Multi-Factor Password-Authenticated Key Exchange

Douglas Stebila[1,2], Poornaprajna Udupi[2], and Sheueling Chang[2]

[1] Dept. of Combinatorics & Optimization, University of Waterloo, Waterloo, ON, Canada
[2] Sun Microsystems Laboratories, Menlo Park, CA, USA
email: `douglas@stebila.ca, {poornaprajna.udupi,sheueling.chang}@sun.com`

2008 April 3

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 the first formal security model for multi-factor password-authenticated key exchange protocols, propose an efficient and secure protocol called MFPAK, and provide a formal 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 the formal analysis proceeds in the random oracle model.

Keywords: passwords, authentication, key exchange, multi-factor.

Contents

1 Introduction

Practical motivation. Two major security problems on the Internet today are phishing and spyware. *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; the phisher can then use the information obtained to impersonate the user. Additionally, many users' computers are compromised with spyware, which can record users' keystrokes (and thus passwords) and again provide this information to a malicious party.

Such attacks are possible not because of the break of any cryptographic protocol but because of externalities such as social engineering and software bugs.

Several techniques to reduce the risks of these attacks are being used in practice. Physical devices that generate one-time passwords are being used to secure corporate virtual private networks (VPNs) and some online banking sessions. Server-side *multi-layer* techniques that take into account additional attributes, such as HTTP cookies, IP address, and browser user agent string, are being deployed as well. These techniques can offer greater assurance as to the identity of the user but, even when deployed over today's web security protocol HTTPS/TLS, remain susceptible to sophisticated impersonation attacks because they do not protect authentication secrets or provide strong, intuitive server-to-client authentication.

Password-authenticated key exchange is a strong technique to defend against impersonation attacks and provide server-to-client authentication, but current protocols depend solely on a long-term password, which can be risky when used on a spyware-infested computer.

Multi-factor authentication adds a further degree of assurance to the authentication procedure. Long-term passwords are easily memorized, infrequently changed, and used repeatedly. One-time responses are used once: they change frequently and, though not easily memorized, can be provided by a small electronic token or a sheet of paper. These factors offer different but complementary resistance to different types of compromise. Together, they offer more assurance in authentication because stealing the long-term password alone (for example, by installing spyware) or losing the one-time password card alone is insufficient to compromise the authentication procedure.

We believe that it is important to design a multi-factor protocol that can leverage multiple client authentication attributes and, equally important, to convey them securely in a multi-factor cryptographic protocol. Our approach builds upon previous work on password-authenticated key exchange by combining multiple authentication factors of complementary natures in a multi-factor authenticated key exchange protocol.

Our contributions. We provide the first formal treatment of multi-factor password-authenticated key exchange. We define a formal model which is an extension of the Bellare-Pointcheval-Rogaway model [BPR00] for password-authenticated key exchange. We formalize the security notion that a multi-factor protocol should remain secure even if all but one of the factors has been compromised by adapting the definition of freshness of a session.

We present an efficient two-factor protocol that is secure in this model under standard cryptographic assumptions in the random oracle model.

Our work differs from previous work in password-authenticated key exchange because it utilizes two independent, complementary factors for authentication. Other work has considered some aspects of multi-factor authentication, but these have either used at least one factor that is a long cryptographic secret (as opposed to our work which allows both factors to be short, human-friendly strings), or have not provided strong server-to-client authentication resistant to man-in-the-middle attacks.

Related work. Password-authenticated key exchange protocols have been extensively studied in the literature. A large list of papers concerning password-based key agreement is maintained by Jablon [Jab07].

Password-authenticated key exchange was first introduced by Bellovin and Merritt in 1992 [BM92] in the form of encrypted key exchange (EKE), a 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) [BM94], removed the requirement that the server have the plaintext password, instead having a one-way function of the password. The former is called a *symmetric* password-based protocol, because both client and server share the same plaintext password, whereas the latter is called *asymmetric*.

A number of formal models for the security of password-authenticated key exchange protocols have been introduced, including one by Shoup [Sho99b], one by Bellare, Pointcheval, and Rogaway [BPR00], and the three-party setting of Abdalla, Fouque, and Pointcheval [AFP05].

Many password-authenticated key exchange protocols have been developed; recent work has focused on protocols with formal security claims and proofs in both the random oracle model and the standard model, with some protocols (e.g., [ABC$^+$06, TWMP06]) focusing on suitability for implementation in existing network protocols such as SSL/TLS.

The design of our protocol combines aspects of PAK [BMP00a, Mac02] and PAK-Z+ [GMR05]. PAK is a symmetric protocol whereas PAK-Z+ is an asymmetric protocol: both have a similar structure but use authentication secrets of different natures. Both have been shown to be secure in the Bellare-Pointcheval-Rogaway model. The technique used to show the security of PAK-Z+ is a specialization of the same authors' later Ω-method [GMR06] for converting a symmetric password-authenticated key exchange protocol into an asymmetric one.

A two-factor authentication scheme for smart cards was proposed by Yang *et al.* [YWWD06a]. Their scheme relies on a smart card storing and returning a cryptographically large (e.g., 160-bit) private value, relies on a public key infrastructure, and requires that the user input a password into the smart card for each login. Other protocols that require the client to store a long cryptographic secret and the server's public key include schemes by Park and Park [PP04] and Yoon and Yoo [YY06].

There are other two-factor authentication schemes used in practice which do not provide cryptographic protection for the two factors. In a *multi-channel* system, the second factor is delivered over over a separate second channel (e.g., 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 user agent 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 [SDOF07].

