An extended abstract version of this work appears in: Ron Steinfeld and Philip Hawkes, editors, *Proc. 15th Australasian Conference on Information Security and Privacy (ACISP) 2010*, LNCS. Springer 2010. To appear.
This is the full version. EPRINT http://eprint.iacr.org/2009/430.

One-time-password-authenticated key exchange (full version)

Kenneth G. Paterson
Information Security Group, Royal Holloway, University of London, Egham, Surrey, UK
kenny.paterson@rhul.ac.uk

Douglas Stebila
Information Security Institute, Queensland University of Technology, Brisbane, Australia
douglas@stebila.ca

April 19, 2010

Abstract

To reduce the damage of phishing and spyware attacks, banks, governments, and other security-sensitive industries are deploying *one-time password* systems, where users have many passwords and use each password only once. If a single password is compromised, it can be only be used to impersonate the user once, limiting the damage caused. However, existing practical approaches to one-time passwords have been susceptible to sophisticated phishing attacks.

We give a formal security treatment of this important practical problem. We consider the use of one-time passwords in the context of password-authenticated key exchange (PAKE), which allows for mutual authentication, session key agreement, and resistance to phishing attacks. We describe a security model for the use of one-time passwords, explicitly considering the compromise of past (and future) one-time passwords, and show a general technique for building a secure one-time-PAKE protocol from any secure PAKE protocol. Our techniques also allow for the secure use of pseudorandomly generated and time-dependent passwords.

Keywords: one-time passwords, key exchange, protocols, cryptography

1 Introduction

Many security attacks on the Internet today, such as phishing and spyware, aim to compromise a user's password. As a result, some businesses and government agencies are deploying *one-time password* systems. In these systems, users carry a sheet of paper listing passwords or an electronic device that generates passwords, and use a different password each time they log in. Ideally, without obtaining this physical list of passwords (or the device generating them), an attacker should be unable to impersonate the user.

It is unfortunately too easy these days for passwords to be compromised. For example, users at an Internet café cannot trust that the café operator has not installed a key logger, yet they may still have an urgent need to login to a particular website. Many home users unknowingly have malware installed on their computer. One-time password systems can help reduce the damage from such compromises: although we cannot prevent the password from being stolen, it can only be used once, and reveals no information about future passwords. As a result, one-time password systems are being deployed by banks, governments, and corporate virtual private networks (VPNs).

However, most deployments of one-time passwords have not used them in the strongest way possible. In a typical usage, Alice visits a bank's website in her browser, views a challenge on the website indicating which one-time password to use, and enters that one-time password into her browser, which transmits the one-time password to the website. This type of usage remains susceptible to the same phishing attacks that threaten regular passwords today: if Alice did not really have an encrypted link with her actual bank, then an attacker may be able to learn the one-time password and impersonate Alice. Unfortunately, average users are not very good at telling if a SSL/TLS connection is really encrypted and authenticated.

More advanced cryptographic protocols, such as *password-authenticated key exchange* (PAKE), can allow us to use passwords in a secure way that reveals no useful information about the password to a phishing or man-in-the-middle attacker. These protocols can provide strong mutual authentication as well: not only does the bank learn whether Alice knows her password, but Alice learns whether the bank knows her password.

To date, one-time password schemes have not been formally studied using techniques from provable security. One existing work [ACP05a] presents a PAKE protocol that uses pseudorandom passwords, but does not consider how the security properties of one-time passwords or pseudorandom passwords differ from normal long-term passwords. The goal of this work is to describe and formalize security properties for one-time password systems, especially in the context of authenticated key exchange protocols.

We emphasize that one-time password schemes are practical, as numerous deployments [RSA09, Nat09, Nor09, F-S05, Bli09] have shown. Businesses that have already deployed one-time passwords in the form of token cards or sheets of paper could benefit from the greater security offered by our techniques by upgrading their back end systems without needing to deploy new password data to users; however, clients would need to upgrade their browsers or VPN clients to support these new protocols.

Contributions. In this work, we aim to answer three questions on the security of one-time password schemes:

1. How should we model the security of one-time password schemes?
2. How should we build secure one-time password schemes?
3. Are existing one-time password schemes secure?

To answer the first question, we describe in Section 2 an extension to the Bellare-Pointcheval-Rogaway [BPR00] PAKE security model that adds one-time passwords and handles the compromise of other past or future one-time passwords.

For the second question, we give a general construction in Section 3 for building a one-time-PAKE protocol from any PAKE protocol and show that this transformation preserves security. The transformation itself is straightforward and efficient, and allows for extensions to the basic functionality of one-time passwords: the secure use of pseudorandomly generated passwords (Section 4), time-dependent passwords (Section 5), and verifier-based one-time passwords, in which the server stores a one-way transformation of the passwords, not the passwords themselves (Section 2.2).

Existing uses of one-time passwords over TLS connections can be troublesome as they require a public key infrastructure and users often have difficulty validating public keys. To our knowledge, the only existing consideration of one-time passwords in PAKE is the OPKeyX protocol [ACP05a], which requires the one-time passwords be of a particular form (namely, a hash chain), and that future passwords not be revealed. We discuss the security of OPKeyX in Section 6, noting that our model is stronger and allows for arbitrary passwords to be revealed.

Outline. The rest of this paper is organized as follows. In Section 1.1, we describe related work. Section 2 deals with the security of one-time password protocols: it introduces the general properties we seek, and then presents a security model encompassing those properties. In Section 3, we give our central theoretical result that secure one-time-password-authenticated key exchange protocols can be built out of secure password-authenticated key exchange protocols. We then discuss the use of pseudorandom (Section 4) and time-dependent (Section 5) passwords. We conclude with a brief discussion of how this work relates to the existing OPKeyX protocol in Section 6 and some general conclusions in Section 7.

1.1 Related work

Many businesses, especially banks, have adopted one-time passwords in their authentication procedures. One-time passwords can be efficiently deployed using electronic tokens [RSA09], using a chip-and-pin card in combination with a reader device as some British banks are doing [Nat09], or on sheets of paper as some European banks do [Nor09]; interestingly, there have subsequently been phishing attacks specifically targeting these sheets of one-time passwords [F-S05]. One-time passwords are also being used for stronger authentication in virtual economies such as World of Warcraft [Bli09]. The Internet Engineering Task Force (IETF) has standardized various mechanisms for deriving [Hal95, HMNIS98] and using [Kau05, Nys07, KS08] one-time passwords. While all of these systems may generate and deploy one-time passwords securely, none of them proceed to use one-time passwords in cryptographically secure way.

Password-authenticated key exchange was first introduced by Bellovin and Merritt in 1992 [BM92] as a protocol in which the client and server share a plaintext password and exchange encrypted information to allow them to derive a shared session key. A later variant [BM93], often called *verifier-based*, removed the requirement that the server have the plaintext password, instead having a one-way transformation of the password.

The most extensively used model for the security of PAKE protocols is the Bellare-Pointcheval-Rogaway (BPR) model [BPR00] and its extension [GMR05] for verifier-based protocols. This model is the starting point of our model for the security of one-time-PAKE protocols. One particular such protocol is the PAK protocol [BMP00a, Mac02], which is the basis of our construction in the full version of this paper.

Various authors have noted the value of using one-time passwords in authenticated key exchange protocols [ACP05a, FMCS04, Ste09]. Abdalla *et al.* [ACP05a] (see also [CSH05]) describe the OPKeyX protocol, a verifier-based one-time-PAKE protocol. It uses a hash chain to derive subsequent one-time passwords from a seed such that the server can verify but not compute the next password. We will discuss OPKeyX in greater detail in Section 6.

2 Security of one-time-password protocols

The main security property that protocols employing one-time passwords should achieve is: strong mutual authentication based on knowledge of one-time passwords. Our work will address one-time passwords in the context of PAKE protocols, which provide an additional property: secure key exchange.

The motivation for using one-time passwords is that the compromise of one password should not affect the security of sessions involving another password. The one-time password serves to mutually authenticate the client and the server; there are no other long-term values like public keys or certificates. Authentication is based on knowledge of the shared password. Informally, a protocol will provide secure mutual authentication if no honest party $\hat{A}$ accepts a session as being with party $\hat{B}$ unless $\hat{B}$ participated in the protocol, and vice versa. We want a one-time-password protocol to give secure mutual authentication for the current session even if other one-time passwords have been revealed. Such passwords could be revealed accidentally by the user or obtained by an adversary who has installed malware on the user's computer, for example.

In addition to mutually authenticating two parties to each other, we want a protocol that will also output a *session key* that can be used to encrypt and protect the integrity of future communications between those two parties. This is a common feature required of many secure communication protocols.

The traditional use of one-time passwords – sending the password over a TLS connection – is not compatible with our approach. Using TLS to establish an authentic channel requires that the user can obtain and properly use an authentic public key for the server. In other words, it requires a public key infrastructure, whereas one-time-PAKE only needs shared passwords. We need not remove the TLS infrastructure, however: one-time password-authenticated key exchange could be provided as a new TLS cipher suite.

2.1 Security model

In the most widely adopted security model for PAKE, that of Bellare, Pointcheval, and Rogaway [BPR00], when the adversary corrupts a party it learns all of the party's authentication secrets at once. In the one-time password setting, we want to model the situation where users have multiple passwords and the attacker can learn the passwords one by one. This more closely models the functionality, design goals, and capabilities of the adversary in many one-time password scenarios.

