Multi-Factor Password-Authenticated Key Exchange

Douglas Stebila[1] **Poornaprajna Udupi**[2] **Sheueling Chang**[3]

[1] Information Security Institute
Queensland University of Technology
Brisbane, Queensland, Australia
Email: douglas@stebila.ca

[2] Sun Microsystems Laboratories
Santa Clara, California, United States
Email: poornaprajna.udupi@sun.com

[3] Email: sheueling.shantz@gmail.com

Abstract

We consider a new form of authenticated key exchange which we call *multi-factor password-authenticated key exchange*, where session establishment depends on successful authentication of multiple short secrets that are complementary in nature, such as a long-term password and a one-time response, allowing the client and server to be mutually assured of each other's identity without directly disclosing private information to the other party.

Multi-factor authentication can provide an enhanced level of assurance in higher-security scenarios such as online banking, virtual private network access, and physical access because a multi-factor protocol is designed to remain secure even if all but one of the factors has been compromised.

We introduce a security model for multi-factor password-authenticated key exchange protocols, propose an efficient and secure protocol called MFPAK, and provide a security argument to show that our protocol is secure in this model. Our security model is an extension of the Bellare-Pointcheval-Rogaway security model for password-authenticated key exchange and accommodates an arbitrary number of symmetric and asymmetric authentication factors.

Keywords: multi-factor authentication, passwords, key exchange, cryptographic protocols

1 Introduction

Phishing and spyware are two of the major security problems on the Internet today. *Phishing*, or server impersonation, occurs when a malicious server convinces a user to reveal sensitive personal information, such as a username and password, to a malicious server instead of the real server. Additionally, many users' computers are compromised with *spyware*, which can record users' keystrokes (and thus passwords) and transmit this information to a malicious party. These attacks are possible not because of the break of any cryptographic protocol but because of externalities such as social engineering and software bugs.

In theory, these attacks can be addressed in part by using trusted cryptographic devices that can store private keys and perform cryptographic operations, but such devices are difficult to deploy and use. Years of experience have shown that passwords are a much more popular and easy-to-use form of authentication, but are more susceptible to phishing and spyware attacks. In this work, we focus on the use of passwords for authentication, since they are easier for users to use and carry between computers than long private keys.

Phishing can be combated by protocols that provide strong, easy-to-use server-to-client authentication. Password-authenticated key exchange (PAKE) can make server-to-client authentication easier and resistant to offline dictionary attacks, and additionally provides a secure key for encryption.

Spyware is more difficult to defend against. If a user's computer is compromised by passive spyware that records keystrokes and occasionally transmits this information to an attacker's server, then the use of *one-time passwords* may be effective, since a previously used one-time password can not be used again. Active spyware – that frequently communicates with the attacker's server and actively alters the user's computer – is nearly impossible to defend against without additional trusted hardware.

To reduce the damage caused by compromising an authentication factor, many organizations with high security requirements – such as financial institutions, governments, and corporate virtual private networks (VPNs) – are deploying *multi-factor authentication*, which depends on a variety of attributes, or *factors*. The factors could include: a long-term password, a set of one-time passwords, a private key, or a biometric. To be effective in practice, factors should have different, complementary natures of compromise. For example, one-time passwords cannot all be compromised unless one obtains the sheet of paper listing all the one-time passwords or the device generating the one-time passwords, whereas a biometric read by a trusted device (such as a secure fingerprint reader) should not be able to be reproduced without the presence of the person in question (or at least their finger).

Contributions. Our goal is to design a framework for multi-factor authentication protocols that provides flexibility in the number and nature of factors. Protocols secure in this framework should provide strong mutual authentication, convey the authentication secrets in a secure manner, and remain secure even if all but one of the authentication factors is compromised. The authentication secrets can be low-entropy secrets, such as passwords. Using multiple low-entropy secrets can allow for passwords that may have different modes of compromise, such as a memorized long-term password and a one-time password generated from a hardware device or transmitted over a mobile phone text message.

First, we define a security model which is an exten-

sion of the Bellare-Pointcheval-Rogaway model (Bellare et al. 2000) for PAKE. Our model allows for an arbitrary number of authentication factors, which can be either symmetric or asymmetric. Our security definition formalizes the notion that a multi-factor protocol should remain secure even if all but one of the factors has been compromised.

Next, we present an efficient multi-factor protocol that is secure in this model under standard cryptographic assumptions in the random oracle model. Our protocol combines facets of the PAK protocol (MacKenzie 2002) for symmetric factors and the PAK-Z+ protocol (Gentry et al. 2005) for asymmetric factors. We discuss how many different types of factors – long-term passwords, one-time passwords, biometrics, and even private keys – can be used in our protocol.

Our work differs from previous work in PAKE because it uses multiple authentication factors and maintains security even if some are compromised. Others have considered some aspects of multi-factor authentication, but these have either used at least one factor that is a long cryptographic secret (Yang et al. 2006, Park & Park 2004, Yoon & Yoo 2006, Pointcheval & Zimmer 2008), or have not provided strong server-to-client authentication resistant to man-in-the-middle attacks.

Outline. The rest of our paper proceeds as follows. In Section 2, we describe the security model for multi-factor PAKE. In Section 3, we present our protocol MFPAK and discuss its efficiency; we show through a formal analysis that the MFPAK protocol is secure and discuss how various types of factors can be used. Section 4 concludes the paper with what we believe are interesting directions for future research. Appendix A presents the one of the cases for our security proof for the MFPAK protocol; the rest appear in the full version of the paper (Stebila et al. 2009).

1.1 Related work

Password-authenticated key exchange was first introduced by Bellovin and Merritt in 1992 (Bellovin & Merritt 1992) as the encrypted key exchange (EKE) protocol, in which the client and server shared the plaintext password and exchanged encrypted information to allow them to derive a shared session key. A later variant by Bellovin and Merritt, Augmented EKE (A-EKE) (Bellovin & Merritt 1993), removed the requirement that the server have the plaintext password, instead having a (non-trivial) one-way transformation of the password, which alone is not sufficient to impersonate the user. The former is called a *symmetric* password-based protocol, because both client and server share the same plaintext password (or a trivial transformation of it), whereas the latter is called *asymmetric*. The dominant model for the security of PAKE protocols was proposed by Bellare, Pointcheval, and Rogaway (Bellare et al. 2000) and extended by Gentry, MacKenzie, and Ramzan (Gentry et al. 2005) to accommodate asymmetric protocols.

Many PAKE protocols have been developed, including PAK (Boyko et al. 2000, MacKenzie 2002) and PAK-Z+ (Gentry et al. 2005) which are relevant to our construction. Although universally composable constructions are attractive to consider when combining primitives, the existing work on universally composable PAKE (Canetti et al. 2005) is only symmetric, not asymmetric, and thus unsuitable for our approach.

A number of two-factor authentication schemes have been proposed that rely on a short password and a long cryptographic secret (Park & Park 2004, Yang et al. 2006, Yoon & Yoo 2006). Pointcheval and Zimmer (Pointcheval & Zimmer 2008) presented a multi-factor authentication scheme using a password, a long cryptographic secret, and biometric data; their scheme has a formal security argument in a variant of the BPR model that shares some features with ours.

There are also non-cryptographic approaches to multi-factor authentication, but these do not provide as strong protection for the authentication factors. In a *multi-channel* system, the second factor is delivered over a separate channel (for example, via an SMS text message on a mobile phone), which the user then inputs into their web browser along side their password. In a *multi-layer* system, software installed on the server evaluates additional attributes such as an HTTP cookie, IP address, and browser identification string to heuristically analyze whether the user is likely to be authentic. Some multi-layer systems try to offer additional reassurance to the user of the server's identity by presenting the user with a customized image or string. While these multi-channel and multi-layer approaches can offer some increased assurance, they can be defeated by non-cryptographic means such as sophisticated man-in-the-middle attacks and spyware, and have been shown to be easily ignored by users (Schecter et al. 2007).

2 Security for multi-factor protocols

In a multi-factor PAKE protocol, multiple authentication secrets of complementary natures, such as a long-term password and a one-time password, are used. We support two general types of authentication factors: symmetric and asymmetric.

The authentication secrets must be used in a way that the client can convince the server that it knows all the authentication secrets, and that the server can convince the client that it knows all the authentication secrets: this provides mutual authentication. However, the protocol must be carefully designed to not reveal any information about the authentication secrets to a passive or even active adversary.

Secure communications often involve both authentication and encryption so, in addition to providing authentication, we want protocols that establish an ephemeral shared secret key between client and server that can be used, for example, for bulk encryption.

Informal security criteria. The general security criteria we use for multi-factor PAKE is that the protocol should remain secure even if all but one authentication factor is known to an adversary. We identify four security properties such a protocol should have:

1. Strong multi-factor server-to-client authentication: without knowledge of all of the authentication factors, a server cannot successfully convince a client of its identity.
2. Strong multi-factor client-to-server authentication: without knowledge of all of the authentication factors, a client cannot successfully convince a server of its identity.
3. Authentication secrets protected: no useful information about the authentication secrets is revealed during the authentication process.
4. Secure session key establishment: at the end of the protocol, an honest client and an honest server end up with a secure shared session key suitable for bulk encryption if and only if the mutual authentication is successful; otherwise no session is established.

2.1 Security model

We define a model for the security of multi-factor PAKE that allows one to argue that a protocol is secure by giving upper bounds on the probability that an adversary can break server-to-client or client-to-server authentication, or determine the session key

established; the authentication secrets are protected from offline dictionary attacks as well.

This model is an extension of the model for PAKE proposed by Bellare, Pointcheval, and Rogaway (Bellare et al. 2000) and modified by Gentry, MacKenzie, and Ramzan (Gentry et al. 2005). The model allows for an arbitrary number of authentication factors, and each factor can be either symmetric or asymmetric.

