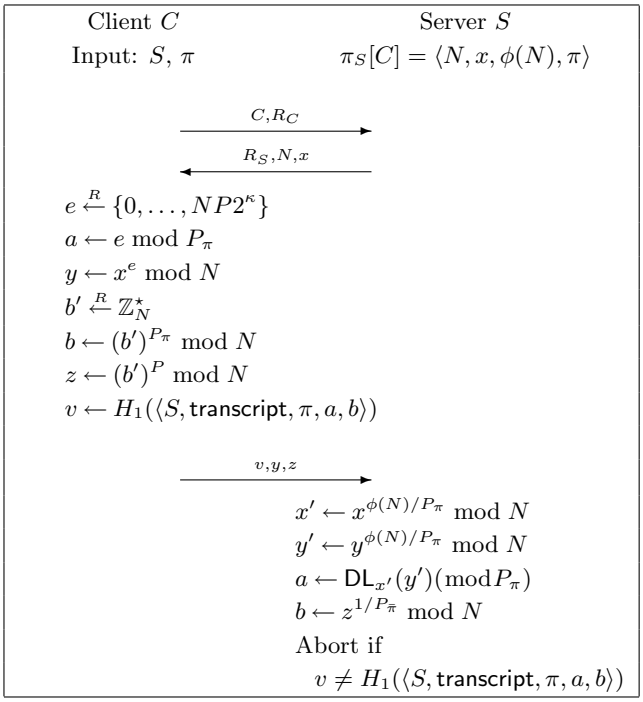

Figure 1: Our PAKE protocol. Session ID is $sid = C \parallel S \parallel R_C \parallel R_S$. Partner ID for C is $pid_C = S$, and partner ID for S is $pid_S = C$. Shared session key is $sk = H_0(\langle S, \mathsf{transcript}, \pi, a, b \rangle)$. Key confirmation is only one-way (client confirms to server). Add one message for mutual key confirmation.

Client First Action. To initiate a session, the client sends its identity C and a nonce $R_C \xleftarrow{R} \{0,1\}^\kappa$ to the server.

Server First Action. The server sends N, x and a nonce $R_S \xleftarrow{R} \{0,1\}^\kappa$ to the client. In [22], it is shown that if the decision subgroup problem is hard, it remains hard when the problem instance also includes a quasi-generator x. Thus (informally) one can say that sending N and x reveals no information that could help in determining the password.

Client Second Action. The client constructs a discrete logarithm problem instance y with answer a (in the subgroup of order P_π), and a root extraction problem instance z (in the entire group) with answer b. The client sends y and z to the server. The client also computes a confirmation value $v = H_1(\langle S, \mathsf{transcript}, \pi, a, b \rangle)$, where $\mathsf{transcript}$ is the transcript of messages sent between the client and server. The client thus proves its knowledge of π explicitly. (The server, on the other hand, has already committed to a password guess implicit in its modulus N.)

Server Second Action. The server solves the problems y and z to get answers a and b, respectively. The root extraction is straightforward since the server actually knows the group order. The discrete logarithm can also be computed

efficiently, since the server can solve it within a (smooth) subgroup in $\mathbb{Z}_N^\star$ of order P_π. Computing discrete logs in this subgroup is easy using Pohlig-Hellman, since the prime divisiors of P_π are all small – e.g., $\ell + 1$ bits, where the number ℓ is quite small (e.g., 10). (See Section 4.1 for more details on how to choose ℓ.)

The server then checks if the value v received from the client is equal to $H_1(\langle S, \mathsf{transcript}, \pi, a, b\rangle)$. If not, then it aborts.

Joint Session Key. The joint session key is equal to $sk = H_0(\langle S, \mathsf{transcript}, \pi, a, b\rangle)$. Intuitively, if N is constructed using the primes corresponding to the correct password (i.e., $\theta(s_\pi) = \mathcal{P}_N$) then (a, b) can be computed from (y, z). (This is shown explicitly below.) However, if N were constructed with an "incorrect" set of primes, then for every "incorrect" prime, the possible (a, b) values the client could have used to compute (y, z) would increase by a multiplicative factor (equal to that prime). Then, as shown in Section 6, because of the error correcting code, the number of possibilities for (a, b) for a given (y, z) will be huge unless the client used the (at most) one password π with $\theta(s_\pi)$ "closest" to $\mathcal{P}_N$.

Correctness. It suffices to show that the (a, b) values the server recovers are identical to the respective values the client generated. For b this is trivial as $\gcd(P_{\bar{\pi}}, \phi(N)) = 1$ yields

$$z^{1/P_{\bar{\pi}}} = (b')^{P/P_{\bar{\pi}}} = (b')^{P_\pi} = b \bmod N.$$

To see that the a values are the same, we first write e as $a + mP_\pi$ for some integer m. Now,

$$\begin{aligned} y' &= y^{\phi(N)/P_\pi} = x^{e\phi(N)/P_\pi} = x^{(a+mP_\pi)\phi(N)/P_\pi} \\ &= (x^{(\phi(N)/P_\pi)})^a \cdot x^{m\phi(N)} = (x')^a \bmod N, \end{aligned}$$

where x' and y' are each in the P_π-order subgroup of $\mathbb{Z}_N^\star$.