Participants. An instance of the protocol takes place between two interacting parties, each of which is a member of the set $\mathsf{Parties}$; each party is identified by a unique fixed length string. Each pair of distinct parties $\{\hat{A}, \hat{B}\}$ shares a set of one-time passwords $\{\mathsf{pw}_{\hat{A},\hat{B},\mathsf{ch}}\}$ indexed by $\mathsf{ch} \in \mathsf{Indices}$, the set $\mathsf{Indices}$ being publicly known (we use the notation ch to suggest that the one-time password may be selected in response to a challenge, although the model does not assume that need be the case). We note that $\mathsf{pw}_{\hat{A},\hat{B},\mathsf{ch}} = \mathsf{pw}_{\hat{B},\hat{A},\mathsf{ch}}$ (this is the symmetric setting; in Section 2.2, we discuss how to model verifier-based one-time passwords). The size of the set $\mathsf{Indices}$ determines the maximum number of passwords shared between each pair of parties. Each one-time password is chosen uniformly at random from the set $\mathsf{Passwords}$.[1]

Protocol execution. The protocol is a message-driven protocol. During execution, a party $\hat{U}$ may have multiple instances of the protocol running; each instance is modelled as an oracle and is denoted by $\Pi^{\hat{U}}_{(\hat{U}',\mathsf{ch})}$: it is indexed by the values $(\hat{U}', \mathsf{ch}) \in \mathsf{Parties} \times \mathsf{Indices}$, where $\hat{U}'$ is its purported partner and ch is the one-time password index for that instance. A party $\hat{U}$ must be *activated* to act as an initiator or a responder with $\hat{U}'$ for a particular instance by having oracle $\Pi^{\hat{U}}_{(\hat{U}',\mathsf{ch})}$ be sent a message of the form "initiator" or "responder", respectively. An instance for a particular partner-index pair can only be activated once. This restriction can be achieved by having each party maintain a record of used one-time passwords. In practice, this is easy to achieve: for example, a user could cross out a one-time password on a piece of paper once it has been used, or increment a counter if pseudorandomly generated passwords are used.

There are distinguished instances $\Pi^{\hat{U}}_{(\hat{U}',\perp)}$ which can be sent messages of the form "initiator" or "responder"; $\hat{U}$ then picks an unused one-time password index ch and activates the corresponding instance $\Pi^{\hat{U}}_{(\hat{U}',\mathsf{ch})}$ with the given role.

There is a sequence of messages, or *flows*, specified by the protocol, starting with a flow from the initiator to the responder, then from the responder to the initiator, and so on. After some number of flows, an instance may *accept*, at which point it holds a *session key* sk, *partner id* pid, and *session id* sid, and, possibly after some additional flows, *terminate*. Alternatively, at any point in time, an instance may *reject* (note that instances that reject have not terminated; accepting is a precondition for terminating). Two instances $\Pi^{\hat{A}}_{(\mathsf{pid},\mathsf{ch})}$ and $\Pi^{\hat{B}}_{(\mathsf{pid}',\mathsf{ch}')}$ are said to be *partnered* if they both accept, hold $(\mathsf{pid}, \mathsf{sid}, \mathsf{sk})$ and $(\mathsf{pid}', \mathsf{sid}', \mathsf{sk}')$, respectively, with $\mathsf{pid} = \hat{B}$, $\mathsf{pid}' = \hat{A}$, $\mathsf{sid} = \mathsf{sid}'$, $\mathsf{sk} = \mathsf{sk}'$, and $\mathsf{ch} = \mathsf{ch}'$, and no other instance accepts with session id equal to sid. It is likely that the session identifier will include the one-time password index ch.

Definition 1 (Correctness) *A protocol is said to be* correct *if, for all distinct* $\hat{A}, \hat{B} \in \mathsf{Parties}$ *and all* $\mathsf{ch} \in \mathsf{Indices}$*, whenever messages are faithfully relayed between* $\Pi^{\hat{A}}_{\hat{B},\mathsf{ch}}$ *and* $\Pi^{\hat{B}}_{\hat{A},\mathsf{ch}}$*, both instances are partnered and terminate with probability 1.*

Queries allowed. The protocol is determined by how participants respond to inputs from the environment, and the environment is considered to be controlled by the adversary, which is a probabilistic algorithm that issues queries to parties' oracle instances and receives responses. For a protocol P, the queries that the adversary can issue are as follows (where clear by the setting, we may omit the subscript P). The first two queries model normal operation of the protocol:

[1] One common complaint about models for PAKE protocols is the typical assumption that passwords are uniformly distributed. In practice, human-selected passwords are rarely uniformly distributed. By contrast, one-time passwords are more likely in practice to be uniformly distributed since they are often generated by a computer.

- $\mathsf{Execute}_P(\hat{A},\hat{B},\mathsf{ch})$: This query activates initiator instance $\Pi^{\hat{A}}_{(\hat{B},\mathsf{ch})}$ and responder instance $\Pi^{\hat{B}}_{(\hat{A},\mathsf{ch})}$ with one-time password indexed by ch, causes them to faithfully execute protocol P, and returns the resulting transcript.
- $\mathsf{Send}_P(\hat{U},(\hat{U}',\mathsf{ch}),M)$: Send message M to user instance $\Pi^{\hat{U}}_{(\hat{U}',\mathsf{ch})}$, which performs the appropriate portion of protocol P based on its current state and the message M, updates its state, and returns any resulting messages.

The next two queries model the compromise of information by the adversary:

- $\mathsf{RevealSessionKey}_P(\hat{U},\hat{U}',\mathsf{ch})$: If instance $\Pi^{\hat{U}}_{(\hat{U}',\mathsf{ch})}$ has accepted, then it returns the session key sk held by $\Pi^{\hat{U}}_{(\hat{U}',\mathsf{ch})}$.
- $\mathsf{RevealPW}_P(\hat{U},\hat{U}',\mathsf{ch})$: Returns the one-time password $\mathsf{pw}_{\hat{U},\hat{U}',\mathsf{ch}}$.

The $\mathsf{RevealPW}$ query models the adversary learning the authentication secrets, which corresponds to weak corruption in the Bellare-Pointcheval-Rogaway model. The adversary cannot modify stored authentication secrets (also called strong corruption). We note that the $\mathsf{RevealPW}(\hat{U},\hat{U}',\mathsf{ch})$ query allows the adversary to reveal any password, regardless of whether it has been used in a session.

The final query is used to define the task that the adversary has to achieve in order for the session key security of the protocol to be considered broken. To define security, the adversary will interact with a *challenger* who, simulating the parties, answers all the queries above, as well as this one:

- $\mathsf{Test}_P(\hat{U},\hat{U}',\mathsf{ch})$: If instance $\Pi^{\hat{U}}_{(\hat{U}',\mathsf{ch})}$ has accepted, then the following happens: the challenger chooses $b \in_R \{0,1\}$; if $b=1$, then it returns the session key held by $\Pi^{U}_{(\hat{U}',\mathsf{ch})}$, otherwise it returns a random string of the same length as the session key. This query may only be asked once.

Freshness. We adapt the notion of freshness in the Bellare-Pointcheval-Rogaway model to allow the adversary to compromise one-time passwords from any session except the target session.

Definition 2 (Freshness) *In a one-time-PAKE protocol, an instance $\Pi^{\hat{U}}_{(\hat{U}',\mathsf{ch})}$ is* fresh *(with forward-secrecy) if and only if none of the following events occur:*

1. *a* $\mathsf{RevealSessionKey}(\hat{U},\hat{U}',\mathsf{ch})$ *query occurs;*
2. *a* $\mathsf{RevealSessionKey}(\hat{U}',\hat{U},\mathsf{ch})$ *query occurs;*
3. *either of the following queries occur before the* Test *query:*
   (a) $\mathsf{RevealPW}(\hat{U},\hat{U}',\mathsf{ch})$ *or (b)* $\mathsf{RevealPW}(\hat{U}',\hat{U},\mathsf{ch})$*;*
   and $\mathsf{Send}(\hat{U},(\hat{U}',\mathsf{ch}),M)$ *occurs for some string M.*

We note that this definition of freshness allows the adversary considerable power in terms of revealed passwords. In particular, the adversary could reveal every one-time password – past and future – except the single password for the target session.

Adversary's goals. The adversary's goals are to break either the confidentiality of the session key or the security of the mutual authentication.

For confidentiality, the goal of an adversary is to guess the bit b used in the Test query of a fresh session: this corresponds to the ability of an adversary to distinguish the session key from a random string of the same length. Let $\mathsf{Succ}^{1\times\mathrm{ake}}_P(\mathcal{A})$ be the event that the adversary $\mathcal{A}$ makes a single Test query to some fresh instance $\Pi^{\hat{U}}_{(\hat{U}',\mathsf{ch})}$ that has accepted and $\mathcal{A}$ eventually outputs a bit b', where $b'=b$ and b is the randomly selected bit in the Test query. The $1\times$ake-*advantage* of $\mathcal{A}$ attacking P is defined to be

$$\mathsf{Adv}^{1\times\mathrm{ake}}_P(\mathcal{A}) = \left|2\Pr\left(\mathsf{Succ}^{1\times\mathrm{ake}}_P(\mathcal{A})\right)-1\right| \quad . \tag{1}$$

We can define a similar notion for *mutual authentication*. Let $\mathsf{Succ}^{1\times\mathrm{ma}}_P(\mathcal{A})$ be the event that the adversary $\mathcal{A}$ causes a participant instance $\Pi^{\hat{U}}_{(\hat{U}',\mathsf{ch})}$ with partner id $\hat{U}'$ and one-time password index ch to terminate without a partnered instance, before either of the $\mathsf{RevealPW}$ queries in part 3 of Definition 2. The $1\times$ma-*advantage* of $\mathcal{A}$ attacking P is defined to be $\mathsf{Adv}^{1\times\mathrm{ma}}_P(\mathcal{A}) = \Pr\left(\mathsf{Succ}^{1\times\mathrm{ma}}_P(\mathcal{A})\right)$.