Participants. In this model, each interacting party is either a *client* or a *server*, is identified by a unique fixed length string, and the identifier is a member of either the set $\mathsf{Clients}$ or $\mathsf{Servers}$, respectively, with $\mathsf{Parties} = \mathsf{Clients} \cup \mathsf{Servers}$.

Each authentication factor can be one of two types: *symmetric* or *asymmetric*. Suppose there are n factors; let I_s denote the indices of symmetric factors and I_a denote the indices of asymmetric factors. For each client-server pair $(C, S) \in \mathsf{Clients} \times \mathsf{Servers}$, n authentication factors exist. The ℓth authentication factor $\mathsf{pw}^\ell_{C,S}$ is chosen uniformly at random from the set $\mathsf{Passwords}^\ell$ and is stored by the client. For symmetric factors, the server also stores $\mathsf{pw}^\ell_{C,S}$; for asymmetric factors, the server stores a *verifier* $\overline{\mathsf{pw}}^\ell_{C,S}$, which is some non-trivial transformation of $\mathsf{pw}^\ell_{C,S}$. (The notion of "non-trivial transformation" will be clear in the freshness definition below, but intuitively the transformation should be such that compromise of the verifier alone should not be sufficient to impersonate the user without performing a dictionary attack.)

Execution of the protocol. During execution, a party may have multiple instances of the protocol running. Each instance i of a party $U \in \mathsf{Parties}$ is treated as an *oracle* denoted by Π^U_i.

In a protocol, there is a sequence of messages, called *flows*, starting with a flow from the client instance, responded to by a server instance, and so on. After some number of flows, an instance may *accept*, at which point it hold a *session key* sk, *partner id* pid, and *session id* sid. Subsequently, it may *terminate*. Two instances Π^C_i and Π^S_j 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 $\mathsf{pid} = S$, $\mathsf{pid}' = C$, $\mathsf{sid} = \mathsf{sid}'$, and $\mathsf{sk} = \mathsf{sk}'$, and no other instance accepts with session id equal to sid. Alternatively, an instance may *reject* at any point in time, meaning it is no longer accepted or terminated.

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 formally a probabilistic algorithm that issues queries to a *challenger* which simulates parties' oracle instances. For a protocol P, the queries that the adversary can issue are defined as follows (where clear by the setting, we may omit the subscript P):

- $\mathsf{Execute}_P(C, i, S, j)$: Causes client instance Π^C_i and server instance Π^S_j to faithfully execute protocol P and returns the resulting transcript.
- $\mathsf{Send}_P(U, i, M)$: Sends message M to user instance Π^U_i, which faithfully performs the appropriate portion of protocol P based on its current state and the message M, updates its state as appropriate, and returns any resulting messages.
- $\mathsf{Test}_P(U, i)$: If user instance Π^U_i has accepted, then the following happens: the challenger chooses $b \in_R \{0, 1\}$; if $b = 1$, then return the session key of Π^U_i, otherwise return a random string of the same length as the session key. This query may only be asked once.
- $\mathsf{RevealSK}_P(U, i)$: If user instance Π^U_i has accepted, then returns session key sk held by Π^U_i.
- $\mathsf{RevealFactor}_P(C, S, \ell)$: Returns the ℓth factor $\mathsf{pw}^\ell_{C,S}$ held by client C with server S.
- $\mathsf{RevealFactorV}_P(S, C, \ell)$: If ℓ is an asymmetric factor: returns the ℓth factor's verifier $\overline{\mathsf{pw}}^\ell_{C,S}$ held by server S with client C.

The $\mathsf{RevealFactor}$ and $\mathsf{RevealFactorV}$ queries model the adversary learning the authentication secrets, which corresponds to weak corruption in the Bellare-Pointcheval-Rogaway model. We do not allow the adversary to modify stored authentication secrets (also called strong corruption).

Definition 2.1 (Freshness) *An instance Π^U_i with partner id U' is* fresh in the ℓth factor *(with forward-secrecy) if and only if none of the following events occur:*

1. *a $\mathsf{RevealSK}(U, i)$ query occurs;*
2. *a $\mathsf{RevealSK}(U', j)$ query occurs, where $\Pi^{U'}_j$ is the partner instance of Π^U_i, if it exists;*
3. *if $U \in \mathsf{Clients}$: $\mathsf{RevealFactor}(U, U', \ell)$ (and/or $\mathsf{RevealFactorV}(U', U, \ell)$ if the ℓth factor is asymmetric) occurs before the Test query, and $\mathsf{Send}(U, i, M)$ occurs for some string M;*
4. *if $U \in \mathsf{Servers}$: $\mathsf{RevealFactor}(U', U, \ell)$ occurs before the Test query, and $\mathsf{Send}(U, i, M)$ occurs for some string M.*

This notion of freshness accommodates the idea that an instance should remain fresh even if all but one of the authentication factors has been fully compromised. If an instance is fresh in all of its factors, then it is also fresh in the original notion of freshness for PAKE.

Adversary's goals. For *session key security*, the goal of an adversary is to guess the bit b used in the Test query of an instance that is fresh in at least one of its factors; this corresponds to the ability of an adversary to distinguish the session key from a random string of the same length. Let $\mathsf{Succ}^{\text{ake-f}\ell}_P(\mathcal{A})$ be the event that the adversary $\mathcal{A}$ makes a single Test query to some fresh in the ℓth factor instance Π^U_i 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 *ake-fℓ advantage* of $\mathcal{A}$ attacking P is defined to be $\mathsf{Adv}^{\text{ake-f}\ell}_P(\mathcal{A}) = 2\Pr\left(\mathsf{Succ}^{\text{ake-f}\ell}_P(\mathcal{A})\right) - 1$.

We can define similar notions for *client-to-server*, *server-to-client*, and *mutual* authentication. For the security experiments involving authentication, the Test query is prohibited. We define $\mathsf{Adv}^{\text{c2s-f}\ell}_P(\mathcal{A})$ to be the probability that a server instance Π^S_j with partner id C terminates without having a partner oracle before the $\mathsf{RevealFactor}$ query in point 4 of the definition of freshness in the ℓth factor. We define $\mathsf{Adv}^{\text{s2c-f}\ell}_P(\mathcal{A})$ to be the probability that a client instance Π^C_i with partner id S terminates without having a partner oracle before the $\mathsf{Reveal}*$ queries in point 3 of the definition of freshness in the ℓth factor. Finally, we define $\mathsf{Adv}^{\text{ma-f}\ell}_P(\mathcal{A}) = \max\{\mathsf{Adv}^{\text{c2s-f}\ell}_P(\mathcal{A}), \mathsf{Adv}^{\text{s2c-f}\ell}_P(\mathcal{A})\}$.

We overload the Adv (and corresponding $\Pr(\mathsf{Succ})$) notation: $\mathsf{Adv}^N_P(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) = \max_{\mathcal{A}}\{\mathsf{Adv}^N_P(\mathcal{A})\}$, where the maximum is taken over all adversaries running in time at most t, making at most q_{se} and q_{ex} queries of type Send_P and $\mathsf{Execute}_P$, respectively, and at most q_{ro} random oracle queries.

Definition 2.2 (Secure multi-factor protocol) *Let κ be a security parameter. A protocol P is*

a secure multi-factor password authenticated key agreement protocol *if there exists a negligible (in κ) ϵ and small constants δ_ℓ, $\ell \in \{1,\ldots,n\}$, such that, for all polynomially-bounded adversaries $\mathcal{A}$,*

$$\mathsf{Adv}_P^{\text{ake-f}\ell}(\mathcal{A}) \leq \begin{cases} \frac{\delta_\ell q_{\mathsf{se}}}{|\mathsf{Passwords}^\ell|} + \epsilon \ , \\ \quad \textit{if the } \ell\textit{th factor is symmetric,} \\ \frac{\delta_\ell\left((1-b_{\mathsf{co}}^\ell)q_{\mathsf{se}} + b_{\mathsf{co}}^\ell q_{\mathsf{ro}}\right)}{|\mathsf{Passwords}^\ell|} + \epsilon \ , \\ \quad \textit{if the } \ell\textit{th factor is asymmetric,} \end{cases}$$

and the corresponding bound applies for $\mathsf{Adv}_P^{\text{ma-f}\ell}(\mathcal{A})$, where, for asymmetric factors ℓ, $b_{\mathsf{co}}^\ell = 1$ if $\mathcal{A}$ makes a $\mathsf{RevealFactorV}(\cdot,\cdot,\ell)$ query and 0 otherwise.

Intuitively, this notion of security says that any polynomially-bounded adversary can only do negligibly better than doing an online dictionary attack at any unknown factors and can gain no advantage by doing an offline dictionary attack. Ideally, δ_ℓ would be 1, indicating the adversary can only rule out one password with each online guess; however, a protocol can still be secure as long as δ_ℓ is small compared to $|\mathsf{Passwords}^\ell|$.

Since an instance that is fresh in all of its factors is also fresh in the original ake notion of PAKE, we have that

$$\mathsf{Adv}_P^{\text{ake}}(\mathcal{A}) \leq \min_{\ell \in \{1,\ldots,n\}} \left\{ \mathsf{Adv}_P^{\text{ake-f}\ell}(\mathcal{A}) \right\} \ .$$

By providing bounds for each factor, we can provide greater granularity in relating the security of factors to their risks of compromise. For example, lower entropy factors (represented by smaller values of $|\mathsf{Passwords}^\ell|$) may be physically distributed and secured in different ways than higher entropy factors, or may be used for a shorter period of time. This contrasts with the approach of (Pointcheval & Zimmer 2008), in which there is a single notion of freshness and a single bound over all factors.

2.2 Using one-time passwords

The model presented in Section 2.1 uses long-term authentication secrets that do not change over time. However, multi-factor authentication may include a factor that varies, such as a one-time password. Such a factor may be the response to a challenge, or may vary with time. The benefit of a one-time password is that the compromise of a single one-time password should not affect the security for a different one-time password. One-time passwords offer some protection against passive spyware, as previously compromised one-time passwords are useless.