4.1 Instantiating the Error-Correcting Code

In order for our scheme to be both efficient and secure, the error-correcting code, and specifically, the (n, k, d) parameters, must be chosen carefully. Recall that n is the number of bits of a codeword, k is the number of bits of the input (i.e., the password representation), and d is the distance of the encoding (the minimum Hamming distance between any two codewords). Also of importance is the parameter ℓ, since all primes in the hidden smooth subgroup of $\mathbb{Z}_N^\star$ must be of length at least $\ell + 1$.

For flexibility, we would like k to be as large as possible, since we would like to be able to handle large dictionaries. (See Section 4.2 for further discussion of this.) To achieve at least the security of 4-digit PINs, we need $k \geq 14$.

For efficiency, we would like $n(\ell+1)$ to be as small as possible (assuming the n primes used for the smooth subgroup are all $\ell + 1$ bits, else this value would be larger), since this affects the size of the modulus, as discussed in Section 3. In particular, we need $|N|$ to be at least four times this value, plus a comfortable margin, to avoid any known attacks. Alternatively, we could generalize the protocol to use multiple moduli, splitting the n bits of the codeword (along with their corresponding primes) among these different moduli, and simply requiring that the size of each modulus is at least four times $n'(\ell+1)$, where n' is the number of bits assigned to that modulus. (Since modular exponentiations take time cubic in the size of the modulus, this alternative can often be more efficient.)

(n, k, d)	ℓ	$\lvert N\rvert$
$(64, 39, 10)$	8	1536(2)
$(64, 39, 10)$	8	1024(3)
$(32, 16, 8)$	10	1536
$(32, 16, 8)$	10	1024(2)
$(64, 45, 8)$	10	1536(2)
$(64, 45, 8)$	10	1280(3)
$(64, 45, 8)$	10	1024(4)
$(32, 21, 6)$	14	1280(2)
$(64, 51, 6)$	14	1536(3)
$(64, 51, 6)$	14	1280(4)

Figure 2: Code parameters for $\kappa = 80$, along with the sizes of primes and moduli.

For security against an adversary posing as a client, we need to make sure the adversary cannot break the decision subgroup assumption, and thus, as discussed above, $|N| \geq \eta$. (For $\kappa = 80$, $\eta \geq 1024$ (see [34]).)

For security against an adversary posing as a server, we need to make sure the adversary cannot construct a modulus that allows it to test the correctness/incorrectness of two candidate passwords simultaneously. We analyze the security as follows.

By our construction, each bit of the codeword corresponds to the inclusion of one $(\ell + 1)$-bit prime and exclusion of a different $(\ell + 1)$-bit prime. Since the distance between codewords is d, this implies that two codewords differ in the inclusion/exclusion of $2d$ primes. This means that any modulus chosen by the adversary is within less than d inclusion/exclusions of at most one codeword. As we will see in the proof of security, if the adversary wants to test for the correctness/incorrectness of a password, for every incorrectly guessed inclusion or exclusion, at least ℓ-bits must be guessed by the adversary. Thus to test two passwords, one password will be at least d inclusions/exclusions away, and the adversary must guess $d\ell$ bits. To achieve κ bits of security, we need $d\ell \geq \kappa$.

In summary, we need to obtain an (n, k, d)-ECC and a value ℓ such that

- $d\ell \geq \kappa$;
- $n(\ell + 1)$ is minimized;
- $k \geq 14$ is maximized;
- $|N| > \max\{z(\kappa), 4n(\ell + 1)\}$.

Figure 2 gives parameters for known codes (which are all BCH codes with a parity bit added), corresponding ℓ values to satisfy our first requirement, along with the size of the modulus (or moduli) necessary for security at the $\kappa = 80$ level. Multiple moduli are indicated by a number in parentheses after the size of the modulus. The size of the moduli have been rounded to a multiple of 256. We have examined other parameters, but they give worse performance.

For PIN security, the $(32, 16, 8)$-ECC with $\ell = 10$ and $|N| = 1536$ seems to be a good choice. For security with ar-

bitrary dictionaries (discussed in Section 4.2), the $(64, 45, 8)$-ECC with $\ell = 10$ and two 1536-bit moduli seems to be a good choice.

4.2 Arbitrary Dictionaries

Now we consider the function f_Π which maps passwords to bit strings. If there is a one-to-one mapping, then we obtain full security against online dictionary attacks, meaning that for q online authentication attempts and dictionary $\mathcal{D}$, the probability of success is at most negligibly more than $q/|\mathcal{D}|$. For instance, when 4-digit PIN numbers are used, f_Π can simply map these numbers into length 14 bit strings corresponding to their numeric value between 0 and 9999. On the other hand, given a dictionary containing arbitrary length strings, there may not be a trivial mapping. Instead, we rely on a hash function to hash strings to k-bits. Then these k-bits can be used as input to the error-correcting code.