Definition 3 (Security) *Let λ be a security parameter. A protocol P is a* secure one-time-password-authenticated key agreement protocol *if, for all adversaries $\mathcal{A}$ running in time polynomial in λ and making at most q_{se} Send_P queries, there exists a constant δ and a negligible $\epsilon(\lambda)$ such that*

$$\mathsf{Adv}_P^{1\times\mathrm{ake}}(\mathcal{A}) \le \frac{\delta q_{\mathsf{se}}}{|\mathsf{Passwords}|} + \epsilon(\lambda) \ , \tag{2}$$

and a similar bound applies for $\mathsf{Adv}_P^{1\times\mathrm{ma}}(\mathcal{A})$.

This notion of security says that no polynomially bounded adversary can do negligibly better than randomly guessing an unknown password in each online attempt and can gain no advantage by doing an offline dictionary attack.

This bound is of the same form as bounds for the security of PAKE. One might expect that we could do better in the one-time password setting, since passwords are not reused. However, the adversary *always* has a password guessing strategy each time it participates in the protocol, leading to the $q_{\mathsf{se}}/|\mathsf{Passwords}|$ factor. Hence, this bound is effectively the best possible, up to making δ or $\epsilon(\lambda)$ smaller. The advantage of one-time password systems comes from their robustness in the face of richer models of compromise.

Remark. This security definition protects users from authentication and confidentiality failures. It offers no protection against denial of service attacks, especially attacks in which an attacker aims to exhaust a user's supply of one-time passwords. An adversary could keep a client and server "out of sync" on which password to use, preventing a connection from being established. Unless there is some additional form of server-to-client authentication – for example, the challenge being signed by a server certificate, which is outside the scope of this work since it would require a public key infrastructure – this appears to be unavoidable.

2.2 Verifier-based one-time passwords

In the verifier-based model, the server stores a *verifier*, which is a one-way transformation of the client's password that cannot be used to impersonate the user. This offers increased security against server database compromise. The security of verifier-based PAKE protocols is defined by the extension of the BPR model given by Gentry *et al.* [GMR05]. The main difference is that an instance can remain fresh even if either the password or the verifier (but not both) is compromised. This necessitates the introduction of a new query for revealing the verifier. Additionally, it allows for the separate definitions of client-to-server and server-to-client authentication.

The model we described in Section 2.1 can be extended in the natural way to use verifier-based one-time passwords by introducing a $\mathsf{RevealV}$ query to reveal one-time verifiers and adjusting the freshness definition appropriately; the details appear in Appendix A.

3 A generic construction for one-time password protocols

We now describe a technique for building a one-time-PAKE protocol, $1(P)$, out of any PAKE protocol P, and then show that the one-time-password protocol is at least as secure as the password protocol out of which it is built. The basic idea is that a PAKE protocol in which passwords are used only once is also a good one-time-PAKE protocol.

3.1 Construction of $1(P)$ from P

The construction proceeds as follows. For each client-server-index combination $(\hat{C}, \hat{S}, \mathsf{ch})$ in the one-time-password protocol, we will construct a new pair of users with compound names $(\hat{C}, \hat{S}, \mathsf{ch})$ and $(\hat{S}, \hat{C}, \mathsf{ch})$ in the password protocol, and pass the queries against the session in the one-time-password protocol down to the new pair of users in the underlying password protocol. Since every PAKE protocol should be secure even if each pair of users is used only once, this constructed one-time-PAKE protocol should also be secure.

We now specify in detail the technique to construct a one-time-PAKE protocol $1(P)$ from a PAKE protocol P. There are two phases: the *registration phase*, in which pairs of clients and servers establish passwords, and the *login phase*, in which pairs of clients and servers attempt to establish a secure session.

Registration phase. The registration phase of the $1(P)$ protocol is specified in Figure 1 below. For every client-server pair $(\hat{C},\hat{S}) \in \mathsf{Parties} \times \mathsf{Parties}$, and for each one-time-password index $\mathsf{ch} \in \mathsf{Indices}$, initiate the registration phase of P with the users $(\hat{C},\hat{S},\mathsf{ch})$ and $(\hat{S},\hat{C},\mathsf{ch})$, and set $\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}}$ in $1(P)$ equal to the corresponding password in P.

Although one might be concerned about the time that it takes to complete the registration phase if $\mathsf{Indices}$ is large, the registration phase of any one-time password protocol can not, in general, be completed in less time asymptotically if truly one-time passwords are used. In other words, this is effectively the same complexity as password establishment in currently deployed one-time password schemes, and hence is quite practical. Moreover, the registration for each challenge can be run in parallel to reduce the number of communication rounds.

Protocol $1(P)$ – Registration Phase		
	Client $\hat{C}$	Server $\hat{S}$
for each $\mathsf{ch} \in \mathsf{Indices}$:		
1.	run registration phase of protocol P with users $(\hat{C},\hat{S},\mathsf{ch})$ and $(\hat{S},\hat{C},\mathsf{ch})$	
2.	$\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}}$ in $1(P) \leftarrow \mathsf{pw}_{(\hat{C},\hat{S},\mathsf{ch}),(\hat{S},\hat{C},\mathsf{ch})}$ in P	
3.	$\mathsf{used}_{\hat{C}}(\hat{S},\mathsf{ch}) \leftarrow \mathsf{false}$	$\mathsf{used}_{\hat{S}}(\hat{C},\mathsf{ch}) \leftarrow \mathsf{false}$
end for each		

Figure 1: Protocol $1(P)$ – Registration Phase; must use a private, authenticated channel.

Login phase. The login phase of the $1(P)$ protocol is specified in Figure 2 below. Each party $\hat{U}$ maintains a set of tables $\mathsf{used}_{\hat{U}}(\hat{U}',\mathsf{ch})$, where each entry in the table is either true or false and indicates whether the one-time-password indexed by ch has been used by $\hat{U}$ with $\hat{U}'$.

To initiate the protocol, instance $\Pi^{\hat{C}}_{(\hat{S},\perp)}$ of user $\hat{C}$ sends a message ("hello", $\hat{C}$) to instance $\Pi^{\hat{S}}_{(\hat{C},\perp)}$ of party $\hat{S}$. When a party $\hat{S}$ receives a message ("hello", $\hat{C}$), it picks a one-time-password index ch from $\mathsf{Indices}$ such that $\mathsf{used}_{\hat{S}}(\hat{C},\mathsf{ch}) = \mathsf{false}$. Then it sets $\mathsf{used}_{\hat{S}}(\hat{C},\mathsf{ch}) \leftarrow \mathsf{true}$ and activates $\Pi^{\hat{S}}_{(\hat{C},\mathsf{ch})}$. Finally, it sends "hello" to instance $\Pi^{\hat{C}}_{(\hat{S},\mathsf{ch})}$ of party $\hat{C}$. It then waits to engage in a single instance of protocol P acting as user $(\hat{S},\hat{C},\mathsf{ch})$ interacting with party $(\hat{C},\hat{S},\mathsf{ch})$. When the corresponding instantiation of protocol P accepts, the instance in $1(P)$ sets its session key to the session key in P and then accepts. When it rejects in P, it rejects in $1(P)$; when it terminates in P, it terminates in $1(P)$ as well.

When instance $\Pi^{\hat{C}}_{(\hat{S},\mathsf{ch})}$ of party $\hat{C}$ receives a message ("hello"), it checks to see if $\mathsf{used}_{\hat{C}}(\hat{S},\mathsf{ch}) = \mathsf{true}$; if so, then it rejects; if not, then it sets $\mathsf{used}_{\hat{C}}(\hat{S},\mathsf{ch}) \leftarrow \mathsf{true}$. It then initiates the login phase of protocol P acting as user $(\hat{C},\hat{S},\mathsf{ch})$ interacting with party $(\hat{S},\hat{C},\mathsf{ch})$. It follows protocol P until it accepts or rejects. When the corresponding instantiation of protocol P accepts, the instance in $1(P)$ sets its session key to the session key in P and then accepts. When it rejects in P, it rejects in $1(P)$; when it terminates in P, it terminates in $1(P)$ as well.

It follows easily from inspection that, if P is correct, $1(P)$ is also correct.

3.2 Security of $1(P)$

Theorem 1 *Let P be a secure password-authenticated key exchange protocol. Then $1(P)$ is a secure one-time-password-authenticated key exchange protocol.*

Due to length restrictions, the security argument appears in Appendix B. The basic idea of the argument is as follows. We will show that attacks against $1(P)$ correspond to attacks against P. We construct a $1(P)$ simulator in which the adversary's queries to $1(P)$ are translated into queries on a P challenger as follows:

- $\mathsf{Execute}_{1(P)}(\hat{A},\hat{B},\mathsf{ch})$: Return the result of $\mathsf{Execute}_P((\hat{A},\hat{B},\mathsf{ch}),1,(\hat{B},\hat{A},\mathsf{ch}),1)$.