Although at first glance it may seem impractical for a user to store a large number of passwords, this is actually quite practical and is already being done in the real world: for example, some European banks issue paper lists of one-time passwords to users (Nordea Bank 2009), and corporations issue hardware devices for pseudorandomly generating one-time passwords for virtual private network (VPN) access (RSA Security Inc. 2009) or electronic commerce (Blizzard Entertainment 2009). Even though a user may be carrying as much data as in a cryptographic key, one-time passwords offer usability benefits: carrying a cryptographic key requires a hardware interface or carefully managed private key files, whereas one-time passwords can be easily entered in only a few keystrokes.

Abdalla *et al.* (Abdalla et al. 2005) present a protocol for the use of one-time passwords in an authenticated key exchange protocol but do not alter the security model from the standard BPR setting. Paterson and Stebila (Paterson & Stebila 2009) do present an alteration to the BPR security model that accommodates the compromise of previous (and future) one-time passwords and we apply their ideas to allow for symmetric factors using one-time passwords as follows.

Adjusting the model. We can alter the security definition of a multi-factor protocol to allow a symmetric factor that corresponds to a one-time password by applying the ideas of Paterson and Stebila (Paterson & Stebila 2009). Let ℓ be the index of a symmetric factor for which we wish to use one-time passwords. Let $\mathsf{Indices}^\ell$ be the set of indices of one-time passwords, and let $\mathsf{ch} \in \mathsf{Indices}^\ell$. When a party is activated, they are activated with the index of the one-time password to use for that instance; a party can only be activated once for each $\mathsf{ch} \in \mathsf{Indices}^\ell$. Let $\{\mathsf{pw}_{C,S,\mathsf{ch}}^\ell\}$ be the set of one-time passwords between C and S, indexed by ch; each such password is chosen uniformly at random from $\mathsf{Passwords}^\ell$. We add an additional parameter ch to the $\mathsf{RevealFactor}$ query:

- $\mathsf{RevealFactor}_P(C,S,\mathsf{ch},\ell)$: Returns the ℓth factor $\mathsf{pw}_{C,S\mathsf{ch}}^\ell$ held by client C with server S for one-time password indexed by ch.

The definition of freshness in the ℓth factor of Π_i^U is adjusted as well, replacing points 3 and 4 in Definition 2.1 with:

3. if $U \in \mathsf{Clients}$: $\mathsf{RevealFactor}(U,U',\mathsf{ch},\ell)$ occurs before the Test query, and $\mathsf{Send}(U,i,M)$ occurs for some string M, where ch is the index of the one-time password with which Π_i^U was activated;
4. if $U \in \mathsf{Servers}$: $\mathsf{RevealFactor}(U',U,\mathsf{ch},\ell)$ occurs before the Test query, and $\mathsf{Send}(U,i,M)$ occurs for some string M, where ch is the index of the one-time password with which Π_i^U was activated.

The definitions of authentication are adjusted analogously as well.

Paterson and Stebila go on to show that any secure PAKE protocol can be used in the natural way to build to a secure one-time PAKE protocol, by using the one-time password in place of the password. This holds even when the one-time passwords are pseudorandomly generated or time-dependent. This means that our MFPAK protocol in the next section can easily accommodate one-time passwords as authentication factors.

3 MFPAK: a multi-factor password-authenticated key exchange protocol

MFPAK is the first PAKE protocol that uses multiple low-entropy authentication factors. It allows for an arbitrary number of factors which can be asymmetric or symmetric, and these factors can be independently changed as users need to change their passwords. Our approach is much more efficient, in terms of number of expensive operations, than the naïve approach of combining existing PAKE protocols as black boxes: we add no expensive operations for each additional symmetric factor, and only one additional expensive operation (signature generation/verification) for each party for each asymmetric factor.

3.1 Design ideas

We designed MFPAK by considering two existing one-factor protocols as our building blocks: the asymmetric password protocol PAK-Z+ for asymmetric factors, and the symmetric password protocol PAK for symmetric factors. These two protocols are similar in structure which allows us to gain some efficiency improvements. All factors are tightly inte-

grated into the authentication and key exchange processes. The underlying session key agreement comes from a hashed Diffie-Hellman construction. Authentication for asymmetric factors is done using a digital signature scheme, while for symmetric factors it is done using hash functions.

Shielded ephemeral key. One of the main efficiency and security gains in the MFPAK protocol comes in the first flow from the client to the server. In this flow, the client shields its ephemeral public key by multiplying it by (the hash of) each factor. The client is made to commit to those values, thereby preventing a malicious client from making an offline dictionary attack later on. Moreover, the server must use the same values to unshield the client's ephemeral public key or Diffie-Hellman key agreement will fail, thereby committing the server to its choice of values. By doing this multiple shielding operation, the client and server achieve mutual authentication, the client saves expensive operations compared to running multiple protocols separately, and the authentication secrets are protected.

Digital signature for asymmetric factors. Authentication for asymmetric factors comes from using a digital signature scheme, where the (short) authentication secret is used to shield the digital signature private key which is stored on the server. During the login stage of the protocol, the server returns the shielded private key, which the client can unwrap only if she knows the correct password. The client uses the private key to perform a signing operation which the server verifies using the public key. This allows for asymmetry: the compromise of the server's database is not enough to impersonate the client to the server without a dictionary attack. This technique, as used in PAK-Z+ (Gentry et al. 2005), is an instantiation of the generic technique proposed by Gentry *et al.* (Gentry et al. 2006) for asymmetric password-based authentication. It is important to note that the digital signature scheme is not used in its normal sense with published or certified public keys, but simply as a convenient asymmetric construction.

Hash function for symmetric factors. The hash of a symmetric factor is stored on the server. The server proves its knowledge of a symmetric factor by hashing it with the session key; the client does the same.

3.2 Protocol specification

The MFPAK protocol, like many other protocols, contains two stages: a user registration stage, completed once per client-server pair, and a login stage, completed each time a user attempts to login. For convenience in presentation of the login stage, we assume there is at least one symmetric factor and one asymmetric factor; however, the protocol can be altered in the natural way to deal with exclusively symmetric or exclusively asymmetric factors. The number and type of factors are fixed and publicly known.

Ingredients and notation. Let κ be a cryptographic security parameter. The notation $z \in_R Z$ denotes an element z selected uniformly at random from a set Z. Angle brackets $\langle\cdot\rangle$ denote a list, and $\cdot||\cdot$ denotes concatenation. The protocol operates over a finite cycle group G of order q, generated by g, for which the Computational Diffie-Hellman (CDH) assumption holds. The function $\mathsf{Acceptable}(\cdot)$ tests whether an element is in G (or, for efficiency reasons, a group containing G; see (MacKenzie 2002, §4)). It makes use of a number of random hash functions based on random oracles (Bellare & Rogaway 1993): H_1 maps $\{0,1\}^*$ to group elements (such as (Coron & Icart 2009) for hashing into elliptic curve groups), while all other hash functions H_i map $\{0,1\}^*$ to $\{0,1\}^\kappa$. We also employ a signature scheme $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ that is existentially unforgeable under chosen message attacks (Goldwasser et al. 1988). Let $(v, V) \leftarrow \mathsf{Gen}(1^\kappa)$, where v is a private key and V is the corresponding public key. Recall that $\mathsf{pw}^\ell_{C,S}$ denotes client C's password for server S for the ℓth factor, and $\overline{\mathsf{pw}}^\ell_{C,S}$ denotes the corresponding value held by the server, which may be equal to $\mathsf{pw}^\ell_{C,S}$ for symmetric factors and is some non-trivial transformation of $\mathsf{pw}^\ell_{C,S}$ for asymmetric factors.

The *user registration stage* of MFPAK is given in Figure 1. This stage should be completed over a private, authentic channel. The user registration stage can be altered in the obvious way to have authentication secrets chosen by the server and supplied to the client, if necessary.

The *login stage* of MFPAK is given in Figure 2. This stage can be completed over a public, untrusted channel. A client C initiates the login stage with a server S.

3.3 Nature of the factors

The MFPAK protocol can accommodate a wide variety of authentication secrets using either symmetric or asymmetric factors, as we note below. Our approach offers improved functionality compared with the nave way of combining multiple authentication secrets by simply concatenating them into one long string: with concatenation, one cannot easily combine passwords that change over time (symmetric factors) with long-term passwords (asymmetric factors) because the server does not store the plaintext password.

Long-term passwords. Long-term passwords are best accommodated as an asymmetric factor, but can be treated asymmetrically as well. Since long-term passwords do not change frequently (or at all), we should reduce the damage that can be caused by compromise of the server database containing data for these factors. Although we can never prevent dictionary attacks against the server's database, we can raise the amount of work an attacker needs to do by using asymmetric factors.

One-time passwords. One-time passwords are usually best accommodated as symmetric factors. Asymmetric factors could be used, but the costs for asymmetric factors may not be worth it for one-time passwords. It may be more efficient to generate one-time passwords from a seed using a challenge-response mechanism or a time-dependent generator. For factors that employ a challenge-response mechanism, an initial message from the server to the client conveying the challenge can be added to the beginning of the login stage of the protocol.