Outline. The rest of our paper proceeds as follows. In Section 2, we describe the security model for multi-factor password-authenticated key exchange. In Section 3, we present our protocol MFPAK, and show in Section 4 (and Appendix B) through a formal analysis that it is secure in our model. Section 5 concludes the paper with what we believe are interesting directions for future research. Appendix A specifies the PAK and PAK-Z+ protocols, design elements of which we use in MFPAK, and states the security results for both of these protocols.

Appendix C provides a sample instantiation of the MFPAK protocol for a common security level.

2 Security for multi-factor protocols

In a multi-factor password-authenticated key exchange protocol, multiple authentication secrets of complementary natures, such as a long-term password and a one-time response value, are combined securely to provide mutual authentication and to establish a shared secret key for a private channel.

The authentication secrets must be combined so 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.

In addition to providing authentication, the protocol should also establish an ephemeral shared secret key between client and server that can then be used, for example, to establish a private channel using bulk encryption.

Informal security criteria. The general security criteria we use for multi-factor password-authenticated key exchange is that the protocol should remain secure even if all but one authentication factor is fully known to an adversary. Throughout this paper, we present the first example of such a protocol using two authentication factors. We identify four security properties that such a protocol should have:

1. Strong two-factor server-to-client authentication: without knowledge of both of the authentication factors, a server cannot successfully convince a client of its identity.
2. Strong two-factor client-to-server authentication: without knowledge of both 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.

Figure 1 compares our scheme with existing password-authenticated key exchange protocols, with a two-factor scheme that transmits the password and response value across an SSL channel, and with a multi-layer scheme that uses non-cryptographic attributes, such as browser version and IP address, for additional assurance.

This table shows that other two-factor schemes that the financial industry is deploying today, such as transmitting one-time values over TLS, will not address the phishing problem. Solving this problem requires a fundamental change in the underlying cryptographic protocol. Our scheme provides a secure solution to this problem, in the form of multi-factor password-authenticated key exchange.

2.1 Formal security model

We define a formal model for the security of multi-factor password-authenticated key exchange that allows us to show that our 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 as well.

This formal model is an extension of the model for password-authenticated key exchange proposed by Bellare, Pointcheval, and Rogaway [BPR00] and modified by Gentry, MacKenzie, and Ramzan [GMR05]. We state our model for two factors, but it could easily be extended for an arbitrary number of factors.

Security property	SSL + multi-channel schemes	SSL + password + one-time response or SSL + multi-layer schemes	Existing password auth. key exchange protocols	Our protocol
1. Strong two-factor server auth.	Susceptible to man-in-the-middle attacks. Server authenticated only by X.509 certificate.		Only one factor.	Yes.
2. Strong two-factor client auth.	Susceptible to man-in-the-middle attacks. Needs second channel.		Only one factor.	Yes.
3. Auth. secrets protected	Authentication occurs after session key established. User authentication secrets sent directly to server.		Yes.	Yes.
4. Secure session key establishment	Yes.		Yes.	Yes.

Figure 1: Comparison of security properties of various schemes.

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 $\mathsf{Parties} = \mathsf{Clients} \dot\cup \mathsf{Servers}$.

Authentication secrets are short strings selected uniformly at random from an appropriate set. The long term passwords (the *first factor*) are chosen from the set $\mathsf{Passwords}$ while the short term responses (the *second factor*) are chosen from the set $\mathsf{Responses}$.

For each client-server pair $(C, S) \in \mathsf{Clients} \times \mathsf{Servers}$, authentication secrets exist. There is a long-term password $\mathsf{pw}_{C,S} \in \mathsf{Passwords}$, and a corresponding password verifier $\mathsf{pw}_S[C]$ which is some transformation of $\mathsf{pw}_{C,S}$; $\mathsf{pw}_S[C]$ is stored on the server S, which is the asymmetric model. There is a short-term response $\mathsf{re}_{C,S} \in \mathsf{Responses}$, which is known to both the client and the server, which is the symmetric model.

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_i^U.

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 fixed number of flows, both instances may *terminate* and *accept*, at which point they hold a *session key* sk, *partner id* pid, and *session id* sid. Two instances Π_i^C and Π_j^S are said to be *partnered* if they both accept, hold $(\mathsf{pid}, \mathsf{sid}, \mathsf{sk})$ and $(\mathsf{pid}', \mathsf{sid}', \mathsf{sk}')$ 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.

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 parties' oracle instances and receives responses. 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_i^C and server instance Π_j^S to faithfully execute protocol P and returns the resulting transcript.
- $\mathsf{Send}_P(U, i, M)$: Sends message M to user instance Π_i^U, 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_i^U has accepted, then causes the following to happen: choose $b \in_R \{0,1\}$; if $b=1$, then return the session key of Π_i^U, otherwise return a random string of the same length as the session key. This query may only be asked once.
- $\mathsf{Reveal}_P(U,i)$: If user instance Π_i^U has accepted, then returns session key sk held by Π_i^U.
- $\mathsf{CorruptPWC}_P(C,S)$: Returns the password $\mathsf{pw}_{C,S}$ of client C with server S.
- $\mathsf{CorruptPWS}_P(S,C)$: Returns the password verifier $\mathsf{pw}_S[C]$ of client C on server S.
- $\mathsf{CorruptRe}_P(C,S)$: Returns the response $\mathsf{re}_{C,S}$ of client C with server S.

The various $\mathsf{Corrupt}*$ 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.

Freshness. We have adapted the notion of freshness to accommodate the idea that a session should remain fresh even if one of the authentication factors has been fully compromised. An instance Π_i^U with partner id U' is *fresh in the first factor* (with forward-secrecy) if and only if none of the following events occur:

1. a $\mathsf{Reveal}(U,i)$ query occurs;
2. a $\mathsf{Reveal}(U',j)$ query occurs, where $\Pi_j^{U'}$ is the partner instance of Π_i^U;
3. $U \in \mathsf{Clients}$, either $\mathsf{CorruptPWC}(U,U')$ or $\mathsf{CorruptPWS}(U',U)$ occurs before the Test query, and $\mathsf{Send}(U,i,M)$ occurs for some string M;
4. $U \in \mathsf{Servers}$, $\mathsf{CorruptPWC}(U',U)$ occurs before the Test query, and $\mathsf{Send}(U,i,M)$ occurs for some string M.