Protocol $1(P)$ **– Login Phase**			
	Client $\hat{C}$		Server $\hat{S}$
1.		$\xrightarrow{\text{“hello”},\hat{C}}$	
2.			pick $\mathsf{ch} \in \mathsf{Indices}$ s.t. $\mathsf{used}_{\hat{S}}(\hat{C},\mathsf{ch}) = \mathsf{false}$
3.			$\mathsf{used}_{\hat{S}}(\hat{C},\mathsf{ch}) \leftarrow \mathsf{true}$
4.		$\Pi^{\hat{C}}_{(\hat{S},\mathsf{ch})} \xleftarrow{\text{“hello”}}$	
5.	if $(\mathsf{used}_{\hat{C}}(\hat{S},\mathsf{ch}) = \mathsf{true})$ then reject		
6.	$\mathsf{used}_{\hat{C}}(\hat{S},\mathsf{ch}) \leftarrow \mathsf{true}$		
7.	$\boxed{\text{run protocol } P \text{ with users } (\hat{C},\hat{S},\mathsf{ch}) \text{ and } (\hat{S},\hat{C},\mathsf{ch}) \text{ and password } \mathsf{pw}_{(\hat{C},\hat{S},\mathsf{ch}),(\hat{S},\hat{C},\mathsf{ch})}}$		
8.	if P accepts then		if P accepts then
8.a)	$\mathsf{sid}_{1(P)} \leftarrow \mathsf{sid}_P$; $\mathsf{pid} \leftarrow \hat{S}$		$\mathsf{sid}_{1(P)} \leftarrow \mathsf{sid}_P$; $\mathsf{pid} \leftarrow \hat{C}$
8.b)	$\mathsf{sk}_{1(P)} \leftarrow \mathsf{sk}_P$		$\mathsf{sk}_{1(P)} \leftarrow \mathsf{sk}_P$
8.c)	accept in $1(P)$		accept in $1(P)$
9.	if P terminates then terminate		if P terminates then terminate
10.	if P rejects then reject		if P rejects then reject

Figure 2: Protocol $1(P)$ – Login Phase; can use a public, unauthenticated channel.

- $\mathsf{Send}_{1(P)}(\hat{U},(\hat{U}',\mathsf{ch}),M)$: If message M is for one of the two flows added by the $1(P)$ construction, then respond as indicated in Figure 2. If message M is for one of the flows from P, then return the result of $\mathsf{Send}_P((\hat{U},\hat{U}',\mathsf{ch}),1,M)$.
- $\mathsf{RevealSessionKey}_{1(P)}(\hat{U},\hat{U}',\mathsf{ch})$: Return $\mathsf{RevealSessionKey}_P((\hat{U},\hat{U}',\mathsf{ch}),1)$.
- $\mathsf{RevealPW}_{1(P)}(\hat{A},\hat{B},\mathsf{ch})$: Return $\mathsf{RevealPW}_P((\hat{A},\hat{B},\mathsf{ch}),(\hat{B},\hat{A},\mathsf{ch}))$.
- $\mathsf{Test}_{1(P)}(\hat{U},\hat{U}',\mathsf{ch})$: Return $\mathsf{Test}_P((\hat{U},\hat{U}',\mathsf{ch}),1)$.

Using this simulation, if an adversary could break $1(P)$, it could break P just as efficiently. But since P is a secure PAKE protocol, no adversary should be able to attack P, and hence no adversary should be able to attack $1(P)$. The argument is a straightforward simulation involving creating separate user instances in P for each instance of $1(P)$ by constructing users in P with identities that are the concatenation of the user name and one-time password index from $1(P)$, and assuring that fresh instances in $1(P)$ correspond to fresh instances in P. We note that the security reduction is tight.

Example instantiation. Suppose we were to construct a one-time-password-authenticated key exchange protocol using the $1(P)$ construction where the underlying password-authenticated key exchange protocol is the (symmetric, non-verifier-based) protocol PAK [BMP00a]. The $1(\mathsf{PAK})$ protocol is particularly interesting because, with an appropriate reordering of messages, it can be made to fit inside the message flow of the TLS handshake protocol. This makes it suitable for use as a new cipher suite in TLS. A full presentation of the $1(\mathsf{PAK})$ protocol is given in Appendix C.

In our example, we wish for an adversary to be able to break the one-time-password protocol with probability at most 2^{-20}, where the adversary runs in time at most 2^{60}, and can only make a limited number (2^{10}) of Send queries. Assuming the hardness of solving the elliptic curve computational Diffie-Hellman problem (using estimates in [BCC+08]), we can achieve this security level using 10-digit numerical passwords ($\mathsf{Passwords} = \{0,\ldots,9\}^{10}$) and a 348-bit elliptic curve group. (See Appendix C.1 for the full analysis.)

3.3 Efficiency and practicality of $1(P)$

Login phase and computational efficiency. During the login phase, the $1(P)$ construction provides no loss of efficiency in terms of the number of expensive operations (such as group exponentiations) or security level of P, since the reduction is tight. $1(P)$ does add two additional message flows to the length of the protocol, but depending on the message flow of protocol P it may be possible to combine some flows without affecting security.

One might think that designing a one-time-password protocol from scratch may lead to greater efficiency, since some of the effort in designing PAKE protocols goes to preventing the transcript of one session leaking information about the password and in a one-time-password protocol we may not have to worry as much about leaking information about passwords during a protocol run. However, many PAKE protocols are already highly efficient in terms of number of operations. For example, the Diffie-Hellman-based PAK [Mac02] protocol can be run with just 2 group exponentiations on each side (plus a group inversion on the client side, which is inexpensive in many groups like elliptic curve groups), which is very close to the operation count of the basic, unauthenticated Diffie-Hellman protocol (2 group exponentiations for both parties). The main efficiency to be gained, then, would be in improving the tightness in the security reduction to the underlying Diffie-Hellman problem so as to allow smaller group sizes.

Registration phase. The registration phase of $1(P)$ obviously requires establishing many more passwords than a single instance of P, but *any* one-time password scheme requires establishing many more passwords than a long-term password scheme. The $1(P)$ registration phase calls the registration phase of P many times. Depending on the PAKE protocol P, the registration phase can be quite efficient: for example, in the PAK protocol [Mac02], the registration phase can be optimized to consist of just one hash function evaluation.

Password storage. In practice it is important to consider how clients will store a list of one-time passwords, especially if they wish to log in to a site while away from their normal computer. One method is to provide a piece of paper with a list of one-time passwords; for example, the Swedish bank Nordea provides its customers with a “scratch sheet” of 120 one-time passwords [Nor09]. Alternatively, one-time passwords could be delivered through an out-of-band channel such as an SMS message to the user’s mobile phone (for example, [Mob]). Passwords can also be stored on or generated by an electronic token device, for example the RSA SecurID [RSA09], or even in a smart card built into credit cards [Pri08].

We can further reduce the complexity of the registration phase and password storage by using pseudorandom or time-based one-time passwords, which we describe in the following sections.

4 Using pseudorandom passwords

To improve the efficiency of password registration and storage, it may be desirable to pseudorandomly generate passwords instead of truly random ones. For example, users may be given a hardware token [RSA09, Bli09] with a preprogrammed private seed which iteratively generates one-time passwords, or the device may accept a challenge as an input and then output a response from a pseudorandom function based on the seed and that challenge. We show that pseudorandomly generated passwords can be safely used in one-time-PAKE protocols.

Suppose P is secure one-time-PAKE protocol. We construct a new protocol $\tilde{P}$ based on P that uses pseudorandomly generated passwords as follows.

We modify the registration phase of $\tilde{P}$ as follows. For each (unordered) pair of users $\{\hat{A}, \hat{B}\} \in \mathsf{Parties} \times \mathsf{Parties}$, choose a random seed $\mathsf{seed}_{\hat{A},\hat{B}} \in_R \{0,1\}^\lambda$, where λ is a security parameter. Let $\mathcal{F} = \{F_k\}$ be a family of pseudorandom functions [GGM86]. For each one-time-password index $\mathsf{ch} \in \mathsf{Indices}$, set $\mathsf{pw}_{\hat{A},\hat{B},\mathsf{ch}} = F_{\mathsf{seed}_{\hat{A},\hat{B}}}(\mathsf{ch})$.

The login phase of $\tilde{P}$ is exactly as in P, except that the passwords chosen in the modified registration phase above are used. For the purposes of the security model in Section 2.1, the $\mathsf{RevealPW}$ queries work exactly as before and only reveal an individual password $\mathsf{pw}_{\hat{A},\hat{B},\mathsf{ch}}$. No query reveals $\mathsf{seed}_{\hat{A},\hat{B}}$.[2]

The only difference between $\tilde{P}$ and P is that pseudorandom passwords are being used instead of random passwords. It is then easy to see that any efficient adversary $\mathcal{A}$ that can defeat session key security or mutual authentication in $\tilde{P}$ can be used to build either an adversary $\mathcal{A}_1$ that breaks

[2]We could add a further query, say $\mathsf{RevealPWSeed}(\hat{A},\hat{B})$, that does reveal the value $\mathsf{seed}_{\hat{A},\hat{B}}$ and then add an additional constraint to the definitions of freshness and authentication so that an instance is not considered fresh if the relevant $\mathsf{RevealPWSeed}$ query is called before the Test query. This enhanced model would make it clear that corruption of one pair of users’ pseudorandom seed should not affect the security of another pair of users. It is not hard to see that this construction would satisfy this enhanced security model, assuming independent random seeds.

session key security or mutual authentication in P, or an algorithm $\mathcal{A}_2$ that acts as a polynomial-time distinguisher for the pseudorandom function family $\mathcal{F}$. Thus $\tilde{P}$ is secure if P is, and we see that pseudorandomly generated one-time passwords can be safely used in any secure one-time-PAKE protocol.

5 Using time-dependent pseudorandom passwords

A further refinement to the use of pseudorandomly generated passwords is to use passwords that also depend on the current time. This allows the client and server to agree upon a challenge – the current time – without any communication, while easily enforcing the one-time use of passwords.

For example, consider a hardware token for party $\hat{A}$ interacting with party $\hat{B}$ which has a pseudorandom function $F_{\mathsf{seed}_{\hat{A},\hat{B}}} \in \mathcal{F}$ and an onboard clock. It generates one-time passwords as follows. Let t be the hardware token's current time. Treat t as the one-time password index ch, and then compute

$$\mathsf{pw}_{\hat{A},\hat{B},t} = F_{\mathsf{seed}_{\hat{A},\hat{B}}}(t) \ . \tag{3}$$

User $\hat{A}$ then participates in the one-time-PAKE protocol using $\mathsf{pw}_{\hat{A},\hat{B},t}$.