Cryptographic keys. Although our primary motivation has been the use of short strings as authentication secrets so users can easily carry their authentication secrets between computers, there is nothing preventing a password-based protocol from using high-entropy secrets (that is, cryptographically large keys) as opposed to low-entropy secrets. We can directly use a cryptographic key as $\mathsf{pw}^\ell_{C,S}$ in either the symmetric or asymmetric case. In the asymmetric case, it would be possible to further streamline the protocol by having the user store the private key v_ℓ from the digital signature scheme, and adjust the remainder of the protocol as follows: set $\mathsf{pw}^\ell_{C,S} \leftarrow v_\ell$; in the registration stage, the server stores $\overline{\mathsf{pw}}^\ell_{C,S} \leftarrow \langle \tau_\ell, \tau_\ell^{-1}, V_\ell \rangle$; in the login stage, the server omits steps 15 and 16 for this factor and the client omits steps 22–25 for this factor. We recommend, however, that situations using exclusively cryptographically large keys should consider traditional authenticated key exchange protocols as the security models (Canetti & Krawczyk

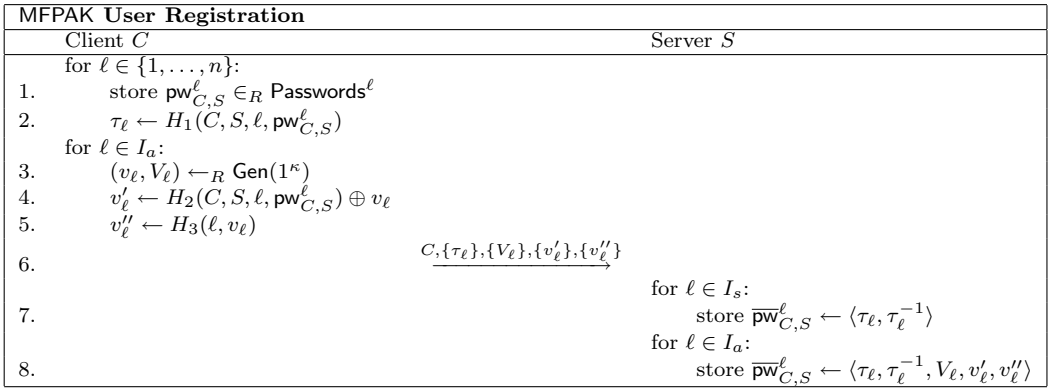

Figure 1: The user registration stage of the MFPAK protocol.

2001, LaMacchia et al. 2007) are stronger and offer resistance to ephemeral key leakage in addition to static key leakage.

Biometrics. Pointcheval and Zimmer (Pointcheval & Zimmer 2008) describe in detail the use of biometric templates in an authenticated key exchange protocol. They use secure sketches and fuzzy extractors to safely see if two biometric templates match.

An alternative approach is to use *fuzzy vaults*, which were introduced by Juels and Sudan (Juels & Sudan 2002). They allow a secret to be embedded in a *vault* which is locked by a set of fuzzy values, such as the minutiae of a fingerprint. Fuzzy vaults could for example be used in a multi-factor protocol as follows: the user receives the fuzzy vault, uses her biometric to unlock the vault, and then uses the embedded secret value as another factor in the multi-factor protocol.

Because of the privacy issues surrounding biometrics, we are not suggesting that biometrics naïvely be used in our construction immediately, as there are numerous issues to consider. For example, should the fuzzy vault be transmitted unencrypted or encrypted under the session key derived from the other factors? Should the secret embedded in the vault contain error correcting information, as suggested in (Juels & Sudan 2002), or not? (We think not, as error correcting information allows an offline "dictionary" attacker to detect whether it has the right input, whereas lack of error correction information would ideally mean the attacker needs to do an online "dictionary" attack.) The use of biometrics in authenticated key exchange merits further study.

3.4 Efficiency

In many e-commerce and online banking situations, the performance-limiting factor is the number of connections a server can handle, and this is in turn limited by the number of expensive operations required by the cryptographic protocol. MFPAK can increase security without a substantial additional computational burden on the server.

Figure 3 compares the number of expensive operations (group exponentiations and signature generation / verification) performed by a naïve combination of PAK and PAK-Z+ versus the MFPAK protocol. MFPAK has a fixed overhead of two group exponentiations each on client and server side. For each symmetric factor, there are no additional expensive operations (only multiplications and hashes, not exponentiations); for each asymmetric factor, there is one additional expensive operation on each side (signature generation by the client, signature verification by the server). This makes MFPAK much more efficient, in terms of number of expensive operations, than if one were to make a multi-factor scheme simply by running PAK and PAK-Z+ in parallel independently.

3.5 Security analysis

The main idea of the security argument is that, if one factor, say the ℓ^*th factor, remains uncompromised, then the difficulty of breaking MFPAK is related to the difficulty of breaking the corresponding one of either PAK (for a symmetric factor) or PAK-Z+ (for an asymmetric factor), each which is in turn related to solving the Computational Diffie-Hellman problem.

For both symmetric and asymmetric factors, we describe a procedure (specified by a modifier $\mathcal{M}$) to transform an adversary $\mathcal{A}$ against MFPAK with the ℓ^*th factor uncompromised into an adversary $\mathcal{A}^*$ against the corresponding one of the two underlying protocols (PAK and PAK-Z+, respectively). The transformations are such that, if the oracle instance in MFPAK against which the Test query is directed is fresh in the ℓ^*th factor, then the corresponding oracle instance is also fresh in the corresponding attack on PAK (resp., PAK-Z+). This is possible because of the design of the MFPAK protocol: it essentially runs both PAK and PAK-Z+ together while still capturing the security of each independently. This design characteristic allows the relatively straightforward (although lengthy) security argument.

Our formal argument proceeds by considering four cases, two corresponding to an asymmetric factor being uncompromised and two corresponding a symmetric factor being uncompromised. The cases are:

1. Asymmetric factor uncompromised, $U^* \in$ Clients: no $\mathsf{RevealFactor}_{\mathsf{MFPAK}}(U^*, U'^*, \ell^*)$ or $\mathsf{RevealFactorV}_{\mathsf{MFPAK}}(U'^*, U^*, \ell^*)$ query.
2. Asymmetric factor uncompromised, $U^* \in$ Servers: no $\mathsf{RevealFactor}_{\mathsf{MFPAK}}(U'^*, U^*, \ell^*)$ query.
3. Symmetric factor uncompromised, $U^* \in$ Clients: no $\mathsf{RevealFactor}_{\mathsf{MFPAK}}(U^*, U'^*, \ell^*)$ query.
4. Symmetric factor uncompromised, $U^* \in$ Servers: no $\mathsf{RevealFactor}_{\mathsf{MFPAK}}(U'^*, U^*, \ell^*)$ query.

These four cases are combined probabilistically to give the overall result. The details are provided in Appendix A. Throughout, we assume passwords are uniformly distributed. The resulting security statement is as follows:

Theorem 3.1 *Let κ be a security parameter. Let G be a finite cyclic group generated by g and let S be a signature scheme. Let $\mathcal{A}$ be an adversary that runs in time polynomial in κ, 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 oracle. If ℓ is an asymmetric factor, then let $b_{\mathsf{co}} = 1$ if $\mathcal{A}$ makes a* $\mathsf{RevealFactorV}(\cdot, \cdot, \ell)$ *query to a server, and 0 otherwise. Then* MFPAK *is a secure multi-factor PAKE*

MFPAK **Login**			
	Client C		Server S
1.	$x \in_R \mathbb{Z}_q$		
2.	$X \leftarrow g^x$		
	for $\ell \in \{1, \ldots, n\}$:		
3.	$\quad \tau_\ell \leftarrow H_1(C, S, \ell, \mathsf{pw}^\ell_{C,S})$		
4.	$m \leftarrow X \cdot \prod_{\ell=1}^n \tau_\ell$		
		$\xrightarrow{C,m}$	
5.			
6.			reject if $\neg\mathsf{Acceptable}(m)$
7.			$y \in_R \mathbb{Z}_q$
8.			$Y \leftarrow g^y$
			for $\ell \in I_s$:
9.			$\quad$ lookup $\langle \tau_\ell, \tau_\ell^{-1} \rangle \leftarrow \overline{\mathsf{pw}}^\ell_{C,S}$
			for $\ell \in I_a$:
10.			$\quad$ lookup $\langle \tau_\ell, \tau_\ell^{-1}, V_\ell, v'_\ell, v''_\ell \rangle \leftarrow \overline{\mathsf{pw}}^\ell_{C,S}$
11.			$X \leftarrow m \cdot \prod_{\ell=1}^n \tau_\ell^{-1}$
12.			$\sigma \leftarrow X^y$
13.			$\mathsf{sid} \leftarrow \langle C, S, m, Y \rangle$
14.			$k \leftarrow H_4(\mathsf{sid}, \sigma, \tau_1, \ldots, \tau_n)$
			for $\ell \in I_a$:
15.			$\quad a'_\ell \leftarrow H_5(\mathsf{sid}, \sigma, \ell, \tau_\ell)$
16.			$\quad a_\ell \leftarrow a'_\ell \oplus v'_\ell$
		$\xleftarrow{Y,k,\{a_\ell\},\{v''_\ell\}}$	
17.			
18.	$\sigma \leftarrow Y^x$		
19.	$\mathsf{sid} \leftarrow \langle C, S, m, Y \rangle$		
20.	reject if $k \neq H_4(\mathsf{sid}, \sigma, \tau_1, \ldots, \tau_n)$		
21.	$k' \leftarrow H_6(\mathsf{sid}, \sigma, \tau_1, \ldots, \tau_n)$		
	for $\ell \in I_a$:		
22.	$\quad a'_\ell \leftarrow H_5(\mathsf{sid}, \sigma, \ell, \tau_\ell)$		
23.	$\quad v'_\ell \leftarrow a'_\ell \oplus a_\ell$		
24.	$\quad v_\ell \leftarrow H_2(C, S, \ell, \mathsf{pw}_{C,S,\ell}) \oplus v'_\ell$		
25.	$\quad$ reject if $v''_\ell \neq H_3(\ell, v_\ell)$		
26.	$\quad s_\ell \leftarrow \mathsf{Sign}_{v_\ell}(\mathsf{sid})$		
		$\xrightarrow{k',\{s_\ell\}}$	
27.			
28.			reject if $k' \neq H_6(\mathsf{sid}, \sigma, \tau_1, \ldots, \tau_n)$
			for $\ell \in I_a$:
29.			$\quad$ reject if $\neg\mathsf{Verify}_{V_\ell}(\mathsf{sid}, s_\ell)$
30.	$\mathsf{sk} \leftarrow H_7(\mathsf{sid}, \sigma, \tau_1, \ldots, \tau_n)$		$\mathsf{sk} \leftarrow H_7(\mathsf{sid}, \sigma, \tau_1, \ldots, \tau_n)$

Figure 2: The login stage of the MFPAK protocol.