An instance Π_i^U with partner id U' is *fresh in the second factor* (with forward-secrecy) if and only if none of the following events occur:

1. a $\mathsf{Reveal}(U,i)$ query occurs;
2. a $\mathsf{Reveal}(U',j)$ query occurs, where $\Pi_j^{U'}$ is the partner instance of Π_i^U;
3. $U \in \mathsf{Clients}$, any $\mathsf{CorruptRe}$ query occurs before the Test query, and $\mathsf{Send}(U,i,M)$ occurs for some string M;
4. $U \in \mathsf{Servers}$, any $\mathsf{CorruptRe}$ occurs before the Test query, and $\mathsf{Send}(U,i,M)$ occurs for some string M.

If a session is fresh in both the first and second factors, then it is also fresh in the original notion of freshness for password-authenticated key exchange.

Adversary's goals. The goal of an adversary is to guess the bit b used in the Test query of a fresh in the first (or second) factor session; this corresponds to the ability of an adversary to distinguish the session key from a random string of the same length. Let $\mathsf{Succ}_P^{\text{ake-f1}}(\mathcal{A})$ (respectively, $\mathsf{Succ}_P^{\text{ake-f2}}(\mathcal{A})$) be the event that the adversary $\mathcal{A}$ makes a single Test query to some fresh in the first (respectively, second) factor instance Π_i^U that has terminated and $\mathcal{A}$ eventually outputs a bit b', where $b'=b$ and b is the randomly selected bit in the Test query. The *ake-f1 advantage* of $\mathcal{A}$ attacking P is defined to be $\mathsf{Adv}_P^{\text{ake-f1}}(\mathcal{A}) = 2\Pr\left(\mathsf{Succ}_P^{\text{ake-f1}}(\mathcal{A})\right)-1$, and analogously for ake-f2.

We can define similar notions for *client-to-server*, *server-to-client*, and *mutual* authentication. We define $\mathsf{Adv}_P^{\text{c2s-f1}}(\mathcal{A})$ (resp., $\mathsf{Adv}_P^{\text{c2s-f2}}(\mathcal{A})$) to be the probability that a server instance Π_j^S with partner id C terminates without having a partner oracle before a $\mathsf{CorruptPWC}_P(C,S)$ (resp., $\mathsf{CorruptRe}_P(C,S)$) query. We define $\mathsf{Adv}_P^{\text{s2c-f1}}(\mathcal{A})$ (resp., $\mathsf{Adv}_P^{\text{s2c-f2}}(\mathcal{A})$) to be the probability that a client instance Π_i^C with partner id S terminates without having a partner oracle before either a $\mathsf{CorruptPWC}_P(C,S)$ or $\mathsf{CorruptPWS}_P(S,C)$ (resp., $\mathsf{CorruptRe}_P(C,S)$) query. Finally, we define $\mathsf{Adv}_P^{\text{ma-f}i}(\mathcal{A})$ to be the probability that any instance terminates without having a partner oracle, for each notion i above.

We overload the Adv (and corresponding $\Pr(\mathsf{Succ})$) notation as follows: $\mathsf{Adv}_P^N(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) = \max_{\mathcal{A}}\{\mathsf{Adv}_P^N(\mathcal{A})\}$, 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 making at most q_{ro} random oracle queries.

Security. We say that a protocol P is a *secure multi-factor password authenticated key agreement protocol* if, for all polynomially-bounded adversaries $\mathcal{A}$, there exists a constant δ and a negligible ϵ such that

$$\mathsf{Adv}_P^{\text{ake-f1}}(\mathcal{A}) \leq \frac{\delta q_{\mathsf{se}}}{|\mathsf{Passwords}|} + \epsilon \quad \text{and} \quad \mathsf{Adv}_P^{\text{ake-f2}}(\mathcal{A}) \leq \frac{\delta q_{\mathsf{se}}}{|\mathsf{Responses}|} + \epsilon \ ,$$

and corresponding bounds apply for $\mathsf{Adv}_P^{\text{ma-f1}}(\mathcal{A})$ and $\mathsf{Adv}_P^{\text{ma-f2}}(\mathcal{A})$. 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.

Since a session that is fresh in both the first and second factors is also fresh in the original ake notion of password-authenticated key exchange, we have that

$$\mathsf{Adv}_P^{\text{ake}}(\mathcal{A}) \leq \min\left\{\mathsf{Adv}_P^{\text{ake-f1}}(\mathcal{A}), \mathsf{Adv}_P^{\text{ake-f2}}(\mathcal{A})\right\} \ .$$

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

MFPAK is the first password-authenticated key exchange protocol to be based on multiple authentication factors. The first factor is meant to represent a long-term user memorized password. This is an asymmetric factor in the terminology of [BPR00], meaning that the value stored on the server is the output of a (supposed) one-way function of the password, so that the compromise of the server's database does not allow the client to be impersonated. The second factor is meant to represent a dynamic one-time response value.[1] This is a symmetric factor, meaning that the client and the server store (effectively) the same value.

We designed MFPAK by considering two existing one-factor protocols as our building blocks: the asymmetric password protocol PAK-Z+ for the long-term password, and the symmetric password protocol PAK for the one-time response values. These two protocols have design characteristics that we need for the design of our two-factor protocol and have formal security arguments. Both factors are tightly integrated into the authentication and key exchange processes.