Unfortunately, collision resistant hash functions require a large output size. For instance, SHA-1 outputs 160 bits. A version of SHA-1 truncated to the number of bits we need for an efficient code, say $k = 45$, will not necessarily be collision resistant. Thus our security may be reduced. Here we give some analysis to show that we still maintain a reasonably high level of security. Basically, the bit security will be on the order of the minimum of k and the log of the dictionary size, which is as good as one could hope for.[4]

First recall the Chernoff bound. Given n experiments with probability p of success, and X as a random variable denoting the number of successes,

$$\Pr(X > (1+\gamma)np) \leq \left(\frac{e^\gamma}{(1+\gamma)^{1+\gamma}}\right)^{np}.$$

In our scenario we may assume a dictionary of size at most 2^a and a random function f_Π with k-bit output, and an experiment (i, π) is whether $f_\Pi(\pi) = i$. This is essentially a balls-in-bins argument with the passwords representing the balls, and the 2^k outputs representing the bins. Then $n = 2^a$ and $p = 2^{-k}$. For convenience, let $\gamma = c2^{k-a} - 1$, so that we are bounding the probability that a bin has greater than c balls. Then we have

$$\Pr(X > c) \leq e^{-2^{a-k}} \left(\frac{e}{c2^{k-a}}\right)^c \leq \left(\frac{e}{c2^{k-a}}\right)^c.$$

We can bound the probability that any bin has greater than c balls (i.e., the probability that any output is mapped to by more than c passwords) by the value $2^k \Pr(X > c)$. We would like this value to be at most $c2^{-a}$, since that is the eventual probability that one k-bit output will correspond to the correct password, assuming that at most c passwords are mapped to any given output. Thus we need

$$c2^{-a} \geq 2^k \left(\frac{e}{c2^{k-a}}\right)^c,$$

which can be reduced to

$$c(\log c(1 + c^{-1}) + k - a) \geq a + k + c \log e.$$

[4]Note that when the dictionary size is much larger than 2^k, by a strict interpretation one could say that we do not have a secure PAKE protocol. However, if one considers that we are essentially changing the password space from $\mathcal{D}$ to the set of k-bit strings, then under the new password space, one could say we have a secure PAKE protocol. Of course, we recommend this only for large k, such as $k > 30$.

The three BCH codes in the table above that could be used for arbitrary dictionaries have either $k = 39$, $k = 45$, or $k = 51$. As an example, take $k = 45$, and a dictionary of size 2^{35} (i.e., $a = 35$, or roughly 32 billion). Then one needs $c \geq 7$, to make sure that the probability that an output word is mapped to by more than 7 passwords is at most $7/2^{35}$, and thus one achieves about 32-bit security.

In Figure 3 we provide the bit security achieved for each k value, ranged over reasonable dictionary sizes. Notice that in general, the bit security is reduced by a small constant for smaller values of a, but as a approaches k, the bit security levels off, and eventually becomes about $k - \log e$ (i.e., roughly $k - 1.44$).

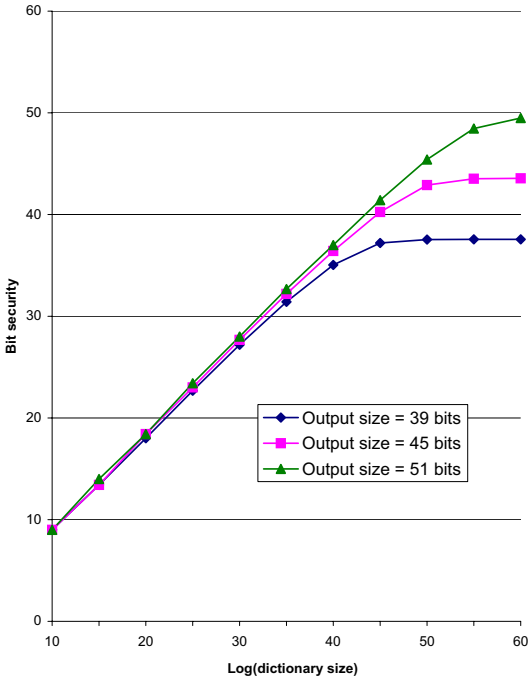

Figure 3: Bit security using different parameters, for different dictionary sizes

5. IMPLEMENTATION

To test the efficiency of our protocol, we implemented a demonstration version that runs both the client and server sides of the protocol on a single machine (thus disregarding communication costs). This implementation was written in C using the GNU MP library for the multi-precision arithmetic. The OpenSSL library was used for the cryptographic hash functions and cryptographic random number generation.

We obtained the following results using a $(32, 16, 8)$-ECC with $\ell = 10$ and a 1536-bit modulus. As discussed above, this would be suitable for the case of 4-digit PIN numbers. Although we have not run experiments for a $(64, 45, 8)$-ECC with two 1536-moduli, which could handle arbitrary size dictionaries, it is clear that it would essentially double the computation cost for both the client and server.

To increase performance we used the following optimizations.

- The server performed exponentiations over the composite modulus N using the knowledge of the two prime factors of N and the Chinese-remainder theorem (as is done often for RSA decryption).
- The smooth subgroup was placed as a subgroup of only one of the factors of N, and thus all computations involved in computing discrete logs could be performed modulo that one factor, instead of N.
- A recursive algorithm was used for determining the 32 small DL problems from the large DL problem of computing $a \equiv \mathsf{DL}_{x'}(y')(\mathrm{mod} P_\pi)$. This is described below.
- The baby-step/giant-step algorithm was used for computing the discrete logarithms.