Operation	PAK & PAK-Z+		MFPAK	
	Client	Server	Client	Server
exponentiations	$2\|I_s\| + 2\|I_a\|$	$2\|I_s\| + 2\|I_a\|$	2	2
signature generation	$\|I_a\|$	0	$\|I_a\|$	0
signature verification	0	$\|I_a\|$	0	$\|I_a\|$
total	$2\|I_s\| + 3\|I_a\|$	$2\|I_s\| + 3\|I_a\|$	$2 + \|I_a\|$	$2 + \|I_a\|$

Figure 3: Comparison of expensive operations for combined PAK & PAK-Z+ and MFPAK.

protocol, with

$$\mathsf{Adv}^{\text{ake-f}\ell}_{\mathsf{MFPAK}}(\mathcal{A}) \leq \begin{cases} \frac{16\delta((1-b_{\mathsf{co}})q_{\mathsf{se}}+b_{\mathsf{co}}q_{\mathsf{ro}})}{|\mathsf{Passwords}^\ell|} + \epsilon \ , \\ \quad \textit{if the } \ell\textit{th factor is symmetric,} \\ \frac{4\delta q_{\mathsf{se}}}{|\mathsf{Passwords}^\ell|} + \epsilon \ , \\ \quad \textit{if the } \ell\textit{th factor is asymmetric,} \end{cases}$$

where ϵ is a negligible function of κ, and $\delta = |\mathsf{Clients}| \cdot |\mathsf{Servers}|$; a similar bound exists for $\mathsf{Adv}^{\text{ma-f}\ell}_{\mathsf{MFPAK}}(\mathcal{A})$.

As with any formal security argument, a proof of security does not imply security against all forms of attack. A protocol may be vulnerable to attack methods not described by the security model. Nonetheless, a security proof is valuable as a heuristic that the protocol is resistant to at least some types of attacks.

4 Conclusion and future work

We have presented a security model for multi-factor password-authenticated key exchange protocols that can accommodate an arbitrary number of factors. We have provided a security argument showing that our new protocol, MFPAK, is secure in this model. Our multi-factor authentication protocol offers enhanced authentication protection through the use of complementary factors, such as a long-term password and a one-time challenge/response. The construction is quite efficient in terms of the number of operations per factor; for example, a two-factor version of our protocol using a long-term password and one-time challenge/response has the same efficiency as the one-factor protocol PAK-Z+. The protocol remains secure even if all but one of the authentication factors is fully known to an adversary. Our multi-factor protocol is resistant to man-in-the-middle and impersonation attacks, providing enhanced authentication in the face of more complex threats like phishing.

Other recent work in the field of PAKE protocols has focused on protocols where the sequence of flows fits existing network protocols such as SSL/TLS. An open question is to design a provably secure multi-factor PAKE protocol with support for asymmetric factors that fits within the message flow of SSL/TLS.

Additionally, multi-factor protocols supporting an

arbitrary number of factors could be designed where some factors are optional and the number of factors used corresponds to differing levels of access depending on the application situation: one factor could be used for read-only access, two factors for small-value transactions, and three factors for large-value transactions.

An interesting future direction would be to further investigate the use of biometric information in a multi-factor authenticated key exchange protocol. We have outlined some ideas involving fuzzy vaults, but consideration of the privacy and security requirements requires further research.

Acknowledgements

This research performed while D.S. was at the University of Waterloo and S.C. was at Sun Microsystems Laboratories. D.S. was supported in part by an NSERC Canada Graduate Scholarship. The authors gratefully acknowledge helpful discussions with Alfred Menezes, Bodo Möller, Michele Mosca, and Berkant Ustaoglu, and appreciate the feedback of anonymous referees.

References

Abdalla, M., Chevassut, O. & Pointcheval, D. (2005), One-time verifier-based encrypted key exchange, in (Vaudenay 2005), pp. 47–64. Full version available as **URL:** *http://www.di.ens.fr/∼mabdalla/papers/ACP05-letter.pdf*

Bellare, M., Pointcheval, D. & Rogaway, P. (2000), Authenticated key exchange secure against dictionary attacks, in (Preneel 2000), pp. 139–155.

Bellare, M. & Rogaway, P. (1993), Random oracles are practical: a paradigm for designing efficient protocols, in *Proc. 1st ACM Conference on Computer and Communications Security (CCS)* (1993), ACM, pp. 62–73.

Bellovin, S. M. & Merritt, M. (1992), 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.

Bellovin, S. M. & Merritt, M. (1993), 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)* (1993), pp. 244–250.

Blizzard Entertainment (2009), Blizzard authenticator.
URL: *http://eu.blizzard.com/support/article.xml?locale=en_GB&articleId=28152*

Boyko, V., MacKenzie, P. & Patel, S. (2000), Provably secure Password-Authenticated Key exchange using Diffie-Hellman, in (Preneel 2000), pp. 156–171. Full version available as **URL:** *http://eprint.iacr.org/2000/044*

Canetti, R., Halevi, S., Katz, J., Lindell, Y. & MacKenzie, P. (2005), Universally composable password-based key exchange, in R. Cramer, ed., *Advances in Cryptology – Proc. EUROCRYPT 2005*, Vol. 3494 of *LNCS*, Springer, pp. 404–421.

Canetti, R. & Krawczyk, H. (2001), Analysis of key-exchange protocols and their use for building secure channels, in B. Pfitzmann, ed., *Advances in Cryptology – Proc. EUROCRYPT 2001*, Vol. 2045 of *LNCS*, Springer, pp. 453–474. Full version available as **URL:** *http://eprint.iacr.org/2001/040*

Coron, J.-S. & Icart, T. (2009), A random oracle into elliptic curves. **URL:** *http://eprint.iacr.org/2009/340*

Gentry, C., MacKenzie, P. & Ramzan, Z. (2005), PAK-Z+. Contribution to the IEEE P1363-2000 study group for Future PKC Standards.
URL: *http://grouper.ieee.org/groups/1363/WorkingGroup/presentations/pakzplusv2.pdf*

Gentry, C., MacKenzie, P. & Ramzan, Z. (2006), A method for making password-based key exchange resilient to server compromise, in C. Dwork, ed., *Advances in Cryptology – Proc. CRYPTO 2006*, Vol. 4117 of *LNCS*, Springer, pp. 142–159.

Goldwasser, S., Micali, S. & Rivest, R. L. (1988), A digital signature scheme secure against adaptive chosen-message attacks, *SIAM Journal on Computing* **17**(2), 281–308.

Juels, A. & Sudan, M. (2002), A fuzzy vault scheme, in *Proc. IEEE International Symposium on Information Theory (ISIT) 2002*, IEEE Press, p. 408. Full version available as **URL:** *http://www.rsa.com:80/rsalabs/node.asp?id=2061*

LaMacchia, B., Lauter, K. & Mityagin, A. (2007), Stronger security of authenticated key exchange, *in* W. Susilo, J. K. Liu & Y. Mu, eds, *First International Conference on Provable Security (ProvSec) 2007*, Vol. 4784 of *LNCS*, Springer, pp. 1–16.

MacKenzie, P. (2002), The PAK suite: Protocols for password-authenticated key exchange, Technical Report 2002-46, DIMACS Center, Rutgers University.
URL: *http://dimacs.rutgers.edu/TechnicalReports/abstracts/2002/2002-46.html*

Nordea Bank (2009), Netbank security.
URL: *http://www.nordea.ee/Private+customers/E-channels++Netbank/Netbank/Netbank+Security/936612.html*

Park, Y. M. & Park, S. G. (2004), Two factor authenticated key exchange (TAKE) protocol in public wireless LANs, *IEICE Transactions on Communications* **E87-B**(5), 1382–1385.

Paterson, K. G. & Stebila, D. (2009), One-time-password-authenticated key exchange. **URL:** *http://eprint.iacr.org/2009/430*

Pointcheval, D. & Zimmer, S. (2008), Multi-factor authenticated key exchange, in S. M. Bellovin & R. Gennaro, eds, *Applied Cryptography and Network Security (ACNS) 2008*, Vol. 5037 of *LNCS*, Springer, pp. 277–295.

Preneel, B., ed. (2000), *Advances in Cryptology – Proc. EUROCRYPT 2000*, Vol. 1807 of *LNCS*, Springer.

RSA Security Inc. (2009), RSA SecurID.
URL: *http://www.rsa.com/node.aspx?id=1156*

Schecter, S., Dhamija, R., Ozment, A. & Fischer, I. (2007), The emperor's new security indicators: An evaluation of website authentication and the effect of role playing on usability studies, in *Proc. IEEE Symposium on Security and Privacy (S&P) 2007*, IEEE Press, pp. 51–65.

Stebila, D., Udupi, P. & Chang, S. (2009), Multi-factor password-authenticated key exchange (full version).
URL: *http://eprint.iacr.org/2008/214*

Vaudenay, S., ed. (2005), *Public Key Cryptography (PKC) 2005*, Vol. 3386 of *LNCS*, Springer.

Yang, G., Wong, D. S., Wang, H. & Deng, X. (2006), Formal analysis and systematic construction of two-factor authentication scheme (short paper), in P. Ning, S. Qing & N. Li, eds, *Proc. 8th International Conference on Information and Communications Security (ICICS) 2006*, Vol. 4307 of *LNCS*, Springer, pp. 82–91. Full version available as **URL:** *http://eprint.iacr.org/2006/270*

Yoon, E.-J. & Yoo, K.-Y. (2006), An optimized two factor authenticated key exchange protocol in PWLANs, *in* V. N. Alexandrov, G. D. van Albada, P. M. Sloot & J. Dongarra, eds, *Computational Science – ICCS 2006*, Vol. 3992 of *LNCS*, Springer, pp. 1000–1007.