3.1 Ingredients

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.

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

[1]If the second factor changes over time, or is the response to challenge issued by the server, then our model accommodates those shorter-lived values as a specialization of the general case. Synchronizing changing response values has been studied: for example, the second factor response value could be the output of a small calculator-like device which outputs pseudorandom values (e.g., the RSA SecurID device [RSA]) or the response to a particular one-time challenge from a sheet of paper listing one-time responses.

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 [BR93]. 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 [GMR88]. Let $(V, W) \leftarrow \mathsf{Gen}(1^\kappa)$. 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 W 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 $\mathsf{Verify}_W(m, \sigma) = \mathsf{true}$, 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.

3.2 Protocol specification

The MFPAK protocol, like many other protocols, contains two stages: a user registration stage, completed once per user, and a login stage, completed each time a user attempts to login.

The user registration stage of MFPAK is given in Figure 2 below. This stage should be completed over a private, authentic channel. As above, we assume the response value re is fixed for each client-server pair, but the scenario could allow for a challenge/response or pseudorandomly-generated response value.

The login stage of MFPAK is given in Figure 3. This stage can be completed over a public, untrusted channel. A client C initiates the login stage with a server S. The client knows the password $\mathsf{pw}_{C,S}$ and response $\mathsf{re}_{C,S}$ that was previously established in the registration stage, and the server S has its databases $\mathsf{pw}_S[C]$ and $\mathsf{re}_S[C]$, for all $C \in \mathsf{Clients}$, of corresponding values. If the response values are meant to be responses to challenges issued by the server, an initial message from the server to the client conveying the challenge can be added to the login stage of the protocol.

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 3 specifies $\textsc{ClientAction}0_{\mathrm{MFPAK}}$, $\textsc{ServerAction}1_{\mathrm{MFPAK}}$, $\textsc{ClientAction}2_{\mathrm{MFPAK}}$, and $\textsc{ServerAction}3_{\mathrm{MFPAK}}$

MFPAK User Registration			
	Client C		Server S
1.	$\mathsf{pw}_{C,S} \in_R \mathsf{Passwords}$		
2.	$\mathsf{re}_{C,S} \in_R \mathsf{Responses}$		
3.	$(V, W) \leftarrow \mathsf{Gen}(1^\kappa)$		
4.	$\gamma' = (H_1(C, S, \mathsf{pw}_{C,S}))^{-1}$		
5.	$V' = H_2(C, S, \mathsf{pw}_{C,S}) \oplus V$		
6.	$V'' = H_3(V)$		
7.	$\tau' = (H_4(C, S, \mathsf{re}_{C,S}))^{-1}$		
8.		$\xrightarrow{C,\gamma',W,V',V'',\tau'}$	
9.			Store $\mathsf{pw}_S[C] = \langle \gamma', W, V', V'' \rangle$
10.			Store $\mathsf{re}_S[C] = \tau'$

Figure 2: The user registration stage of the MFPAK protocol.

Efficiency. We consider group exponentiations, group inversions, and signature generation and verification operations to be expensive, but group multiplication, addition, and hash function computation to be inexpensive. Compared to the one-factor protocol PAK-Z+, our protocol MFPAK achieves two-factor security with almost the same efficiency. In particular, MFPAK uses the same number expensive operations on the server side as PAK-Z+ (2 exponentiations, 1 signature verification), and only one more expensive operation on client side than PAK-Z+ (2 exponentiations, 1 signature generation, and 2 inversions (compared to the same but with only 1 inversion)). In many situations, such as e-commerce and online banking, the limiting factor is the number of connections a server can handle, and so MFPAK can increase security without substantial additional computational burden on the server.

4 Formal security analysis

Our general technique is to show that, if one factor remains uncompromised, then the difficulty of breaking MFPAK is related to the difficulty of breaking the corresponding one of either PAK or PAK-Z+.

More precisely, for each of the two factors (password and response), we describe a procedure specified by a modifier $\mathcal{M}$ to transform an adversary $\mathcal{A}$ against MFPAK with the specified factor uncompromised into an adversary $\mathcal{A}^*$ against the corresponding one of the two underlying protocols (PAK-Z+ and PAK, respectively). The transformations are such that, if the oracle instance in MFPAK against which the Test query is directed is fresh in the first (resp, second) factor, then the corresponding oracle instance is also fresh in the corresponding attack on PAK-Z+ (resp., PAK).

Our formal argument proceeds by considering four cases. There are two cases corresponding to the password being uncompromised and two cases corresponding to the response being uncompromised, and for each of those one case is when $U \in \mathsf{Clients}$ and the other case is when $U \in \mathsf{Servers}$, where U is the user instance towards which the Test query is directed. We can then combine the four cases probabilistically and obtain a security argument for the general setting.

In the main body of the paper, we provide the detailed description of the procedure for one case and state the overall result. The other three cases follow in an analogous way and are provided in Appendix B.