We performed twenty-five runs of the protocol execution on each machine, and averaged the computation times.

$(32, 16, 8)$-ECC, $\ell = 10$, $\lvert N\rvert = 1536$		
	Client (msec)	Server (msec)
Xeon 3.20GHz	26	23
Pentium M 2.00GHz	46	45
Celeron 2.66GHz	100	86

The major computation cost for the client is the exponentiation over a 1536-bit modulus with an exponent that is more than 1536 bits. The major computation cost for the server is roughly evenly split between determining the 32 DL problems, and then solving those problems.

These performance numbers indicate that a server with a Xeon processor running at 3.2GHz could process more than 40 authentication attempts per second. Obviously, as with any cryptographic protocol, denial-of-service (DoS) attacks should be considered. Fortunately, the server need not perform any cryptographic operations until it receives its second message, so attacks with spoofed IP addresses are of little concern. However, attacks from real IP addresses could still occur, and thus adequate protections (such as client-side puzzles [20]) should be built in to mitigate the effects of these attacks.

Recursive Algorithm for Computing Small DL Problems. The server has computed two values x' and y', and wants to compute $\mathsf{DL}_{x'}(y') \bmod P_\pi$, where P_π is the product of n $(\ell+1)$-bit primes. (In our case above, $n = 32$ and $\ell = 10$.) To do this efficiently, one can compute the discrete logarithm modulo p_i, for each prime p_i that divides P_π. However the order of x' is P_π, and thus one should compute $x'_i \leftarrow (x')^{P_\pi/p_i} \bmod N$ and $y'_i \leftarrow (y')^{P_\pi/p_i} \bmod N$ (i.e., compute x'_i and y'_i by raising x' and y' respectively by the product of all primes *except* p_i), so that the order of x'_i is p_i, and one can compute $\mathsf{DL}_{x'_i}(y'_i)$ using a standard method (such as baby-step/giant-step) taking $O(\sqrt{p_i})$ steps.

So the object is to efficiently compute x'_i and y'_i for each i. The naive way requires n exponentiations over base N with exponents of size roughly $(n-1)/(\ell+1)$ (or about 32 exponentiations with each exponent about $31 \cdot 11 = 341$ bits in our case). A much faster way is to compute the x'_i and y'_i values using a recursive algorithm. We focus on x'_i values; the y'_i values are similar. Split the product of primes into two products, the first half and the second half. Raise x' to each product, and call these x'_{low} and x'_{high}. Then recursively solve the problem of computing the first half of the x'_i values starting with x'_{high}, and the second half of the x'_i values starting with x'_{low}. Note that each x'_i value will eventually be x' raised to all p_j values where $j \neq i$.

6. SECURITY MODEL AND THEOREM

We describe our security model and provide a formal theorem statement regarding security for any given choice of system parameters κ, n, d, k, and ℓ.

Security Model. For our proofs of security we use the model of [2] (which builds on [4] and [5], and is also used by [31]).[5] This model is designed for the problem of authenticated key exchange (ake) between two parties, a client and a server, that share a secret. The goal is for them to engage in a protocol such that after the protocol is completed, they each hold a session key that is known to nobody but the two of them.

In the following, we will assume some familiarity with the model of [2].

Protocol participants. Let ID be a nonempty set of principals, each of which is either a client or a server. Thus $ID \stackrel{\text{def}}{=} \mathit{Clients} \cup \mathit{Servers}$, where $\mathit{Clients}$ and $\mathit{Servers}$ are finite, disjoint, nonempty sets. We assume each principal $U \in ID$ is labeled by a string, and we simply use U to denote this string.

Each client $C \in \mathit{Clients}$ has a secret password π_C and each server $S \in \mathit{Servers}$ has a vector $\pi_S = \langle \pi_S[C] \rangle_{C \in \mathit{Clients}}$. Entry $\pi_S[C]$ is the *password record*. (Note that $\pi_S[C]$ may contain more information than simply π_C.) Let $\mathit{Password}_C$ be a (possibly small) set from which passwords for client C are selected. We will assume that $\pi_C \stackrel{R}{\leftarrow} \mathit{Password}_C$ (but our results easily extend to other password distributions). Clients and servers are modeled as probabilistic poly-time algorithms with an input tape and an output tape.

Execution of the protocol. A protocol P is an algorithm that determines how principals behave in response to inputs from their environment. In the real world, each principal is able to execute P multiple times with different partners, and we model this by allowing unlimited number of *instances* of each principal. Instance i of principal $U \in ID$ is denoted Π_i^U.

To describe the security of the protocol, we assume there is an adversary $\mathcal{A}$ that has complete control over the environment (mainly, the network), and thus provides the inputs to instances of principals. Formally, the adversary is a probabilistic algorithm with a distinguished query tape. Queries written to this tape are responded to by principals according to P; the allowed queries are formally defined in [2] and summarized here:

Send $(U,\ i,\ M)$**:** sends message M to instance Π_i^U. The instance computes what the protocol says to, state is updated, and the output of the computation is given to $\mathcal{A}$. If this query causes Π_i^U to accept or terminate, this will also be shown to $\mathcal{A}$.[6] To initiate a session between client C and server S, the adversary should send

[5] However, due to space limitations, we will not consider the issues of providing explicit authentication or dealing with corruption of parties.