A Security analysis

This section contains the details of the security analysis supporting Theorem 3.1.

It is helpful to be able to refer to the action of a party upon receipt of a message. We use the notation $\textsc{ClientAction}i_P$ and $\textsc{ServerAction}i_P$ to refer to the portion of the protocol P performed by the client or server, respectively, after the ith flow. Thus, MFPAK as described in Figure 2 specifies $\textsc{ClientAction}0_{\mathsf{MFPAK}}$, $\textsc{ServerAction}1_{\mathsf{MFPAK}}$, $\textsc{ClientAction}2_{\mathsf{MFPAK}}$, and $\textsc{ServerAction}3_{\mathsf{MFPAK}}$

A.1 Ingredients

Computational Diffie-Hellman assumption. MFPAK operates over a finite cycle group G for which the Computational Diffie-Hellman (CDH) assumption holds. Let G be a finite cyclic group of order q, let g be a generator of G, and let t_{exp} be the time it takes to perform an exponentiation in 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. For two values X and Y, define $\mathsf{DH}(X, Y) = X^y$, if $\mathsf{Acceptable}(X)$ and $Y = g^y$, or $\mathsf{DH}(X, Y) = Y^x$, if $\mathsf{Acceptable}(Y)$ and $X = g^x$. Let $\mathcal{A}$ be a probabilistic algorithm with input (G, g, X, Y) that outputs a subset of G, and define

$$\mathsf{Adv}^{\mathrm{cdh}}_{G,g}(\mathcal{A}) = \Pr\left(\mathsf{DH}(X, Y) \in \mathcal{A}(G, g, X, Y) : (x, y) \in_R \mathbb{Z}_q, X = g^x, Y = g^y\right) .$$

Let $\mathsf{Adv}^{\mathrm{cdh}}_{G,g}(t, n) = \max_{\mathcal{A}}\{\mathsf{Adv}^{\mathrm{cdh}}_{G,g}(\mathcal{A})\}$ where the maximum is taken over all algorithms running in time t and outputting a subset of size at most n. The CDH assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{cdh}}_{G,g}(\mathcal{A})$ is negligible.

Random hash functions. MFPAK makes use of a number of random hash functions based on random oracles (Bellare & Rogaway 1993). A *random hash function* $H : \{0,1\}^* \to \{0,1\}^k$ is constructed by selecting each bit of $H(x)$ uniformly at random and independently for every $x \in \{0,1\}^*$. We make use of a number of independent random hash functions $H_1, H_2, \ldots$, which can be constructed from a single random hash function H by setting $H_\ell(x) = H(\ell||x)$. Constructing a hash function that outputs elements of a group instead of $\{0,1\}^*$ is also possible and efficient, and in fact all of the hash functions used in MFPAK are into the group G.

Signature scheme. MFPAK makes use of a signature scheme $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ that is existentially unforgeable under chosen message attacks (Goldwasser et al. 1988). Let $(v, V) \leftarrow \mathsf{Gen}(1^\kappa)$, where v is a private key and V is the corresponding public key. Let t_{Gen} be the runtime of $\mathsf{Gen}(1^\kappa)$, and t_{sig} be the runtime of Sign and Verify. A forger $\mathcal{F}$ is given a public key V and must forge signatures; it can query an oracle that returns $\mathsf{Sign}_v(m)$ for any messages m of its choice. It succeeds if it can output a forgery (m, σ) such that $\mathsf{Verify}_V(m, \sigma) = \mathsf{true}$, where m was not queried to the signing oracle. Let $\mathsf{Succ}^{\text{eu-cma}}_{\mathcal{S},\kappa}(\mathcal{F}) = \Pr(\mathcal{F} \text{ succeeds})$, and $\mathsf{Succ}^{\text{eu-cma}}_{\mathcal{S},\kappa}(t, q_{\mathsf{Sign}}) = \max_{\mathcal{F}}\left\{\mathsf{Succ}^{\text{eu-cma}}_{\mathcal{S},\kappa}(\mathcal{F})\right\}$ where the maximum is taken over all forgers running in time t and making at most q_{Sign} queries to the signing oracle. A signature scheme $\mathcal{S}$ is *existentially unforgeable under chosen message attacks (eu-cma)* if, for any probabilistic polynomial time algorithm $\mathcal{F}$, $\mathsf{Succ}^{\text{eu-cma}}_{\mathcal{S},\kappa}(\mathcal{F})$ is negligible.

A.2 Case 1: Attacking a client instance, asymmetric factor uncompromised

This case addresses impersonation of the server when the instance being attacked is a client instance and the uncompromised ℓ^*th factor is asymmetric.

The modifier $\mathcal{M}$ first uniformly at random guesses $U^* \in_R \mathsf{Clients}$ and $U'^* \in_R \mathsf{Servers}$ as its guess of who the adversary $\mathcal{A}$ will end up attacking. If the attacker ends up attacking the pair of users the modifier has guessed, then we will show how to transform the attack into an attack on PAK-Z+.

Let GuessCS be the event that the modifier $\mathcal{M}$ correctly guesses U^* and U'^*. Then

$$\Pr(\mathsf{GuessCS}) = \Pr((U^* \text{ correct}) \wedge (U'^* \text{ correct})) \quad (1)$$

$$\geq \frac{1}{|\mathsf{Clients}| \cdot |\mathsf{Servers}|} . \quad (2)$$

For this case, we assume that no $\mathsf{RevealFactor}_{\mathsf{MFPAK}}(U^*, U'^*, \ell^*)$ or $\mathsf{RevealFactorV}_{\mathsf{MFPAK}}(U'^*, U^*, \ell^*)$ query is issued against $\mathcal{M}$: this case models server impersonation in the ℓ^*th factor, which is why no $\mathsf{RevealFactorV}_{\mathsf{MFPAK}}(U'^*, U^*, \ell^*)$ query is allowed. Furthermore, no $\mathsf{RevealFactor}_{\mathsf{MFPAK}}(U^*, U'^*, \ell^*)$ is allowed because an adversary can easily recover the verifier $\overline{\mathsf{pw}}^{\ell^*}_{U^*,U'^*}$ from the secret $\mathsf{pw}^{\ell^*}_{U^*,U'^*}$ and one interaction with U'^*.

The modifier $\mathcal{M}$ does the following to convert an MFPAK adversary $\mathcal{A}$ into a PAK-Z+ adversary $\mathcal{A}^*$.

Password preparation. For each $(C, S, \ell) \in \mathsf{Clients} \times \mathsf{Servers} \times \{1, \ldots, n\} \setminus \{(U^*, U'^*, \ell^*)\}$, $\mathcal{M}$ sets $\mathsf{pw}^\ell_{C,S} \in_R \mathsf{Passwords}^\ell$ and constructs the corresponding $\overline{\mathsf{pw}}^\ell_{C,S}$. Of all the authentication secrets, only $\mathsf{pw}^{\ell^*}_{U^*,U'^*}$ and $\overline{\mathsf{pw}}^{\ell^*}_{U^*,U'^*}$ remain unknown to $\mathcal{M}$ at this point. Compute the corresponding τ_ℓ, for $\ell \neq \ell^*$, and set $\pi \leftarrow \prod_{\ell=1,\ell\neq\ell^*}^{n} \tau_\ell$.

Instantiation of PAK-Z+ simulator. We instantiate the PAK-Z+ simulator $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$ with the following random oracles: $H^*_i(C, S, \mathsf{pw}_{C,S}) := H_i(C, S, \ell^*, \mathsf{pw}_{C,S})$ for $i = 1, 2$; $H^*_3(v) := H_3(\ell^*, v)$;

$$H^*_4(\langle C,S,m,Y\rangle, \sigma, \tau^{-1}) := H_4(\langle C, S, m \cdot \pi, Y\rangle, \sigma, \tau_1, \ldots, \tau, \ldots, \tau_n) \| _{\ell\in I_a,\ell\neq\ell^*} H_5(\langle C, S, m \cdot \pi, Y\rangle, \sigma, \ell, \tau_\ell) ;$$

$H^*_5(\langle C, S, m, Y\rangle, \sigma, \tau^{-1}) := H_5(\langle C, S, m \cdot \pi, Y\rangle, \sigma, \ell^*, \tau)$; and $H^*_7(\langle C, S, m, Y\rangle, \sigma, \tau^{-1}) := H_7(\langle C, S, m \cdot \pi, Y\rangle, \sigma, \tau_1, \ldots, \tau, \ldots, \tau_n)$.[1] These 'starred' functions are independent random oracles if the corresponding unstarred functions are. The above construction is possible since $\{\tau_\ell\}_{\ell\neq\ell^*}$ and π are fixed and known to $\mathcal{M}$ because of the guesses made at the beginning of this case. By using a concatenation of random oracles, the PAK system computes the values we need in $\mathcal{M}$'s handling of Execute and Send queries.

Further, $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$ is instantiated with the following signature scheme $(\mathsf{Gen}, \mathsf{Sign}^*, \mathsf{Verify}^*)$:

$$\mathsf{Sign}^*_v(\langle C, S, m, Y\rangle) := \mathsf{Sign}_v(\langle C, S, m \cdot \pi, Y\rangle)$$
$$\mathsf{Verify}^*_V(\langle C, S, m, Y\rangle, s) := \mathsf{Verify}_V(\langle C, S, m \cdot \pi, Y\rangle, s).$$

Since the transformation that sends $\langle C, S, m, Y\rangle \mapsto \langle C, S, m \cdot \pi, Y\rangle$ is just a permutation, it follows that $(\mathsf{Gen}, \mathsf{Sign}^*, \mathsf{Verify}^*)$ is an eu-cma signature scheme whenever $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is.

[1] Note that we do not need to instantiate H^*_6 because this oracle is not used by PAK-Z+.