Whenever clocks are used by two parties, one must consider the issue of *clock skew*, in which the two clocks may not be perfectly synchronized. For example, ordinary quartz clocks drift at a rate of approximately 10^{-6} seconds per second, or about 1 second every 12 days.

One solution is to have a common network time server that both parties use for synchronization. This is problematic for two reasons: (1) the network time server must be trusted (or at least dealt with in the security model); (2) all of the parties participating must have a way of synchronizing with the clock server; an inexpensive, credit-card-sized hardware token may not be connected to the network, making synchronization difficult or impossible.

Another method for dealing with clock skew is to have the server accept multiple passwords from a small window around the server's current time (say, plus or minus 60 seconds). However, this is a problem for PAKE protocols, as the server never receives the client's password directly. Rather, each party uses what it believes to be the password in the protocol, and at the end the two parties know that the same password was used if and only if they arrive at the same session key. This prevents the server from accepting multiple passwords as valid. (Traditional one-time password systems often avoid this problem by having the client send the password itself to the server over an existing encrypted but not mutually authenticated channel.)

A simpler alternative mechanism for dealing with clock skew is for one party (the initiator) to just tell the other party (the responder) what time t it used in the protocol. If the time used by the initiator is acceptable to the responder (say within plus or minus 60 seconds of the responder's clock) then the responder continues the protocol using the specified time. This provides a simple mechanism for ensuring both sides use the same time-dependent password while accommodating clock skew.

Adjusting the model. In order to accommodate this alternate mechanism in the security model described in Section 2.1, the definition of freshness would need to be adapted (in part 3.(a) and 3.(b) of Definition 2) so that a responder instance $\Pi^{\hat{U}}_{(\hat{U}',t)}$ is fresh provided that no $\mathsf{RevealPW}(\hat{U},\hat{U}',t)$ or $\mathsf{RevealPW}(\hat{U}',\hat{U},t)$ query was issued. This captures the notion that the authentication should be secure as long as the currently valid password has not been revealed.

With this modified security definition, and assuming $\mathcal{F}$ is a secure family of pseudorandom functions, one-time time-based passwords generated in equation (3) can be safely used in a secure one-time-PAKE protocol as a result of the discussion about pseudorandom passwords in Section 4.

6 Analysis of the OPKeyX protocol

The OPKeyX protocol [ACP05a] is a PAKE protocol that uses a sequence of passwords derived via a hash chain from a single seed. The protocol is a verifier-based protocol, meaning that the compromise of the value stored on the server should not allow someone to impersonate the client. We note that [ACP05a] omits a complete analysis of OPKeyX: it gives a proof for a non-verifier-based PAKE protocol in the BPR model but no proof that OPKeyX, a verifier-based hash-chain variant of the protocol, is also secure.

The sequence of passwords in OPKeyX is as follows. Each client $\hat{C}$ picks, for each server $\hat{S}$, a seed password pw. Let $N_{\max}$ be the maximum number of login sessions for the seed pw. During the registration phase, the client gives the server its verifier $V_{N_{\max}} \leftarrow f^{N_{\max}+1}(\mathsf{pw})$, where f is a random oracle [BR93] and f^i denotes the i-fold application of f. The parties each maintain internal counters n of the current login phase, starting from $n = N_{\max}$ and decreasing to 1. During login phase with internal counter equal to n, the client and server do an encrypted key exchange where the Diffie-Hellman ephemeral public keys are encrypted using a value derived from the verifier $V_n = f^{n+1}(\mathsf{pw})$. Then, the client encrypts $f^n(\mathsf{pw})$ under a value s derived from the shared Diffie-Hellman key (but distinct from the session key sk) and sends it to the client. The server decrypts to obtain V', verifies $V_n = f(V')$, and sets $V_{n-1} \leftarrow V'$ and $n \leftarrow n-1$.

OPKeyX relies on the correct sequence of passwords being used. In the security model for verifier-based one-time-PAKE in Appendix A, we allow the adversary to reveal one-time passwords in any order. As a result, OPKeyX cannot be a secure verifier-based one-time-PAKE protocol in that sense. For example, an adversary could reveal the password for session with counter i, which is $f^{i+1}(\mathsf{pw})$, and then be able to derive the password for the earlier session with counter $i+1$ (recalling that counters decrease as time passes), which is $f^{i+2}(\mathsf{pw}) = f(f^{i+1}(\mathsf{pw}))$. To describe the security of OPKeyX, we would need to further restrict our model so that a session is not fresh if the password or verifier of a subsequent session has been revealed which, although weaker from a theoretical perspective, still models a plausible practical scenario. The situation is even more complicated if RevealSessionKey is deemed to reveal the value s (which encrypted the next verifier V' and is in some sense a "session key") in addition to sk, in which case no earlier s value for the target users can have been revealed before the Test query.

7 Conclusions

One-time password systems are already being widely deployed by banks, governments, and corporate virtual private networks (VPNs) to reduce the effects of password compromise. Bank customers today are using sheets of paper with lists of one-time passwords. Online shoppers and gamers today are using hardware one-time password generators. The money being spent on deploying one-time passwords is wasted if these passwords are not being used safely and securely.

By using one-time passwords in one-time-PAKE protocols, as we have proposed in this paper, we can be assured that one-time passwords are being used in a more secure way. We have presented a model for the secure use of one-time passwords in PAKE protocols, taking into account the idea that such protocols should be secure even if previous or future one-time passwords have been compromised. We have given a generic technique for constructing secure one-time password protocols. Our construction can be used with pseudorandomly generated one-time passwords or time-based one-time passwords, providing greater efficiency in one-time password distribution.

An important open problem based on this work is the task of determining whether it is possible to construct one-time password protocols that are more efficient than regular password protocols, as discussed in Section 3.3.

As with all cryptographic protocols, an essential precondition to security is *getting users to use the protocol*. If an adversary can trick a user into entering their password in a non-secure manner so that the secure protocol is never used – a so-called *chosen protocol attack* – then the cryptographic countermeasures are bypassed. For any PAKE protocol to succeed, user training and user interface design will be very important.

Spyware remains a significant threat to password security. In the face of passive spyware, such as a keystroke logger which collects information and occasionally relays it back to the attacker, both traditional one-time password schemes and one-time-PAKE are useful since used one-time passwords are useless to an attacker. If the spyware is active – it captures a one-time password, terminates the user's connection, and immediately sends the password to the attacker – the captured password may still be useful to an attacker, and it seems that neither traditional one-time password schemes nor one-time-PAKE can do much unless time-dependent passwords are used with careful expiration procedures.

An additional challenge is widespread deployment of such secure protocols. Passwords, as they are used in HTTP and TLS on the Internet today, remain susceptible to phishing attacks. The huge installed base of web browsers and web servers has significantly slowed efforts to deploy PAKE. Our techniques, like PAKE, require some changes to TLS implementations. It may be possible to implement a large portion of a new security protocol as a browser add-on (like a Firefox extension), making deployment easier.

Our approach may see more immediate application in corporate virtual private network (VPN) software. Many corporate VPNs use one-time passwords now, albeit in a less secure way than we have proposed. Moreover, both endpoints – the user's computer and the VPN server – are often under control of the same organization and using software from the same vendor, making it easier to deploy enhancements. An interesting avenue of future research is the integration of secure PAKE and one-time-PAKE protocols into IPsec for use in corporate VPNs. Indeed, IKEv2 (one of the key exchange protocols for IPsec) notes the need for password authentication: after showing how to derive a shared key for authenticated Diffie-Hellman key exchange in IKEv2, the RFC goes on to say:

> "... deriving the shared secret from a password is not secure. This construction is used because it is anticipated that people will do it anyway" [Kau05, p. 30].

One-time-password-authenticated key exchange is one way in which one-time passwords can be used more securely.

References

[ACP05a] Michel Abdalla, Olivier Chevassut, and David Pointcheval. One-time verifier-based encrypted key exchange. In Serge Vaudenay, editor, *Public Key Cryptography (PKC) 2005*, *LNCS*, volume 3386, pp. 47–64. Springer, 2005. DOI:10.1007/b105124. Full version available as [ACP05b].

[ACP05b] Michel Abdalla, Olivier Chevassut, and David Pointcheval. One-time verifier-based encrypted key exchange, 2005. URL `http://www.di.ens.fr/~mabdalla/papers/ACP05-letter.pdf`. Extended abstract published as [ACP05a].

[BCC+08] Steve Babbage, Dario Catalano, Carlos Cid, Orr Dunkelman, Christian Gehrmann, Louis Granboulan, Tanja Lange, Arjen Lenstra, Phong Q. Nguyen, Christof Paar, Jan Pelzl, Thomas Pornin, Bart Preneel, Christian Rechberger, Vincent Rijmen, Matt Robshaw, Andy Rupp, Nigel Smart, and Michael Ward. ECRYPT yearly report on algorithms and keysizes (2007–2008), July 2008. URL `http://www.ecrypt.eu.org/documents/D.SPA.28-1.1.pdf`.

[Bli09] Blizzard Entertainment. Blizzard authenticator, 2009. URL `http://eu.blizzard.com/support/article.xml?locale=en_GB&articleId=28152`.

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the 1992 IEEE Computer Society Conference on Research in Security and Privacy*. IEEE, May 1992. DOI:10.1109/RISP.1992.213269.

[BM93] Steven M. Bellovin and Michael Merritt. Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. In *Proc. 1st ACM Conference on Computer and Communications Security (CCS)* [CCS93], pp. 244–250. DOI:10.1145/168588.168618.

[BMP00a] Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably secure Password-Authenticated Key exchange using Diffie-Hellman. In Preneel [Pre00], pp. 156–171. DOI:10.1007/3-540-45539-6_12. Full version available as [BMP00b].