[6] Recall that accepting implies generating a triple

a message containing the server name S to an unused instance of C.

Execute $(C,\ i,\ S,\ j)$**:** executes P to completion between Π_i^C (where $C \in$ *Clients*) and Π_j^S (where $S \in$ *Servers*), and outputs the transcript of the execution. This query captures the intuition of a passive adversary who simply eavesdrops on the execution of P.

Reveal $(U,\ i)$**:** outputs the session key held by Π_i^U.

Test $(U,\ i)$**:** causes Π_i^U to flip a bit b. If $b = 1$ the session key sk_U^i is output; otherwise, a string is drawn uniformly from the space of session keys and output. A Test query may be asked at any time during the execution of P, but may only be asked once.

Partnering. A client or server instance that accepts holds a partner-id *pid*, session-id *sid*, and a session key *sk*. Then instances Π_i^C (with $C \in$ *Clients*) and Π_j^S (with $S \in$ *Servers*) are said to be *partnered* if both accept, they hold (pid, sid, sk) and (pid', sid', sk'), respectively, with $pid = S$, $pid' = C$, $sid = sid'$, and $sk = sk'$, and no other instance accepts with session-id equal to *sid*.

Freshness. An instance Π_i^U is fresh unless either (1) a Reveal $(U,\ i)$ query occurs, or (2) a Reveal $(U',\ j)$ query occurs where $\Pi_{U'}^j$ is the partner of Π_i^U.

Advantage of the adversary. We now formally define the authenticated key exchange (ake) advantage of the adversary against protocol P. Let $\mathsf{Succ}_P^{\mathrm{ake}}(\mathcal{A})$ be the event that $\mathcal{A}$ makes a single Test query directed to some fresh instance Π_i^U that has terminated, and eventually outputs a bit b', where $b' = b$ for the bit b that was selected in the Test query. The ake advantage of $\mathcal{A}$ attacking P is defined to be

$$\mathsf{Adv}_P^{\mathrm{ake}}(\mathcal{A}) \stackrel{\mathrm{def}}{=} 2\Pr\left[\mathsf{Succ}_P^{\mathrm{ake}}(\mathcal{A})\right] - 1.$$

The following fact is easily verified.

FACT 1.

$$\Pr(\mathsf{Succ}_P^{\mathrm{ake}}(\mathcal{A})) = \Pr(\mathsf{Succ}_{P'}^{\mathrm{ake}}(\mathcal{A})) + \epsilon$$
$$\iff \qquad \mathsf{Adv}_P^{\mathrm{ake}}(\mathcal{A}) = \mathsf{Adv}_{P'}^{\mathrm{ake}}(\mathcal{A}) + 2\epsilon.$$

SECURITY THEOREM. Having provided the model, we now precisely state our security theorem. To precisely quantify the concrete security loss in the reduction, the theorem statement refers to three quantities: $t_{\mathsf{prot}}, t_{\mathsf{samp}}, t_{\mathsf{exp}}$. These are respectively the time required for a single protocol execution, the time to sample a random element of the group G over which the protocol is executed, and the time to perform an exponentiation in that group.

THEOREM 1. *Let P be the protocol described in Figure 1, and with a password dictionary $\mathcal{D}$ and f_Π a one-to-one mapping into $\{0,1\}^k$. Fix an adversary $\mathcal{A}$ that runs in time t, and makes $n_{\mathrm{se}}, n_{\mathrm{ex}}, n_{\mathrm{re}}$ queries of type* Send, Execute, Reveal, *respectively, and n_{ro} queries to the random oracles. Then for $t' = O(t + (n_{\mathrm{se}} + n_{\mathrm{ex}})t_{\mathsf{prot}} + t_{\mathsf{samp}} + n \cdot t_{\mathsf{exp}})$:*

$$\mathsf{Adv}_P^{\text{ake-nfs}}(\mathcal{A}) = \frac{n_{\mathrm{se}}}{|\mathcal{D}|} + O\left(n_{\mathrm{se}}\mathsf{Adv}^{\mathsf{dsga}}(t') + \frac{n_{\mathrm{ro}}}{2^{d\ell}} + \frac{(n_{\mathrm{ro}} + n_{\mathrm{se}} + n_{\mathrm{ex}})^2}{2^{2\kappa}}\right).$$

We remark that if f_Π is a random mapping to $\{0,1\}^k$, then the first term becomes $(n_{\mathrm{se}} + 1)2^{-\delta}$, where δ is the bit security computed in Section 4.2, and the extra additive $2^{-\delta}$ comes from the probability that more than a $2^{-\delta}$ fraction of passwords map to any given k-bit string.

PROOF. (Sketch) We first introduce a series of protocols $P_0, P_1, \ldots, P_6$ related to P, with $P_0 = P$. In P_6, $\mathcal{A}$ will be reduced to a simple online guessing attack that will admit a straightforward analysis.

P_0 The original protocol P.