$\mathcal{M}$'s handling of $\mathcal{A}$'s queries. The modifier $\mathcal{M}$ performs the following modifications to the queries of $\mathcal{A}$. The main goal is for $\mathcal{M}$ to simulate all queries except for ones that are related to the U^* and U'^* guessed at the beginning of the case: these queries are passed to the underlying PAK-Z+ simulator $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$.

RevealFactor(C, S, ℓ):

1. If $(C, S, \ell) \neq (U^*, U'^*, \ell^*)$:
   Return $\mathsf{pw}^{\ell}_{C,S}$.
2. If $(C, S, \ell) = (U^*, U'^*, \ell^*)$:
   Reject; if this query occurs, then $\mathcal{M}$'s guess of U^* and U'^* at the beginning of this case was incorrect.

RevealFactorV(S, C, ℓ):

1. If $(C, S, \ell) \neq (U^*, U'^*, \ell^*)$:
   Return $\overline{\mathsf{pw}}^{\ell}_{C,S}$.
2. If $(C, S, \ell) = (U^*, U'^*, \ell^*)$:
   Reject; if this query occurs, then $\mathcal{M}$'s guess of U^* and U'^* at the beginning of this case was incorrect.

Test(U, i):

1. If $U = U^*$:
   Send a $\mathsf{Test}_{\mathsf{PAK\text{-}Z+}}(U, i)$ query to PAK-Z+ simulator $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$ and return the result to $\mathcal{A}$.
2. If $U \neq U^*$:
   Reject; if this query occurs, then $\mathcal{M}$'s guess of U^* at the beginning of this case was incorrect.

RevealSK(U, i):

1. If $U = U^*$ or $U = U'^*$:
   Send a $\mathsf{RevealSK}_{\mathsf{PAK\text{-}Z+}}(U, i)$ query to PAK-Z+ simulator $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$ and return the result to $\mathcal{A}$.
2. Otherwise:
   Return sk for instance Π^U_i.

Execute(C, i, S, j):

1. If $(C, S) \neq (U^*, U'^*)$:
   $\mathcal{M}$ performs $\mathsf{Execute}_{\mathsf{MFPAK}}(C, i, S, j)$ with all the values it has and returns the transcript.
2. If $(C, S) = (U^*, U'^*)$:
   $\mathcal{M}$ will use the PAK-Z+ simulator $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$ to obtain a transcript for this query.
   (a) Send an $\mathsf{Execute}_{\mathsf{PAK\text{-}Z+}}(C, i, S, j)$ query to $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$ and receive $\langle C, m, Y, k, a, v'', s\rangle$.
   (b) Set $\hat{m} \leftarrow m \cdot \pi$.
   (c) Set $\hat{k'} \in_R \text{range}(H_6)$.
   (d) Extract $\hat{k}$ as the first component of k.
   (e) Extract $\{a'_\ell\}_{\ell \in I_a, \ell \neq \ell^*}$ from the remaining $|I_a| - 1$ components of k.
   (f) Compute $\{a_\ell\}_{\ell \in I_a, \ell \neq \ell^*}$.
   (g) Set $a_{\ell^*} \leftarrow a$.
   (h) Set $v''_{\ell^*} \leftarrow v''$.
   (i) Compute $\{s_\ell\}_{\ell \in I_a, \ell \neq \ell^*}$.
   (j) Set $s_{\ell^*} \leftarrow s$.
   (k) Return $\langle C, \hat{m}, Y, \hat{k}, \{a_\ell\}, \{v''_\ell\}, \hat{k'}, \{s_\ell\}\rangle$ to $\mathcal{A}$.

Send(U, i, M):

If M is not a valid protocol message in a meaningful sequence, then reject as would be done in MFPAK.

1. If $M = \langle$"start"$, S\rangle$ and $(U, S) \neq (U^*, U'^*)$:
   Perform ClientAction0$_{\mathsf{MFPAK}}$ and return $\langle U, m\rangle$.
2. If $M = \langle$"start"$, S\rangle$ and $(U, S) = (U^*, U'^*)$:
   (a) Send a $\mathsf{Send}_{\mathsf{PAK\text{-}Z+}}(U, i, M)$ query to $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$ and receive $\langle U, m\rangle$.
   (b) Set $\hat{m} \leftarrow m \cdot \pi$.
   (c) Return $\langle U, \hat{m}\rangle$.
3. If $M = \langle C, m\rangle$ and $(C, U) \neq (U^*, U'^*)$:
   Perform ServerAction1$_{\mathsf{MFPAK}}$ and return $\langle Y, k, \{a_\ell\}, \{v''_\ell\}\rangle$.
4. If $M = \langle C, m\rangle$ and $(C, U) = (U^*, U'^*)$:
   (a) Set $\hat{m} \leftarrow m \cdot \pi^{-1}$.
   (b) Send a $\mathsf{Send}_{\mathsf{PAK\text{-}Z+}}(U, i, \langle C, \hat{m}\rangle)$ query to $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$ and receive $\langle Y, k, a, v''\rangle$.
   (c) Extract $\hat{k}$ as the first component of k.
   (d) Extract $\{a'_\ell\}_{\ell \in I_a, \ell \neq \ell^*}$ from the remaining $|I_a| - 1$ components of k.
   (e) Compute $\{a_\ell\}_{\ell \in I_a, \ell \neq \ell^*}$.
   (f) Set $a_{\ell^*} \leftarrow a$.
   (g) Set $v''_{\ell^*} \leftarrow v''$.
   (h) Return $\langle C, \hat{m}, Y, \hat{k}, \{a_\ell\}, \{v''_\ell\}\rangle$.
5. If $M = \langle Y, k, \{a_\ell\}, \{v''_\ell\}\rangle$ and $(U, U') \neq (U^*, U'^*)$, where U' is the partner of U:
   Perform ClientAction2$_{\mathsf{MFPAK}}$ and return $\langle k', \{s_\ell\}\rangle$.
6. If $M = \langle Y, k, \{a_\ell\}, \{v''_\ell\}\rangle$ and $(U, U') = (U^*, U'^*)$, where U' is the partner of U:
   (a) Verify $\{v''_\ell\}_{\ell \in I_a, \ell \neq \ell^*}$.
   (b) Set $\hat{k'} \leftarrow k \| _{\ell \in I_a, \ell \neq \ell^*} a'_\ell$.
   (c) Send a $\mathsf{Send}_{\mathsf{PAK\text{-}Z+}}(U, i, \langle Y, \hat{k}, a_{\ell^*}, v''_{\ell^*}\rangle)$ query to $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$ and receive $\langle s\rangle$.
   (d) Set $\hat{k'} \in_R \text{range}(H_6)$ and store.
   (e) Compute $\{s_\ell\}_{\ell \in I_a, \ell \neq \ell^*}$.
   (f) Set $s_{\ell^*} \leftarrow s$.
   (g) Return $\langle \hat{k'}, \{s_\ell\}\rangle$.
7. If $M = \langle k', \{s_\ell\}\rangle$ and $(U', U) \neq (U^*, U'^*)$, where U' is the partner of U:
   Perform ServerAction3$_{\mathsf{MFPAK}}$.
8. If $M = \langle k', \{s_\ell\}\rangle$ and $(U', U) = (U^*, U'^*)$, where U' is the partner of U:
   (a) Reject if k' is not the same as the $\hat{k'}$ generated in Case 6 above.
   (b) Verify $\{s_\ell\}_{\ell \in I_a, \ell \neq \ell^*}$.
   (c) Send a $\mathsf{Send}_{\mathsf{PAK\text{-}Z+}}(U, i, \langle s_{\ell^*}\rangle)$ query to $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$.

Differences from MFPAK simulator. We must now analyze the differences between a true MFPAK simulator and the view presented to the MFPAK adversary $\mathcal{A}$ by the modifier $\mathcal{M}$.

First we note that the distributions of generated passwords exactly match the MFPAK specifications. Furthermore, all the generated passwords exactly match the PAK-Z+ specifications.

Next, we note that $\mathcal{M}$'s handling of $\mathcal{A}$'s queries precisely matches what an MFPAK simulator would do except in a small number of cases. The messages received from and forwarded from the use of the PAK-Z+ simulator $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$ can by inspection be seen to match what the MFPAK simulator would do because $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$ is using the specially constructed random oracles H^*_i. The differences between $\mathcal{M}$ and what a true MFPAK simulator would do are as follows:

- RevealFactor(C, S, ℓ) when $(C, S, \ell) = (U^*, U'^*, \ell^*)$, RevealFactorV$(S, C, \ell)$ when $(C, S, \ell) = (U^*, U'^*, \ell^*)$, and Test$(U, i)$ when $U \neq U^*$:
  The modifier $\mathcal{M}$ rejects here, while a true MFPAK simulator should not. If $\mathcal{M}$ correctly guessed U^* and U'^* at the beginning of this case, then none of these queries would occur, for if one did then the instance in which a Test query is directed to $\Pi^{U^*}_i$ would not be fresh in the ℓ^*th factor.
- Execute(C, i, S, j) when $(C, S) = (U^*, U'^*)$, Send(U, i, M) when $M = \langle Y, k, a, v''\rangle$ and $(U, U') = (U^*, U'^*)$, where U' is the partner of U, and Send(U, i, M) when $M = \langle k', s\rangle$ and $(U, U') = (U'^*, U^*)$, where U' is the partner of U:

The modifier $\mathcal{M}$ generated a random value $\hat{k}'$ for this instance instead of generating $k' = H_6(\mathsf{sid}, \sigma, \tau_1, \ldots, \tau_n)$. Since H_6 is a random oracle, this substitution is distinguishable by the adversary $\mathcal{A}$ if and only if $\mathcal{A}$ queries H_6 on the arguments $\mathsf{sid}, \sigma, \tau_1, \ldots, \tau_n$. But if that occurs, then $\mathcal{A}$ must know τ_{ℓ^*}. These are the same inputs to the H_7^* oracle used to compute the session key in the PAK-Z+ simulation $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$, so the same adversary could distinguish the output of $\mathsf{Test}_{\mathsf{PAK\text{-}Z+}}(U^*, i)$ received from $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$. The latter event corresponds to the event $\mathsf{Succ}^{\mathrm{ake}}_{\mathsf{PAK\text{-}Z+}}$, and so the substitution is distinguishable with probability at most $\Pr(\mathsf{Succ}^{\mathrm{ake}}_{\mathsf{PAK\text{-}Z+}}(\mathcal{A}))$.