The four cases, and the sections in which the details appear, are as follows:

1. $U^* \in \mathsf{Clients}$, no $\mathsf{CorruptPWC}_{\mathrm{MFPAK}}(U^*, U'^*)$ or $\mathsf{CorruptPWS}_{\mathrm{MFPAK}}(U'^*, U^*)$ query (Section 4.1),
2. $U^* \in \mathsf{Servers}$, no $\mathsf{CorruptPWC}_{\mathrm{MFPAK}}(U'^*, U^*)$ query (Appendix B.1),
3. $U^* \in \mathsf{Clients}$, no $\mathsf{CorruptRe}_{\mathrm{MFPAK}}$ query (Appendix B.2), and

MFPAK Login			
	Client C		Server S
1.	$x \in_R \mathbb{Z}_q$		
2.	$X = g^x$		
3.	$\gamma = H_1(C, S, \mathsf{pw}_{C,S})$		
4.	$\tau = H_4(C, S, \mathsf{re}_{C,S})$		
5.	$m = X \cdot \gamma \cdot \tau$		
6.		$\xrightarrow{C,m}$	
7.			Abort if $\neg\mathsf{Acceptable}(m)$
8.			$y \in_R \mathbb{Z}_q$
9.			$Y = g^y$
10.			$\langle \gamma', W, V', V'' \rangle = \mathsf{pw}_S[C]$
11.			$\tau' = \mathsf{re}_S[C]$
12.			$X = m \cdot \gamma' \cdot \tau'$
13.			$\sigma = X^y$
14.			$\mathsf{sid} = \langle C, S, m, Y \rangle$
15.			$k = H_5(\mathsf{sid}, \sigma, \gamma', \tau')$
16.			$a' = H_6(\mathsf{sid}, \sigma, \gamma', \tau')$
17.			$a = a' \oplus V'$
18.		$\xleftarrow{Y,k,a,V''}$	
19.	$\sigma = Y^x$		
20.	$\gamma' = \gamma^{-1}$		
21.	$\tau' = \tau^{-1}$		
22.	$\mathsf{sid} = \langle C, S, m, Y \rangle$		
23.	Abort if $k \neq H_5(\mathsf{sid}, \sigma, \gamma', \tau')$		
24.	$k' = H_7(\mathsf{sid}, \sigma, \gamma', \tau')$		
25.	$a' = H_6(\mathsf{sid}, \sigma, \gamma', \tau')$		
26.	$V' = a' \oplus a$		
27.	$V = H_2(C, S, \mathsf{pw}_{C,S}) \oplus V'$		
28.	Abort if $V'' \neq H_3(V)$		
29.	$s = \mathsf{Sign}_V(\mathsf{sid})$		
30.		$\xrightarrow{k',s}$	
31.			Abort if $k' \neq H_7(\mathsf{sid}, \sigma, \gamma', \tau')$
32.			Abort if $\neg\mathsf{Verify}_W(\mathsf{sid}, s)$
33.	$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \gamma', \tau')$		$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \gamma', \tau')$

Figure 3: The login stage of the MFPAK protocol.

4. $U^* \in \mathsf{Servers}$, no $\mathsf{CorruptRe}_{\mathrm{MFPAK}}$ query (Appendix B.3).

These four cases are combined into the overall result in Section 4.2.

4.1 Case 1: Attacking a client instance, first factor uncompromised

This case addresses impersonation of the server when the session being attacked is a client instance and the first factor remains uncompromised.

The modifier $\mathcal{M}$ first uniformly at randomly 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 $\mathsf{GuessCS}$ be the event that the modifier $\mathcal{M}$ correctly guesses U^* and U'^*. Then

$$\Pr(\mathsf{GuessCS}) = \Pr((U^* \text{ correct}) \wedge (U'^* \text{ correct})) \geq \frac{1}{|\mathsf{Clients}| \cdot |\mathsf{Servers}|} \ . \tag{1}$$

For this case, we assume that no $\mathsf{CorruptPWC}_{\mathrm{MFPAK}}(U^*, U'^*)$ or $\mathsf{CorruptPWS}_{\mathrm{MFPAK}}(U'^*, U^*)$ query is issued against $\mathcal{M}$: this case models server impersonation in the first factor, which is why no $\mathsf{CorruptPWS}_{\mathrm{MFPAK}}(U'^*, U^*)$ query is allowed. Furthermore, no $\mathsf{CorruptPWC}_{\mathrm{MFPAK}}(U^*, U'^*)$ is allowed because an adversary can easily recover the verifier $\mathsf{pw}_{U'^*}[U^*]$ from the password pw_{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 and response preparation. For each $(C, S) \in \mathsf{Clients} \times \mathsf{Servers}$, $\mathcal{M}$ sets $\mathsf{re}_{C,S} \in_R \mathsf{Responses}$ and constructs the corresponding $\mathsf{re}_S[C]$. In particular, $\mathcal{M}$ sets $\tau^* = H_4(U^*, U'^*, \mathsf{re}_{U^*,U'^*})$ and $\tau'^* = (\tau^*)^{-1}$. For each $(C, S) \in (\mathsf{Clients} \times \mathsf{Servers}) \setminus \{(U^*, U'^*)\}$, $\mathcal{M}$ sets $\mathsf{pw}_{C,S} = \mathsf{CorruptPWC}_{\text{PAK-Z+}}(C, S)$ and $\mathsf{pw}_S[C] = \mathsf{CorruptPWS}_{\text{PAK-Z+}}(S, C)$. Of all the password and response values, only pw_{U^*,U'^*} and $\mathsf{pw}_{U'^*}[U^*]$ remain unknown to $\mathcal{M}$ at this point.

Instantiation of PAK-Z+ simulator. We instantiate the PAK-Z+ simulator $\mathcal{S}_{\text{PAK-Z+}}$ with the following random oracles: $H^*_\ell = H_\ell$, for $\ell = 1, 2, 3$, and $H^*_\ell(\langle C, S, m, Y\rangle, \sigma, \gamma') = H_\ell(\langle C, S, m \cdot \tau^*, Y\rangle, \sigma, \gamma', \tau'^*)$, for $\ell = 5, 6, 8$.[2] These 'starred' functions are independent random oracles if the corresponding unstarred functions are. The above construction is possible since τ^* and τ'^* are fixed and known to $\mathcal{M}$ because of the guesses made at the beginning of this case.