[BMP00b] Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably secure Password-Authenticated Key exchange using Diffie-Hellman, 2000. EPRINT `http://eprint.iacr.org/2000/044`. Short version published as [BMP00a].

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Preneel [Pre00], pp. 139–155. DOI:10.1007/3-540-45539-6_11.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security (CCS)* [CCS93], pp. 62–73. DOI:10.1145/168588.168596.

[CCS93] *Proc. 1st ACM Conference on Computer and Communications Security (CCS)*. ACM, 1993.

[CSH05] Olivier Chevassut, Frank Siebenlist, and Mike Helm. Secure (one-time-) password authentication for the Globus toolkit. In *GlobusWorld Conference*, February 2005. URL http://acs.lbl.gov/Projects/OPKeyX/Talks/GlobusWorld05/GlobusWorld05.html.

[F-S05] F-Secure. Weblog: More on international phishing, October 2005. URL http://www.f-secure.com/weblog/archives/00000689.html.

[FMCS04] Liang Fang, Samuel Meder, Olivier Chevassut, and Frank Siebenlist. Secure password-based authenticated key exchange for web services. In *Proc. 2004 Workshop on Secure Web Service (SWS)*, pp. 9–15. ACM, 2004. DOI:10.1145/1111348.1111350.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, **33**(4):792–807, October 1986. DOI:10.1145/6490.6503.

[GMR05] Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. PAK-Z+, August 2005. URL http://grouper.ieee.org/groups/1363/WorkingGroup/presentations/pakzplusv2.pdf. Contribution to the IEEE P1363-2000 study group for Future PKC Standards.

[Hal95] Neil Haller. The S/KEY one-time password system, February 1995. URL http://www.ietf.org/rfc/rfc1760.txt. RFC 1760.

[HMNIS98] Neil Haller, Craig Metz, Philip J. Nesser II, and Mike Straw. A one-time password system, February 1998. URL http://www.ietf.org/rfc/rfc2289.txt. RFC 2289.

[Kau05] Charlie Kaufman. Internet Key Exchange (IKEv2) protocol, December 2005. URL http://www.ietf.org/rfc/rfc4306.txt. RFC 4306.

[KS08] Sunil Kumar and Abhishek Sing. One time password in IKE version 2 (non-EAP based), November 2008. URL http://tools.ietf.org/id/draft-sunabhi-otp-ikev2-03.txt. Internet-Draft.

[Mac02] Philip MacKenzie. The PAK suite: Protocols for password-authenticated key exchange. Technical Report 2002-46, DIMACS Center, Rutgers University, 2002. URL http://dimacs.rutgers.edu/TechnicalReports/abstracts/2002/2002-46.html.

[Mob] Mobile-OTP Project. Mobile one time passwords. URL http://motp.sourceforge.net/.

[Nat09] Nationwide Building Society. Card reader security, May 2009. URL http://www.nationwide.co.uk/rca/.

[Nor09] Nordea Bank. Netbank security, 2009. URL http://www.nordea.ee/Private+customers/E-channels++Netbank/Netbank/Netbank+Security/936612.html.

[Nys07] Magnus Nystroem. The EAP protected one-time password protocol (EAP-POTP), February 2007. URL http://www.ietf.org/rfc/rfc4793.txt. RFC 4793.

[Pre00] Bart Preneel, editor. *Advances in Cryptology – Proc. EUROCRYPT 2000, LNCS*, volume 1807. Springer, 2000. DOI:10.1007/3-540-45539-6.

[Pri08] Mark Prigg. The new credit card with keypad that promises to fight online fraud, November 2008. URL http://www.dailymail.co.uk/sciencetech/article-1085642/The-new-credit-card-keypad-promises-fight-online-fraud.html?ITO=1490. The Daily Mail Online.

[RSA09] RSA Security Inc. RSA SecurID, 2009. URL http://www.rsa.com/node.aspx?id=1156.

[Sho99a] Victor Shoup. On formal models for secure key exchange. Report RZ 3120, IBM Research, April 1999. URL http://www.zurich.ibm.com/security/publications/1999/Shoup99.ps.gz.

[Sho99b] Victor Shoup. On formal models for secure key exchange (version 4), November 1999. URL http://shoup.net/papers/skey.pdf. Earlier version appeared as [Sho99a].

[Ste09] Douglas Stebila. *Classical Authenticated Key Exchange and Quantum Cryptography*. PhD thesis, University of Waterloo, 2009. EPRINT http://hdl.handle.net/10012/4295, URL http://www.douglas.stebila.ca/research/papers/ste09/.

A Verifier-based one-time passwords

In this section, we follow that approach and modify the security model of Section 2.1 to define the security of a verifier-based one-time password-authenticated key exchange protocol.

Participants. We distinguish between two types of parties: Clients and Servers, with $\mathsf{Parties} = \mathsf{Clients} \dot\cup \mathsf{Servers}$. Each distinct client-server pair $(\hat{C}, \hat{S}) \in \mathsf{Clients} \times \mathsf{Servers}$ has a set of one-time password-verifier pairs $\{(\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}}, \mathsf{V}_{\hat{C},\hat{S},\mathsf{ch}})\}$ indexed by ch. Each of the one-time passwords $\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}}$ is chosen uniformly at random from the set Passwords, and $\mathsf{V}_{\hat{C},\hat{S},\mathsf{ch}}$ is the verifier for the corresponding password, the derivation of which is specified by the registration phase of the protocol. The client stores the one-time passwords and the server stores the one-time verifiers.

Queries allowed. The allowed queries are modified as follows. The RevealPW query is modified to model the compromise of the client's one-time password, and a new RevealV query is introduced to model the comprise of the server's one-time verifier.

- $\mathsf{RevealPW}_P(\hat{A}, \hat{B}, \mathsf{ch})$: If $\hat{A} \in \mathsf{Clients}$ and $\hat{B} \in \mathsf{Servers}$, then it returns the one-time password $\mathsf{pw}_{\hat{U},\hat{U}',\mathsf{ch}}$; otherwise, it returns an error.
- $\mathsf{RevealV}_P(\hat{A}, \hat{B}, \mathsf{ch})$: If $\hat{A} \in \mathsf{Clients}$ and $\hat{B} \in \mathsf{Servers}$, then it returns the one-time verifier $\mathsf{V}_{\hat{A},\hat{B},\mathsf{ch}}$; otherwise, it returns an error.

Freshness. The notion of freshness for verifier-based one-time-password-authenticated key exchange protocol is adapted as follows.

Definition 4 (Freshness, verifier-based) *In a verifier-based one-time-password authenticated protocol, an instance* $\Pi^{\hat{U}}_{(\hat{U}',\mathsf{ch})}$ *is* fresh *(with forward-secrecy) if and only if none of the following events occur:*

1. *a* $\mathsf{RevealSessionKey}(\hat{U}, \hat{U}', \mathsf{ch})$ *query occurs;*
2. *a* $\mathsf{RevealSessionKey}(\hat{U}', \hat{U}, \mathsf{ch})$ *query occurs;*
3. *if* $\hat{U} \in \mathsf{Clients}$, *either of the following queries occur before the* Test *query:*
   (a) $\mathsf{RevealPW}(\hat{U}, \hat{U}', \mathsf{ch})$, *or*
   (b) $\mathsf{RevealV}(\hat{U}, \hat{U}', \mathsf{ch})$,
   and $\mathsf{Send}(\hat{U}, (\hat{U}', \mathsf{ch}), M)$ *occurs for some string* M.
4. *if* $\hat{U} \in \mathsf{Servers}$, $\mathsf{RevealPW}(\hat{U}', \hat{U}, \mathsf{ch})$ *occurs before the* Test *query, and* $\mathsf{Send}(\hat{U}, (\hat{U}', \mathsf{ch}), M)$ *occurs for some string* M.

Adversary's goals. The adversary's goal related to confidentiality is the same, using $\mathsf{Adv}^{1\times\mathsf{ake}}_P(\mathcal{A})$.

The adversary's goals related to authentication are modified as follows. We introduce separate notions of *client-to-server*, *server-to-client*, and *mutual* authentication. Let $\mathsf{Succ}^{1\times\mathsf{c2s}}_P(\mathcal{A})$ be the event that the adversary $\mathcal{A}$ causes a server instance $\Pi^{\hat{S}}_{(\hat{C},\mathsf{ch})}$ with partner id $\hat{C}$ to terminate without a partnered instance, before the RevealPW or RevealV queries in part 3 of Definition 4. Let $\mathsf{Succ}^{1\times\mathsf{s2c}}_P(\mathcal{A})$ be the event that the adversary $\mathcal{A}$ causes a client instance $\Pi^{\hat{C}}_{(\hat{S},\mathsf{ch})}$ with partner id $\hat{S}$ to terminate without a partnered instance, before the RevealPW query in part 4 of Definition 4. Finally, let $\mathsf{Succ}^{1\times\mathsf{ma}}_P(\mathcal{A}) = \mathsf{Succ}^{1\times\mathsf{c2s}}_P(\mathcal{A}) \vee \mathsf{Succ}^{1\times\mathsf{s2c}}_P(\mathcal{A})$. We define the corresponding advantages $\mathsf{Adv}^{1\times\mathsf{c2s}}_P(\mathcal{A})$, $\mathsf{Adv}^{1\times\mathsf{s2c}}_P(\mathcal{A})$, and $\mathsf{Adv}^{1\times\mathsf{ma}}_P(\mathcal{A})$ analogously to Section 2.1.

In order to accommodate security in the case of the verifier being compromised, we employ the random oracle model [BR93] to achieve a definition similar to the definition of verifier-based PAKE by Gentry, MacKenzie, and Ramzan [GMR05, Theorem 5.1].