Let $\mathsf{Dist}_1|\mathsf{GuessCS}$ be the event that the simulation $\mathcal{M}$ is distinguishable from a real MFPAK simulator from $\mathcal{A}$'s perspective given that the modifier correctly guessed U^* and U'^* at the beginning of this case. Then $\Pr(\mathsf{Dist}_1|\mathsf{GuessCS}) \leq 3\Pr(\mathsf{Succ}^{\mathrm{ake}}_{\mathsf{PAK\text{-}Z+}}(\mathcal{A}))$ by the argument above.

Result for case 1. Let $U^* \in \mathsf{Clients}$, $U'^* \in \mathsf{Servers}$ and let E_1 be the event that neither $\mathsf{RevealFactor}_{\mathsf{MFPAK}}(U^*, U'^*, \ell^*)$ nor $\mathsf{RevealFactorV}_{\mathsf{MFPAK}}(U'^*, U^*, \ell^*)$ occurs. The instance involving U^*, U'^* in $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$ is fresh if and only if the corresponding instance in $\mathcal{M}$ is fresh in the ℓ^*th factor. Thus, if event E_1 occurs and event GuessCS occurs, then, whenever $\mathcal{A}$ wins against $\mathcal{M}$, $\mathcal{A}^*$ wins against $\mathcal{S}_{\mathsf{PAK\text{-}Z+}}$, except with probability at most $\Pr(\mathsf{Dist}_1|\mathsf{GuessCS})$. Therefore,

$$\Pr\left(\mathsf{Succ}^{\text{ake-f}\ell}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})|\mathsf{E}_1, \mathsf{GuessCS}\right) \leq \Pr\left(\mathsf{Succ}^{\mathrm{ake}}_{\mathsf{PAK\text{-}Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})\right) \ ,$$

where $q'_{\mathsf{ro}} \leq n(q_{\mathsf{ro}} + z + 6q_{\mathsf{ex}} + 4q_{\mathsf{se}})$, $t' \leq t + n(q_{\mathsf{ro}} + 1)t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + |I_a|t_{\mathsf{sig}}) + q_{\mathsf{se}}(2nt_{\mathsf{exp}} + |I_a|t_{\mathsf{sig}})$, and $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$. Moreover,

$$\left|\Pr\left(\mathsf{Succ}^{\text{ake-f}\ell}_{\mathsf{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})|\mathsf{E}_1, \mathsf{GuessCS}\right) - \Pr\left(\mathsf{Succ}^{\text{ake-f}\ell}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})|\mathsf{E}_1, \mathsf{GuessCS}\right)\right| \leq \Pr(\mathsf{Dist}_1|\mathsf{GuessCS}) \ .$$

Combining these two expressions yields the following result:

Lemma A.1 *Let $U^* \in \mathsf{Clients}$, $U'^* \in \mathsf{Servers}$, and suppose that neither $\mathsf{RevealFactor}_{\mathsf{MFPAK}}(U^*, U'^*, \ell^*)$ nor $\mathsf{RevealFactorV}_{\mathsf{MFPAK}}(U'^*, U^*, \ell^*)$ occurs (which is event E_1). Then*

$$\Pr\left(\mathsf{Succ}^{\text{ake-f}\ell}_{\mathsf{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})|\mathsf{E}_1, \mathsf{GuessCS}\right) \leq 4\Pr\left(\mathsf{Succ}^{\mathrm{ake}}_{\mathsf{PAK\text{-}Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})\right) \ ,$$

where $q'_{\mathsf{ro}} \leq n(q_{\mathsf{ro}} + z + 6q_{\mathsf{ex}} + 4q_{\mathsf{se}})$, $t' \leq t + n(q_{\mathsf{ro}} + 1)t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + |I_a|t_{\mathsf{sig}}) + q_{\mathsf{se}}(2nt_{\mathsf{exp}} + |I_a|t_{\mathsf{sig}})$, and $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$, and a similar bound exists for $\mathsf{Adv}^{\text{s2c-f}\ell}_{\mathsf{MFPAK}}$.

A.3 Remaining cases

The remaining cases are as follows:

2. Asymmetric factor uncompromised, $U^* \in \mathsf{Servers}$: no $\mathsf{RevealFactor}_{\mathsf{MFPAK}}(U'^*, U^*, \ell^*)$ query.
3. Symmetric factor uncompromised, $U^* \in \mathsf{Clients}$: no $\mathsf{RevealFactor}_{\mathsf{MFPAK}}(U^*, U'^*, \ell^*)$ query.
4. Symmetric factor uncompromised, $U^* \in \mathsf{Servers}$: no $\mathsf{RevealFactor}_{\mathsf{MFPAK}}(U'^*, U^*, \ell^*)$ query.

The proofs for each of these cases proceed in an analogous manner. For case 2, the modifier simulates an MFPAK system to the adversary using an underlying PAK-Z+ system and assuring that the underlying system remains fresh. For cases 3 and 4, the modifier simulates an MFPAK system to the adversary using an underlying PAK system and assuring that it remains fresh.

The details for these three cases appear in the full version of the paper (Stebila et al. 2009).

A.4 Overall result

By combining cases 1 and 2, we can obtain a result for instances that are fresh in the ℓ^*th factor when that factor is asymmetric, and by combining cases 3 and 4 we can obtain a result for instances that are fresh in the ℓ^*th factor when that factor is symmetric.

For the ake-fℓ advantage for an asymmetric factor, we have

$$\Pr\left(\mathsf{Succ}^{\text{ake-f}\ell}_{\mathsf{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})\right) \leq |\mathsf{Clients}| \cdot |\mathsf{Servers}| \cdot 8\Pr\left(\mathsf{Succ}^{\mathrm{ake}}_{\mathsf{PAK\text{-}Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})\right) \ ,$$

where $t' \leq t + n(q_{\mathsf{ro}} + 1)t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + |I_a|t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + |I_a|t_{\mathsf{sig}})$, $q'_{\mathsf{ro}} \leq n(q_{\mathsf{ro}} + z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}})$, and $z = \max\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$.

For the ake-fℓ advantage for a symmetric factor, we have

$$\Pr\left(\mathsf{Succ}^{\text{ake-f}\ell}_{\mathsf{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})\right) \leq |\mathsf{Clients}| \cdot |\mathsf{Servers}| \cdot 2\Pr\left(\mathsf{Succ}^{\mathrm{ake}}_{\mathsf{PAK}}(t'', q_{\mathsf{se}}, q_{\mathsf{ex}}, q''_{\mathsf{ro}})\right) \ ,$$

where $q''_{\mathsf{ro}} \leq n(2q_{\mathsf{ro}} + 1 + 4z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}})$, $t'' \leq t + z|I_a|t_{\mathsf{Gen}} + (q_{\mathsf{ro}}+1)t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + |I_a|t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + |I_a|t_{\mathsf{sig}})$, and $z = \max\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$.

In each case, a similar bound applies for $\mathsf{Adv}^{\text{ma-f}\ell}_{\mathsf{MFPAK}}$.

Substituting the security statements for PAK (MacKenzie 2002, Thm. 6.9) and PAK-Z+ (Gentry et al. 2005, Thm. 5.1) and simplifying the expressions, we obtain the following theorem describing the security of MFPAK:

Theorem A.2 *Let G be a finite cyclic group generated by g and let S be a signature scheme with security parameter κ. 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 oracle. Let $b_{\mathsf{co}} = 1$ if $\mathcal{A}$ makes a $\mathsf{RevealFactorV}(\cdot, \cdot, \ell)$ query to a server, and 0 otherwise. Then* MFPAK *is a secure multi-factor password-authenticated key exchange protocol, with*

$$\mathsf{Adv}^{\text{ake-f}\ell}_{\mathsf{MFPAK}}(\mathcal{A}) \leq \begin{cases} \frac{16\delta((1-b_{\mathsf{co}})q_{\mathsf{se}} + b_{\mathsf{co}}q_{\mathsf{ro}})}{|\mathsf{Passwords}^{\ell}|} + \epsilon, \\ \quad \textit{if the } \ell\textit{th factor is symmetric,} \\ \frac{4\delta q_{\mathsf{se}}}{|\mathsf{Passwords}^{\ell}|} + \epsilon \ , \\ \quad \textit{if the } \ell\textit{th factor is asymmetric,} \end{cases}$$

where $\epsilon = 8q_{\mathsf{se}}\mathsf{Adv}^{\mathrm{cdh}}_{G,g}(t', q'^2_{\mathsf{ro}}) + 6q_{\mathsf{se}}\mathsf{Succ}^{\text{eu-cma}}_{\mathcal{S},\kappa}(t', q_{\mathsf{se}}) + \frac{5(q_{\mathsf{se}}+q_{\mathsf{ex}})(q_{\mathsf{ro}}+q_{\mathsf{se}}+q_{\mathsf{ex}})}{|G|}$ and $\delta = |\mathsf{Clients}| \cdot |\mathsf{Servers}|$, for $t' = t + (z|I_a| + 8(q'^2_{\mathsf{ro}} + |I_a|q_{\mathsf{se}} + |I_a|q_{\mathsf{ex}}))t_{\mathsf{exp}}$, $q'_{\mathsf{ro}} = n(2q_{\mathsf{ro}} + 4z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}})$, and $z = \max\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$; a similar bound exists for $\mathsf{Adv}^{\text{ma-f}\ell}_{\mathsf{MFPAK}}(\mathcal{A})$.