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

$$\mathsf{Sign}^*_V(\langle C, S, m, Y\rangle) = \mathsf{Sign}_V(\langle C, S, m \cdot \tau'^*, Y\rangle)$$
$$\mathsf{Verify}^*_W(\langle C, S, m, Y\rangle, s) = \mathsf{Verify}_W(\langle C, S, m \cdot \tau'^*, Y\rangle, s) \ .$$

Since the transformation that sends $\langle C, S, m, Y\rangle \mapsto \langle C, S, m \cdot \tau'^*, Y\rangle$ 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.

$\mathcal{M}$'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 $\mathcal{S}_{\text{PAK-Z+}}$.

$\underline{\mathsf{CorruptPWC}(C, S)}$:

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

$\underline{\mathsf{CorruptPWS}(S, C)}$:

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

$\underline{\mathsf{CorruptRe}(C, S)}$: Return $\mathsf{re}_{C,S}$.

$\underline{\mathsf{Test}(U, i)}$:

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

$\underline{\mathsf{Reveal}(U, i)}$:

[2] Note that we do not need to instantiate H^*_4 and H^*_7 because these oracles are not used by PAK-Z+.

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

<u>$\mathsf{Execute}(C, i, S, j)$:</u>

1. If $(C, S) \neq (U^*, U'^*)$:
   $\mathcal{M}$ performs $\mathsf{Execute}_{\text{MFPAK}}(C, i, S, j)$ with all the values it has and returns the transcript.
2. If $(C, S) = (U^*, U'^*)$:
   $\mathcal{M}$ will use the PAK-Z+ simulator $\mathcal{S}_{\text{PAK-Z+}}$ to obtain a transcript for this query.
   (a) Send an $\mathsf{Execute}_{\text{PAK-Z+}}(C, i, S, j)$ query to $\mathcal{S}_{\text{PAK-Z+}}$ and receive $\langle C, m, Y, k, a, V'', s\rangle$.
   (b) Set $\hat{m} = m \cdot \tau^*$.
   (c) Set $\hat{k}' \in_R \text{range}(H_7)$.
   (d) Return $\langle C, \hat{m}, Y, k, a, V'', s, \hat{k}'\rangle$ to $\mathcal{A}$.

<u>$\mathsf{Send}(U, i, M)$:</u>

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

1. If $M = \langle\text{“start”}, S\rangle$ and $(U, S) \neq (U^*, U'^*)$:
   Perform $\textsc{ClientAction0}_{\text{MFPAK}}$ and return $\langle U, m\rangle$.
2. If $M = \langle\text{“start”}, S\rangle$ and $(U, S) = (U^*, U'^*)$:
   (a) Send a $\mathsf{Send}_{\text{PAK-Z+}}(U, i, M)$ query to $\mathcal{S}_{\text{PAK-Z+}}$ and receive $\langle U, m\rangle$.
   (b) Set $\hat{m} = m \cdot \tau^*$.
   (c) Return $\langle U, \hat{m}\rangle$.
3. If $M = \langle C, m\rangle$ and $(C, U) \neq (U^*, U'^*)$:
   Perform $\textsc{ServerAction1}_{\text{MFPAK}}$ and return $\langle Y, k, a, V''\rangle$.
4. If $M = \langle C, m\rangle$ and $(C, U) = (U^*, U'^*)$:
   (a) Set $\hat{m} = m \cdot \tau'^*$.
   (b) Send a $\mathsf{Send}_{\text{PAK-Z+}}(U, i, \langle C, \hat{m}\rangle)$ query to $\mathcal{S}_{\text{PAK-Z+}}$ and receive $\langle Y, k, a, V''\rangle$.
   (c) Return $\langle Y, k, a, V''\rangle$.
5. If $M = \langle Y, k, a, V''\rangle$ and $(U, U') \neq (U^*, U'^*)$, where U' is the partner of U:
   Perform $\textsc{ClientAction2}_{\text{MFPAK}}$ and return $\langle k', s\rangle$.
6. If $M = \langle Y, k, a, V''\rangle$ and $(U, U') = (U^*, U'^*)$, where U' is the partner of U:
   (a) Send a $\mathsf{Send}_{\text{PAK-Z+}}(U, i, \langle Y, k, a, V''\rangle)$ query to $\mathcal{S}_{\text{PAK-Z+}}$ and receive $\langle s\rangle$.
   (b) Set $\hat{k}' \in_R \text{range}(H_7)$ and store.
   (c) Return $\langle \hat{k}', s\rangle$.
7. If $M = \langle k', s\rangle$ and $(U', U) \neq (U^*, U'^*)$, where U' is the partner of U:
   Perform $\textsc{ServerAction3}_{\text{MFPAK}}$.
8. If $M = \langle k', s\rangle$ and $(U', U) = (U^*, U'^*)$, where U' is the partner of U:
   (a) Abort if k' is not the same as the $\hat{k}'$ generated in Case 6 above.
   (b) Send a $\mathsf{Send}_{\text{PAK-Z+}}(U, i, \langle s\rangle)$ query to $\mathcal{S}_{\text{PAK-Z+}}$.

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 and responses 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 $\mathcal{S}_{\text{PAK-Z+}}$ can by inspection be seen to match what the MFPAK simulator would do because $\mathcal{S}_{\text{PAK-Z+}}$ is using the specially-constructed random oracles H_ℓ^*. The differences between $\mathcal{M}$ and what a true MFPAK simulator would do are as follows:

- $\mathsf{CorruptPWC}(C, S)$ when $(C, S) = (U^*, U'^*)$, $\mathsf{CorruptPWS}(S, C)$ when $(C, S) = (U^*, U'^*)$, and $\mathsf{Test}(U, i)$ when $U \neq U^*$:
  The modifier $\mathcal{M}$ aborts 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 session in which a Test query is directed to $\Pi_i^{U^*}$ would not be fresh.