P_1 If honest parties randomly choose R_C or R_S values seen previously in the execution of the protocol, the protocol halts and the adversary fails.

P_2 The protocol answers Send and Execute queries without making any random oracle queries. Subsequent random oracle queries by the adversary are backpatched, as much as possible, to be consistent with the responses to the Send and Execute queries. Note that the server will abort on non-matching sessions if it receives a v value for which the adversary has not made a correct random oracle $H_1(\cdot)$ query. Also the simulation halts and the adversary succeeds if it makes a correct random oracle query to determine the session key or verification value of a party. (This is a standard technique for proofs involving random oracles.)

P_3 If an "almost correct" password guess is made against a server instance (determined by a verification value v sent to the server instance, where v is equal to the output of an $H_1(\cdot)$ query with a correct password, regardless of the a and b values), the protocol halts and the adversary automatically succeeds.

P_4 Server instances use a dummy password to compute the modulus N for each client.

P_5 If an $H_0(\cdot)$ or $H_1(\cdot)$ query is made, it is not checked for consistency against Execute queries. That is, instead of aborting the simulation, a random response is returned from the query.

P_6 If the adversary makes two password guesses against the same client instance, the protocol halts and the adversary fails.

For each i from 1 to 6, we need to prove that the advantage of $\mathcal{A}$ attacking protocol P_{i-1} is at most negligibly more than the advantage of $\mathcal{A}$ attacking protocol P_i. The proofs concerning the first three "transitions" are relatively straightforward. For the transition from P_3 to P_4, we use the hardness of the decision subgroup problem. For the transition from P_4 to P_5, and from P_5 to P_6, we use the following claim.

CLAIM 1. *Take a modulus N and $x \in \mathbb{Z}_N^\star$. Take a password π. Let $\mathcal{P}'$ be the set of primes that are in either $\mathcal{P}_N$ or $\theta(s_\pi)$, but not both, and let $d = |P'|$. Then for any (y, z) produced by the client, the probability that (a, b) was produced along with (y, z) is at most $(1 + 2^{-\kappa})2^{-d\ell}$.*

Roughly speaking, this claim implies that if the server chooses its modulus N such that the set difference of $\mathcal{P}_N$ and $\theta(s_\pi)$ has large cardinality, then the server cannot feasibly retrieve the "correct" value of (a, b) from (y, z) because a huge number (about $2^{d\ell}$) of pairs (a, b) *all* correspond to *same* (y, z).

(pid, sid, sk), terminating implies accepting and no more messages will be output. To indicate the protocol not sending any more messages, but not terminating, *state* is set to DONE, but *term* is set to FALSE.

PROOF. For given (N, x, π), the value of a is independent of the values of b and z, while the value of b is independent of the values of a and y; thus, $\Pr[(a,b)|(y,z)] = \Pr[a|y]\Pr[b|z]$.

First, we bound $\Pr[a|y]$. Let $\mathcal{P}_{\pi-N} = \theta(s_\pi) \setminus \mathcal{P}_N$ (i.e., the set of primes in $\theta(s_\pi)$ but not $\mathcal{P}_N$), and let $P_{\pi-N}$ be the product of the primes in $\mathcal{P}_{\pi-N}$. Intuitively, since $\phi(N)$ is relatively prime to $P_{\pi-N}$, the value of y (which constrains the possible values of $e \bmod \phi(N)$) does not really constrain the value of $e \bmod P_{\pi-N}$ when e is taken from a large-enough interval, and consequently $a \leftarrow e \bmod P_\pi$ can assume $P_{\pi-N}$ different values with essentially equal probability. More formally, let $|\langle x\rangle|$ be the order of x modulo N. Let $\mathcal{E}_y = \{e_y \in \{0,\ldots,NP2^\kappa\} : y = x^{e_y} \bmod N\}$. Then, $\Pr[a|y] = \Pr[e_y \equiv a \bmod P_\pi | e_y \in \mathcal{E}_y] \leq \Pr[e_y \equiv a \bmod P_{\pi-N} | e_y \in \mathcal{E}_y]$. Since the values of $\mathcal{E}_y$ form an arithmetic progression with difference $|\langle x\rangle|$, and since $|\langle x\rangle|$ is relatively prime to $P_{\pi-N}$, the values of $\mathcal{E}_y$ cycle (with period $P_{\pi-N}$) repeatedly through all residues modulo $P_{\pi-N}$, and we obtain $\Pr[e_y \equiv a \bmod P_{\pi-N} | e_y \in \mathcal{E}_y] \leq \lceil |\mathcal{E}_y|/P_{\pi-N}\rceil / |\mathcal{E}_y| \leq 1/P_{\pi-N} + 1/|\mathcal{E}_y| \leq 1/P_{\pi-N} + 1/P2^\kappa$, where the final equality follows from $|\mathcal{E}_y| \geq \lfloor NP2^\kappa / |\langle x\rangle| \rfloor$ and $|\langle x\rangle| < N$. Thus, $\Pr[a|y] \leq 1/P_{\pi-N} + 1/P2^\kappa \leq (1+2^{-\kappa})/P_{\pi-N}$.