Definition 5 (Security, verifier-based) *Let λ be a security parameter. A protocol P is a* secure verifier-based one-time-password-authenticated key agreement protocol *in the random oracle model if, for all adversaries $\mathcal{A}$ running in time polynomial in λ and making at most q_{se} Send_P queries and at most q_{ro} random oracle queries, there exists a constants δ_1 and δ_2 and a negligible $\epsilon(\lambda)$ such that*

$$\mathsf{Adv}_P^{1\times\mathsf{ake}}(\mathcal{A}) \le \frac{\delta_1 q_{\mathsf{se}}(1-b_{\mathsf{RV}}) + \delta_2 q_{\mathsf{ro}} b_{\mathsf{RV}}}{|\mathsf{Passwords}|} + \epsilon(\lambda) \ , \tag{4}$$

and a similar bound applies for $\mathsf{Adv}_P^{1\times\mathsf{ma}}(\mathcal{A})$, where $b_{\mathsf{RV}} = 1$ if a $\mathsf{RevealV}$ *query occurs and $b_{\mathsf{RV}} = 0$ otherwise.*

The construction of $1(P)$ can be modified in the obvious way to support verifier-based protocols as well. Using an argument similar to the one for Theorem 1, we can easily show the following result:

Theorem 2 *Let P be a secure verifier-based password-authenticated key exchange protocol. Then $1(P)$, modified to support verifier-based protocols, is a secure verifier-based one-time-password-authenticated key exchange protocol.*

B Security argument for $1(P)$ construction

The basic idea of the argument is as follows. We will show that attacks against $1(P)$ correspond to attacks against P. As a result, if an adversary could break $1(P)$, it could break P. Thus, an adversary could use the $1(P)$ construction, which is efficient, as part of its algorithm for breaking P. But since P is a secure password-authenticated key exchange protocol, no adversary should be able to attack P, and hence no adversary should be able to attack $1(P)$ either. The security argument is a straightforward simulation involving creating separate user instances in P for each instance of $1(P)$.

PROOF. In order to show that $1(P)$ is a secure one-time-password-authenticated key exchange protocol, we need to show that it provides secure key exchange and secure mutual authentication.

First, we construct a $1(P)$ simulator in which the adversary's queries to $1(P)$ are translated into queries on a P challenger as follows:

- $\mathsf{Execute}_{1(P)}(\hat{A},\hat{B},\mathsf{ch})$: Return the result of $\mathsf{Execute}_P((\hat{A},\hat{B},\mathsf{ch}),1,(\hat{B},\hat{A},\mathsf{ch}),1)$.
- $\mathsf{Send}_{1(P)}(\hat{U},(\hat{U}',\mathsf{ch}),M)$: If message M is for one of the two flows added by the $1(P)$ construction, then respond as indicated in Figure 2. If message M is for one of the flows from P, then return the result of $\mathsf{Send}_P((\hat{U},\hat{U}',\mathsf{ch}),1,M)$.
- $\mathsf{RevealSessionKey}_{1(P)}(\hat{U},\hat{U}',\mathsf{ch})$: Return $\mathsf{RevealSessionKey}_P((\hat{U},\hat{U}',\mathsf{ch}),1)$.
- $\mathsf{RevealPW}_{1(P)}(\hat{A},\hat{B},\mathsf{ch})$: Return $\mathsf{RevealPW}_P((\hat{A},\hat{B},\mathsf{ch}),(\hat{B},\hat{A},\mathsf{ch}))$.
- $\mathsf{Test}_{1(P)}(\hat{U},\hat{U}',\mathsf{ch})$: Return $\mathsf{Test}_P((\hat{U},\hat{U}',\mathsf{ch}),1)$.

Next, we show that a fresh session in the $1(P)$ simulator corresponds to a fresh session in the P challenger. Then we show that a session-key distinguisher for a fresh session of $1(P)$ is a session-key distinguisher for a fresh session of P, and hence $1(P)$ provides secure key agreement.

Suppose $\Pi_1^{(\hat{U},\hat{U}',\mathsf{ch})}$ with partner id $(\hat{U}',\hat{U},\mathsf{ch})$ is not a fresh instance in the P challenger. We will show that $\Pi_{(\hat{U}',\mathsf{ch})}^{\hat{U}}$ is not a fresh instance in the $1(P)$ simulator.

If $\Pi_1^{(\hat{U},\hat{U}',\mathsf{ch})}$ with partner id $(\hat{U}',\hat{U},\mathsf{ch})$ is not a fresh instance in the P challenger, then one of the following must have occurred:

- $\mathsf{RevealSessionKey}_P((\hat{U},\hat{U}',\mathsf{ch}),1)$ occurred. A $\mathsf{RevealSessionKey}_{1(P)}(\hat{U},\hat{U}',\mathsf{ch})$ query must have occurred, since no other query in $1(P)$ leads to this query in P. Hence, $\Pi_{(\hat{U}',\mathsf{ch})}^{\hat{U}}$ is not fresh in $1(P)$.
- $\mathsf{RevealSessionKey}_P((\hat{U}',\hat{U},\mathsf{ch}),1)$ occurred, where $\Pi_1^{(\hat{U}',\hat{U},\mathsf{ch})}$ is the partner instance of $\Pi_1^{(\hat{U},\hat{U}'\mathsf{ch})}$ in P. In this case, a $\mathsf{RevealSessionKey}_{1(P)}(\hat{U}',\hat{U},\mathsf{ch})$ query must have occurred, since no other query in $1(P)$ leads to this query in P. Hence, $\Pi_{(\hat{U}',\mathsf{ch})}^{\hat{U}}$ is not fresh in $1(P)$.

- $\mathsf{RevealPW}_P((\hat{U},\mathsf{ch}),(\hat{U}',\mathsf{ch}))$ occurred before the Test_P query and $\mathsf{Send}_P((\hat{U},\mathsf{ch}),1,M)$ occurred for some message M. In this case, $\mathsf{RevealPW}_{1(P)}(\hat{U},\hat{U}',\mathsf{ch})$ must have occurred in $1(P)$ before the $\mathsf{Test}_{1(P)}$ query and $\mathsf{Send}_{1(P)}(\hat{U},(\hat{U}',\mathsf{ch}),M)$ must have occurred, since no other sequence of queries in $1(P)$ leads to this sequence of queries in P. Hence, $\Pi^{\hat{U}}_{(\hat{U}',\mathsf{ch})}$ is not fresh in $1(P)$.

Thus, $\Pi_1^{(\hat{U},\hat{U}',\mathsf{ch})}$ with partner id $(\hat{U}',\hat{U},\mathsf{ch})$ is a fresh instance in P.

Having shown that fresh sessions in the $1(P)$ simulator are also fresh in the P chappenger, we now show that breaking session key security of the $1(P)$ simulator leads to breaking session key security of the P challenger.

We first note that, since each one-time password $\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}}$ for $1(P)$ was chosen according to the distribution Passwords for P, the distribution of passwords with which P is initialized satisfied the protocol requirements of P.

By the argument above, every fresh session in $1(P)$ corresponds to a fresh session in P. Since the session key in $1(P)$ is equal to the session key in P and since the output of $\mathsf{Test}_{1(P)}$ is equal to the output of Test_P, a session key distinguisher for fresh sessions of $1(P)$ will also distinguish session keys for P. Thus, an adversary's $1\times$ake-advantage in $1(P)$ cannot be better than its ake-advantage in P:

$$\mathsf{Adv}^{1\times\mathrm{ake}}_{1(P)}(t,q_{\mathsf{se}},q_{\mathsf{ex}},q_{\mathsf{pw}},q_{\mathsf{ro}}) \le \mathsf{Adv}^{\mathrm{ake}}_P(t,q_{\mathsf{se}},q_{\mathsf{ex}},q_{\mathsf{pw}},q_{\mathsf{ro}}) \ . \tag{5}$$

Finally, we show that the ability of an adversary to break authentication in $1(P)$ is related to its ability to break authentication in P, in a manner analogous to the discussion above for key agreement security. Suppose an instance of $1(P)$ terminates without a partnered instance, before any of the queries in part 3 of Definition 2. Instances in $1(P)$ terminate if and only if the corresponding instance in P has terminated. Moreover, none of the prohibited queries in the definition of mutual authentication for password-authenticated key exchange could have occurred since otherwise one of the prohibited queries for mutual authentication of one-time-password-authenticated key exchange must have occurred. Hence, mutual authentication must have been broken for P as well. Thus, an adversary's $1\times$ma-advantage in $1(P)$ cannot be better than its ma-advantage in P:

$$\mathsf{Adv}^{1\times\mathrm{ma}}_{1(P)}(t,q_{\mathsf{se}},q_{\mathsf{ex}},q_{\mathsf{pw}},q_{\mathsf{ro}}) \le \mathsf{Adv}^{\mathrm{ma}}_P(t,q_{\mathsf{se}},q_{\mathsf{ex}},q_{\mathsf{pw}},q_{\mathsf{ro}}) \ . \tag{6}$$

Thus $1(P)$ is a secure one-time-PAKE protocol if P is a secure PAKE protocol. □

We note that the converse does not necessarily hold, namely, that it is not necessarily the case that, if $1(P)$ is secure, so is P. For example, an attacker trying to attack P may do so by causing the users in P to run multiple sessions. If the adversary causes a pair of users $(\hat{C},\hat{S},\mathsf{ch})$ and $(\hat{S},\hat{C},\mathsf{ch})$ to run multiple sessions in P, then there is no way of mapping this to an allowable instance of $1(P)$, and we cannot translate the attack on P into an attack on $1(P)$.