- $\mathsf{Execute}(C, i, S, j)$ when $(C, S) = (U^*, U'^*)$, $\mathsf{Send}(U, i, M)$ when $M = \langle Y, k, a, V'' \rangle$ and $(U, U') = (U^*, U'^*)$, where U' is the partner of U, and $\mathsf{Send}(U, i, M)$ 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 session instead of generating $k' = H_7(\mathsf{sid}, \sigma, \gamma', \tau')$. Since H_7 is a random oracle, this substitution is distinguishable by the adversary $\mathcal{A}$ if and only if $\mathcal{A}$ queries H_7 on the arguments $\mathsf{sid}, \sigma, \gamma', \tau'$. But if that occurs, then $\mathcal{A}$ must know γ'. These are the same inputs to the H_8^* oracle used to compute the session key in the PAK-Z+ simulation $\mathcal{S}_{\text{PAK-Z+}}$, so the same adversary could distinguish the output of $\mathsf{Test}_{\text{PAK-Z+}}(U^*, i)$ received from $\mathcal{S}_{\text{PAK-Z+}}$. The latter event corresponds to the event $\mathsf{Succ}^{\text{ake}}_{\text{PAK-Z+}}$, and so the substitution is distinguishable with probability at most $\Pr(\mathsf{Succ}^{\text{ake}}_{\text{PAK-Z+}})$.

Let $\mathsf{Dist}_1 \wedge \mathsf{GuessCS}$ 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 $\Pr(\mathsf{Dist}_1 \wedge \mathsf{GuessCS}) \leq 3\Pr(\mathsf{Succ}^{\text{ake}}_{\text{PAK-Z+}})$ 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 $\mathsf{CorruptPWC}_{\text{MFPAK}}(U^*, U'^*)$ nor $\mathsf{CorruptPWS}_{\text{MFPAK}}(U'^*, U^*)$ occurs. The session involving U^*, U'^* in $\mathcal{S}_{\text{PAK-Z+}}$ is fresh if and only if the corresponding session in $\mathcal{M}$ is fresh in the first factor Thus, if event E_1 occurs and event $\mathsf{GuessCS}$ occurs, then, whenever $\mathcal{A}$ wins against $\mathcal{M}$, $\mathcal{A}^*$ wins against $\mathcal{S}_{\text{PAK-Z+}}$, except with probability at most $\Pr(\mathsf{Dist}_1 \wedge \mathsf{GuessCS})$. Therefore,

$$\Pr(\mathsf{Succ}^{\text{ake-f1}}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_1 \wedge \mathsf{GuessCS}) \leq \Pr(\mathsf{Succ}^{\text{ake}}_{\text{PAK-Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ ,$$

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

$$\left| \Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_1 \wedge \mathsf{GuessCS}) - \Pr(\mathsf{Succ}^{\text{ake-f1}}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_1 \wedge \mathsf{GuessCS}) \right|$$
$$\leq \Pr(\mathsf{Dist}_1 \wedge \mathsf{GuessCS}) \ .$$

Combining these two expressions yields the following result:

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

$$\Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_1 \wedge \mathsf{GuessCS}) \leq 4\Pr(\mathsf{Succ}^{\text{ake}}_{\text{PAK-Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ ,$$

where $q'_{\mathsf{ro}} \leq q_{\mathsf{ro}} + z + 1 + 6q_{\mathsf{ex}} + 4q_{\mathsf{se}}$, $t' \leq t + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(2t_{\mathsf{exp}} + t_{\mathsf{sig}})$, *and* $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$, *and a similar bound exists for* $\mathsf{Adv}^{\text{s2c-f1}}_{\text{MFPAK}}$.

4.2 Overall result

By combining the cases 1 and 2, we can obtain a result for sessions that are fresh in the first factor, and by combining cases 3 and 4 we can obtain a result for sessions that are fresh in the second factor. For the ake-f1 advantage, we have

$$\begin{aligned}
\Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})) &\leq \Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_1 \wedge \mathsf{GuessCS}) / \Pr(\mathsf{GuessCS}) \\
&\quad + \Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_2 \wedge \mathsf{GuessSC}) / \Pr(\mathsf{GuessSC}) \\
&\leq |\mathsf{Clients}| \cdot |\mathsf{Servers}| \cdot 8\Pr(\mathsf{Succ}_{\text{PAK-Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ ,
\end{aligned}$$

where $t' \leq t + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}})$, $q'_{\mathsf{ro}} \leq q_{\mathsf{ro}} + 1 + z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, and $z = \max\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$. For the ake-f2 advantage, we have

$$\begin{aligned}
\Pr(\mathsf{Succ}^{\text{ake-f2}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})) &\leq \Pr(\mathsf{Succ}^{\text{ake-f2}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3 \wedge \mathsf{GuessCS}) / \Pr(\mathsf{GuessCS}) \\
&\quad + \Pr(\mathsf{Succ}^{\text{ake-f2}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3 \wedge \mathsf{GuessSC}) / \Pr(\mathsf{GuessSC}) \\
&\leq |\mathsf{Clients}| \cdot |\mathsf{Servers}| \cdot 2\Pr(\mathsf{Succ}_{\text{PAK}}(t'', q_{\mathsf{se}}, q_{\mathsf{ex}}, q''_{\mathsf{ro}})) \ ,
\end{aligned}$$

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

Similar bounds apply for $\mathsf{Adv}^{\text{ma-f1}}_{\text{MFPAK}}$ and $\mathsf{Adv}^{\text{ma-f2}}_{\text{MFPAK}}$.