Next, we bound $\Pr[b|z]$. Let $\mathcal{B}_{all} = \{b' \in \mathbb{Z}_N^* : z = (b')^P \bmod N\}$. Let $b_0 \in \mathbb{Z}_N^*$ be such that the cardinality of the set $\mathcal{B}_0 = \{b' \in \mathbb{Z}_N^* : (b')^{P_\pi} = b_0 \bmod N \wedge z = b_0^{P_{\bar\pi}} \bmod N\}$ is maximized. Notice that $\Pr[b|z] \leq \Pr[b_0|z] = |\mathcal{B}_0|/|\mathcal{B}_{all}|$. Let $u \in \mathbb{Z}_N^*$ be a primitive P_N-th root of unity modulo N. Let $\mathcal{P}_{N-\pi} = \mathcal{P}_N \setminus \theta(s_\pi)$ (i.e., the set of primes in $\mathcal{P}_N$ but not $\theta(s_\pi)$), and let $P_{N-\pi}$ be the product of the primes in $\mathcal{P}_{N-\pi}$. Then, for $t \in [0, P_{N-\pi} - 1]$, the values $b_t = b_0 u^{tP_\pi} \bmod N$ are distinct and thus the sets $\mathcal{B}_t = \{b' \in \mathbb{Z}_N^* : (b')^{P_\pi} = b_t \bmod N \wedge z = b_t^{P_{\bar\pi}} \bmod N\}$ are disjoint. Moreover, $|\mathcal{B}_t| = |\mathcal{B}_0|$, since $\mathcal{B}_t = \{b' : b'/u^t \in \mathcal{B}_0\}$. Thus, $|\mathcal{B}_{all}| \geq P_{N-\pi}|\mathcal{B}_0|$ and $\Pr[b|z] \leq 1/P_{N-\pi}$.

Finally, we get that $\Pr[a|y]\Pr[b|z] \leq (1+2^{-\kappa})/P_{\pi-N}P_{N-\pi} \leq (1+2^{-\kappa})2^{-d\ell}$, since the primes in $\mathcal{P}$ are greater than or equal to 2^ℓ. □

Finally, in P_6, it is easy to see that the adversary can make at most one password guess against each client and server instance. This adds $\frac{n_{\mathrm{se}}}{|\mathcal{D}|}$ to the probability that the adversary succeeds. If the adversary does not guess the password, then it is straightforward to show that the view of the adversary is independent of each fresh session key, and thus the probability of success from a Test query is exactly $\frac{1}{2}$.

Therefore $\mathsf{Adv}^{\mathrm{ake}}_{P_6}(\mathcal{A}) \leq \frac{n_{\mathrm{se}}}{|\mathcal{D}|}$. The theorem follows from this by adding in the advantage gained by the adversary between each protocol P_{i-1} and its successor protocol P_i.

7. CONCLUSIONS AND FUTURE WORK

We presented a password authenticated key exchange protocol that is provably secure against offline dictionary attacks in the random oracle model (based on the decision subgroup assumption). Our approach involved using the password to construct a multiplicative group of partially smooth order; this appears to be a new paradigm in the design of password-authenticated key exchange protocols (which typically involve using the password to encrypt the messages in a standard key exchange protocol or using the password to choose the parameters of a standard key exchange protocol). Our scheme has some similarity to a recent private information retrieval scheme [22].

8. REFERENCES

[1] M. Abdalla and D. Pointcheval. Simple password-based encrypted key exchange protocols. In *Proc. of CT-RSA 2005*, LNCS 3376, pp. 191–208. Springer, 2005.

[2] M. Bellare, D. Pointcheval and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Proc. of Eurocrypt 2000*, LNCS 1807, pp. 139–155. Springer, 2000.

[3] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. of ACM CCS 1993*, pp. 62–73. ACM, 1993.

[4] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proc. of Crypto 1993*, LNCS 773, pp. 232–249. Springer, 1994.

[5] M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. In *Proc. of STOC 1995*, pp. 57–66. ACM, 1995.

[6] S. M. Bellovin and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In *IEEE Symposium on Research in Security and Privacy*, pp. 72–84. IEEE, 1992.

[7] S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. In *ACM CCS 1993*, pp. 244–250. ACM, 1993.

[8] D. Boneh. The decision Diffie-Hellman problem. In *Proc. of ANTS-III*, LNCS 1423, pp. 48–63. Springer, 1998.

[9] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Inf. Control*, 3, pp. 68–79, 1960.

[10] V. Boyko, P. MacKenzie and S. Patel. Provably secure password authentication and key exchange using Diffie-Hellman. In *Proc. of Eurocrypt 2000*, LNCS 1807, pp. 156–171. Springer, 2000.

[11] C. Cachin, S. Micali, M. Stadler, Computational private information retrieval with polylogarithmic communication. In *Proc. of Eurocrypt 1999*, LNCS 1592, pp. 402–414, Springer, 1999.

[12] R. Canetti, O. Goldreich and S. Halevi. The random oracle methodology, rivisited. In *JACM*, vol. 51, no. 4, pp. 557-594. ACM, 2004. (Preliminiary version appeared in STOC 1998, pp. 209–218.)