C 1(PAK): a one-time-password-authenticated key exchange protocol

In this section, we present the protocol 1(PAK), an adaptation of the PAK protocol to use one-time passwords using the $1(P)$ construction of Section 3.1. In the presentation, the notation has been simplified from the $1(P)$ construction where appropriate (for example, we avoid the repetitive subscripting $\mathsf{pw}_{(\hat{C},\hat{S},\mathsf{ch}),(\hat{S},\hat{C},\mathsf{ch})}$ and simplify to the unambiguous $\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}}$). The PAK protocol was introduced by Boyko, MacKenzie, and Patel [BMP00a]. It is a symmetric, or non-verifier-based, protocol. The original paper [BMP00a] gave a proof that PAK was secure in the simulation model of Shoup [Sho99b] and was later shown to be secure in the BPR model by MacKenzie [Mac02].

Let G be a finite cyclic group of order q and let g be a generator of G. Let $\mathsf{Acceptable} : \overline{G} \to \{\mathsf{true},\mathsf{false}\}$ such that $\mathsf{Acceptable}(z) = \mathsf{true}$ if and only if $z \in \overline{G}$, where $\overline{G}$ is a specified abelian group which has G as a subgroup. H_1 is a full-domain random hash function returning elements of G; H_2, H_3, and H_4 are random hash functions returning suitably large bit strings.

The user registration phase of the 1(PAK) protocol is given in Figure 3. For each one-time password index ch $\in$ Indices, the server $\hat{S}$ uniformly at random picks a password $\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}} \in_R$ Passwords and stores a related value τ^{-1}. Then, using a private, authenticated channel, the server provides all the one-time passwords to the client $\hat{C}$ who stores them. Alternatively, the set of one-time passwords for each client-server pair could be selected by each client and supplied to the server over a secure channel; the end result would be the same.

Protocol 1(PAK) **– Registration Phase**			
	Client $\hat{C}$		Server $\hat{S}$
for each ch $\in$ Indices:			
1.			choose $\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}} \in_R$ Passwords
2.			$\tau^{-1} \leftarrow (H_1(\hat{C},\hat{S},\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}}))^{-1}$
3.			store $\mathsf{pw}_{\hat{S}}[\hat{C},\mathsf{ch}] \leftarrow \tau^{-1}$
end for each			
4.			$\mathcal{P}_{\hat{C},\hat{S}} \leftarrow \{(\mathsf{ch},\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}}) : \mathsf{ch} \in \mathsf{Indices}\}$
5.		$\xleftarrow{\mathcal{P}_{\hat{C},\hat{S}}}$	
6.	store $\mathcal{P}_{\hat{C},\hat{S}}$		
7.	$\mathsf{used}_{\hat{C}}(\hat{S},\mathsf{ch}) \leftarrow$ false for all ch $\in$ Indices		$\mathsf{used}_{\hat{S}}(\hat{C},\mathsf{ch}) \leftarrow$ false for all ch $\in$ Indices

Figure 3: Protocol 1(PAK) – Registration Phase. This phase must use a private, authenticated channel.

The login phase of the 1(PAK) protocol is given in Figure 4. In this phase, the client sends a "hello" message to the server to obtain the one-time password index ch and, provided that one-time password index is unused. follows the user login phase of the PAK protocol, which includes mutual authentication based on the shared password $\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}}$ and the computation of a shared session key sk based on the Diffie-Hellman shared secret.

Mackenzie [Mac02, Theorem 6.9] shows that PAK is a secure password-authenticated key exchange protocol in the BPR model, assuming the hardness of the computational Diffie-Hellman problem and using the random oracle model. Combining that result with our Theorem 1, we have that 1(PAK) is a secure one-time-password-authenticated key exchange protocol:

Theorem 3 *Let G be a finite cyclic group generated by g and let t_{exp} denote the running time of exponentiation in G. Assume passwords are uniformly distributed among the set* Passwords. *Let $\mathcal{A}$ be an adversary that runs in time t and makes at most q_{se} and q_{ex} queries of type* Send *and* Execute, *respectively, and at most q_{ro} queries to the random oracles. Then, for $t' = t + (4q_{\mathsf{ro}}^2 + q_{\mathsf{se}} + 2q_{\mathsf{ex}})t_{\mathsf{exp}}$,*

$$\mathsf{Adv}^{1\times\mathsf{ake}}_{1(\mathsf{PAK})}(\mathcal{A}) \leq \frac{q_{\mathsf{se}}}{|\mathsf{Passwords}|} + \epsilon \ , \tag{7}$$

where

$$\epsilon = 2q_{\mathsf{se}}\mathsf{Adv}^{\mathrm{CDH}}_{G,g}\left(t', q_{\mathsf{ro}}^2\right) + 2\frac{(q_{\mathsf{se}}+q_{\mathsf{ex}})(q_{\mathsf{ro}}+q_{\mathsf{se}}+q_{\mathsf{ex}})}{|G|} \tag{8}$$

and $\mathsf{Adv}^{\mathrm{CDH}}_{G,g}(t,\ell)$ is the advantage an adversary running in time t and outputting a list of ℓ items has in solving the (list) computational Diffie-Hellman (CDH) problem. Moreover, the same bound applies for $\mathsf{Adv}^{1\times\mathsf{ma}}_{1(\mathsf{PAK})}(\mathcal{A})$.

C.1 Parameter sizes for example instantiation

As a consequence of Theorem 3, we can pick a desired security level and under a suitable assumption on the difficulty of solving CDH, choose a set of parameters that achieve that security level.

Suppose we wish for an adversary running in time 2^{60} to have an ake advantage of at most 2^{-20} against 1(PAK).

To give an example instantiation, we have to pick appropriate values for the various parameters in the statement of the theorem. We choose $q_{\mathsf{se}} = 2^{10}$, $q_{\mathsf{ex}} = 2^{20}$, $q_{\mathsf{ro}} = 2^{40}$, $t = 2^{80}$, and $t_{\mathsf{exp}} = 2^{20}$.

Protocol 1(PAK) **– Login Phase**

	Client $\hat{C}$		Server $\hat{S}$
1.	input username $\hat{C}$		
2.		$\xrightarrow{\text{“hello”},\hat{C}}$	
3.			pick $\mathsf{ch} \in \mathsf{Indices}$ s.t. $\mathsf{used}_{\hat{S}}(\hat{C},\mathsf{ch}) = \mathsf{false}$
4.			$\mathsf{used}_{\hat{S}}(\hat{C},\mathsf{ch}) \leftarrow \mathsf{true}$
5.		$\xleftarrow{\text{“hello”},\mathsf{ch}}$	
6.	if $(\mathsf{used}_{\hat{C}}(\hat{S},\mathsf{ch}) = \mathsf{true})$ then reject		
7.	$\mathsf{used}_{\hat{C}}(\hat{S},\mathsf{ch}) \leftarrow \mathsf{true}$		
8.	lookup $\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}}$		
9.	$\tau = H_1(\hat{C},\hat{S},\mathsf{pw}_{\hat{C},\hat{S},\mathsf{ch}})$		
10.	$x \in_R \mathbb{Z}_q$		
11.	$X \leftarrow g^x$		
12.	$m \leftarrow X \cdot \tau$		
13.		$\xrightarrow{\hat{C},m}$	
14.			if $\neg\mathsf{Acceptable}(m)$ then reject
15.			$y \in_R \mathbb{Z}_q$
16.			$Y \leftarrow g^y$
17.			lookup $\tau^{-1} \leftarrow \mathsf{pw}_{\hat{S}}[\hat{C},\mathsf{ch}]$
18.			$X \leftarrow m \cdot \tau^{-1}$
19.			$\sigma \leftarrow X^y$
20.			$\mathsf{sid} \leftarrow (\hat{C},\hat{S},\mathsf{ch},m,Y)$; $\mathsf{pid} \leftarrow \hat{C}$
21.			$\mathsf{sk} \leftarrow H_2(\mathsf{sid},\sigma,\tau^{-1})$; accept
22.			$M_1 \leftarrow H_3(\mathsf{sid},\sigma,\tau^{-1})$
23.		$\xleftarrow{Y,M_1}$	
24.	$\sigma \leftarrow Y^x$		
25.	compute τ^{-1}		
26.	$\mathsf{sid} \leftarrow (\hat{C},\hat{S},\mathsf{ch},m,Y)$; $\mathsf{pid} \leftarrow \hat{S}$		
27.	$\mathsf{sk} \leftarrow H_2(\mathsf{sid},\sigma,\tau^{-1})$; accept		
28.	if $(M_1 \neq H_3(\mathsf{sid},\sigma,\tau^{-1}))$ then reject		
29.	$M_2 \leftarrow H_4(\mathsf{sid},\sigma,\tau^{-1})$		
30.		$\xrightarrow{M_2}$	
31.			if $(M_2 \neq H_4(\mathsf{sid},\sigma,\tau^{-1}))$ then reject

Figure 4: Protocol 1(PAK) – Login Phase. This phase can use a public, unauthenticated channel.

We need $\frac{q_{\mathsf{se}}}{|\mathsf{Passwords}|} \leq 2^{-21}$. Suppose each one-time password is comprised solely of numerical characters 0-9. We need $|\mathsf{Passwords}| \geq 2^{31}$ which can be achieved by using uniformly distributed 10 digit numerical passwords (since $10^{10} \approx 2^{33.2}$).

We also need $\epsilon \leq 2^{-21}$. Of the two terms in expression (8), the latter is dominated by the former. Noting that $t' = 2^{82}$, we require that $2^{11}\mathsf{Adv}^{\mathrm{CDH}}_{G,g}(2^{82}, 2^{60}) \leq 2^{-21}$. Assuming that the best technique to solve CDH is to solve the Discrete Logarithm problem and that the best method of doing so is as described in the detailed analysis found in an ECRYPT report [BCC+08, §6.1], we need an elliptic curve group of size $q \geq 2^{2(11+21+82+60)} = 2^{348}$.