[13] R. Canetti, S. Halevi, J. Katz, Y. Lindell and P. MacKenzie. Universally-composable password-based key exchange. In *Proc. of Eurocrypt 2005*, LNCS 3494, pp. 404–421. Springer, 2005.

[14] D. Coppersmith, Finding a small root of a bivariate integer equation; factoring with high bits known. In *Proc. of Eurocrypt 1996*, LNCS 1070, pp. 178–189. Springer, 1996.

[15] D. Coppersmith, Finding a small root of a univariate modular equation. In *Proc. of Eurocrypt 1996*, LNCS 1070, pp. 155–165. Springer, 1996.

[16] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Proc. of Crypto 1998*, LNCS 1462, pp. 13–25. Springer, 1998.

[17] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for chosen ciphertext secure public key

encryption. In *Proc. of Eurocrypt 2002*, LNCS 2332, pp. 45–64. Springer, 2002.

[18] I. Damgard and M. Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In *Proc. of Eurocrypt 2002*, LNCS 2332, pp. 256–271. Springer, 2002.

[19] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Info. Theory*, 22(6): 644–654, 1976.

[20] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Proc. of Crypto 1992*, LNCS 740, pp. 139–147. Springer, 1993.

[21] R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In *Proc. of Eurocrypt 2003*, LNCS 2656, pp. 524–543. Springer, 2003.

[22] C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In *Proc. of ICALP 2005*, pp. 803-814. Springer, 2005.

[23] O. Goldreich and Y. Lindell. Session-key generation using human passwords only. In *Proc. of Crypto 2001*, LNCS 2139, pp. 408–432. Springer, 2001.

[24] L. Gong. Optimal authentication protocols resistant to password guessing attacks. In *IEEE Computer Security Foundations Workshop*, pages 24–29. IEEE, 1995.

[25] L. Gong, T. M. A. Lomas, R. M. Needham and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. In *IEEE Journal on Selected Areas in Communications*, 11(5): 648–656, June 1993.

[26] A. Hocquenghem. Codes corecteurs d'erreurs. *Chiffres*, 2, pp. 147–156, 1959.

[27] IEEE Standard 1363-2000, Standard specifications for public key cryptography, 2000.

[28] D. Jablon. Strong password-only authenticated key exchange. In *ACM Computer Communication Review, ACM SIGCOMM*, 26(5): 5–20, 1996.

[29] D. Jablon. Extended password key exchange protocols immune to dictionary attack. In *WETICE'97 Workshop on Enterprise Security*, pp. 248-255. IEEE, 1997.

[30] S. Jiang and G. Gong. Password based key exchange with mutual authentication. In *Proc. of SAC 2004*, LNCS 3357, pp. 267–279. Springer, 2004.

[31] J. Katz, R. Ostrovsky and M. Yung. Practical password-authenticated key exchange provably secure under standard assumptions. In *Proc. of Eurocrypt 2001*, LNCS 2045, pp. 475–494. Springer, 2001.

[32] C. Kaufmann and R. Perlman. PDM: a new strong password-based protocol. In *Usenix Security Symposium*, 2001.

[33] T. Kwon. Authentication and key agreement via memorable passwords. In *Proc. of NDSS 2001*. ISOC, 2001.

[34] A. Lenstra and E. Verheul. Selecting cryptographic key sizes. In *Journal of Cryptology*, 14(4): 255293, 2001.

[35] S. Lin and D. J. Costello, Jr. Error control coding: fundamentals and applications. Prentice Hall, Englewood Cliffs, NJ, 1983.

[36] S. Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Proc. of Security Protocols Workshop*, LNCS 1361, pp. 79-90. Springer, 1998.

[37] P. MacKenzie, S. Patel and R. Swaminathan. Password authenticated key exchange based on RSA. In *Proc. of Asiacrypt 2000*, LNCS 1976, pp. 599–613. Springer, 2000.

[38] A. May. A tool kit for finding small roots of bivariate polynomials over the integers. In *Proc. of Eurocrypt 2005*, LNCS 3494, pp. 251–267. Springer, 2005.

[39] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. of STOC 1999*, pp. 245–254. ACM, 1999.

[40] National Institute of Standards and Technology (NIST). Announcing the Secure Hash Standard, FIPS 180-1, U.S. Department of Commerce, April, 1995.

[41] S. Patel. Number theoretic attacks on secure password schemes. In *Proc. of the IEEE Sym. on Research in Security and Privacy*, pp. 236–247, 1997.

[42] M. O. Rabin. How to exchange secrets by oblivious transfer. Unpublished manuscript, 1981.

[43] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signature and public key cryptosystems. In *Communications of the ACM*, 21:120–126, 1978.

[44] M. Steiner, G. Tsudik, and M. Waidner. Refinement and extension of encrypted key exchange. In *Operating System Review*, 29:22–30. ACM, 1995.

[45] T. Wu. The secure remote password protocol. In *Proc. of NDSS 1998*, pp. 97–111. ISOC, 1998.

[46] T. Wu. A real-world analysis of Kerberos password security. In *Proc. of NDSS 1999*, ISOC, 1999.

[47] M. Zhang. New approaches to password authenticated key exchange based on RSA. In *Proc. of Asiacrypt 2004*, LNCS 3329, pp. 230–244. Springer, 2004.