Substituting the security statements for PAK (Appendix A.1) and PAK-Z+ (Appendix A.2) and simplifying the expressions, we obtain the following theorem describing the security of MFPAK:

Theorem 1 *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. Then* MFPAK *is a secure multi-factor password-authenticated key exchange protocol, with*

$$\mathsf{Adv}^{\text{ake-f1}}_{\text{MFPAK}}(\mathcal{A}) \le \frac{16\delta q_{\mathsf{se}}}{|\mathsf{Passwords}|} + \epsilon \quad \textit{and} \quad \mathsf{Adv}^{\text{ake-f2}}_{\text{MFPAK}}(\mathcal{A}) \le \frac{4\delta q_{\mathsf{se}}}{|\mathsf{Responses}|} + \epsilon \ ,$$

where $\epsilon = O\left(q_{\mathsf{se}}\mathsf{Adv}^{\text{cdh}}_{G,g}(t', q'^2_{\mathsf{ro}}) + q_{\mathsf{se}}\mathsf{Succ}^{\text{eu-cma}}_{\mathcal{S},\kappa}(t', q_{\mathsf{se}}) + \frac{(q_{\mathsf{se}}+q_{\mathsf{ex}})(q_{\mathsf{ro}}+q_{\mathsf{se}}+q_{\mathsf{ex}})}{|G|}\right)$ *and* $\delta = |\mathsf{Clients}| \cdot |\mathsf{Servers}|$, *for* $t' = O(t + (z + q'^2_{\mathsf{ro}} + q_{\mathsf{se}} + q_{\mathsf{ex}})t_{\mathsf{exp}})$, $q'_{\mathsf{ro}} = O(q_{\mathsf{ro}} + z + q_{\mathsf{ex}} + q_{\mathsf{se}})$, *and* $z = \max\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$*; similar bounds exist for* $\mathsf{Adv}^{\text{ma-f1}}_{\text{MFPAK}}(\mathcal{A})$ *and* $\mathsf{Adv}^{\text{ma-f2}}_{\text{MFPAK}}(\mathcal{A})$.

In Appendix C, we give an example set of parameters that instantiates MFPAK so that the advantage of an adversary running in time 2^{80} is at most 2^{-25}. In this case, MFPAK can be instantiated with 9-character passwords and a 452-bit elliptic curve, using the ECDSA signature scheme and SHA-512 hash function, where we assume that $|\mathsf{Clients}| = 2^{15}$ and $|\mathsf{Servers}| = 2^5$; q_{se}, q_{ex}, and q_{ro} are chosen reasonably.

5 Conclusion and future work

We have presented the first formal security model for multi-factor password-authenticated key exchange protocols and provided a formal argument showing that our new protocol, MFPAK, is secure in this model. Our multi-factor authentication protocol involves two factors, a long-term password and a one-time response, and and achieves two-factor security with the same server-side 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 spyware and phishing.

We also expect that there are opportunities for protocols that offer improved efficiency, have a tighter security reduction, are secure in the standard model, or are based on other cryptographic assumptions. We hope to see a wide range of multi-factor protocol designs developed by the community on the subject of multi-factor authentication.

Other recent work in the field of password-authenticated key exchange protocols has focused on protocols where the sequence of flows fits existing network protocols such as SSL/TLS. We are working on designing a two-factor password-authenticated key exchange protocol that fits within the message flow of the dominant Internet security protocol HTTPS/TLS to offer greater security for e-commerce transactions on the Internet.

While our model, in its currently stated form, only addresses two factors, it can be easily extended to accommodate additional factors if an application demands greater authentication security. We believe that an efficient protocol can be developed for more than two factors by combining additional factors in the same way as we combined aspects of the PAK and PAK-Z+ protocols.

An interesting future direction would be to integrate "fuzzy" attributes, such as biometric information, into a multi-factor authenticated key exchange protocol. Multi-factor authentication is often colloquially described as being based on 'something you know' (a password), 'something you have' (a one-time response value), and 'something you are' (a biometric attribute, such as a fingerprint). Biometric attributes can be challenging to use in a cryptographic protocol, however, because they are not perfectly reproduced every time, and thus some techniques must be used to accommodate their "fuzziness".

Acknowledgements

D.S. is 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.

References

[ABC$^+$06] Michel Abdalla, Emmanuel Bresson, Olivier Chevassut, Bodo Möller, and David Pointcheval. Provably secure password-based authentication in TLS. In Shiuh-pyng Shieh and Sushil Jajodia, editors, *Proc. 2006 ACM Symposium on Information, Computer and Communications Security (ASIACCS'06)*, pp. 35–45. ACM Press, 2006. DOI:10.1145/1128817.1128827.

[AFP05] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In Serge Vaudenay, editor, *Public Key Cryptography (PKC) 2005*, *LNCS*, volume 3386, pp. 65–84. Springer, 2005. DOI:10.1007/b105124.

[BCC$^+$06] Steve Babbage, Dario Catalano, Carlos Cid, Louis Granboulan, Tanja Lange, Arjen Lenstra, Phong Nguyen, Christof Paar, Jan Pelzl, Thomas Pornin, Bart Preneel, Matt Robshaw, Andy Rupp, Nigel Smart, and Michael Ward. ECRYPT yearly report on algorithms and keysizes (2005), January 2006. URL `http://www.ecrypt.eu.org/documents/D.SPA.16-1.0.pdf`.

[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*, May 1992. DOI:10.1109/RISP.1992.213269. URL `http://www.alw.nih.gov/Security/FIRST/papers/crypto/neke.ps`.

[BM94] Steven M. Bellovin and Michael Merritt. Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. Technical report, AT&T Bell Laboratories, c. 1994. URL `http://www.alw.nih.gov/Security/FIRST/papers/crypto/aeke.ps`.

[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)*, pp. 62–73. ACM, 1993. DOI:10.1145/168588.168596.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, **17**(2):281–308, April 1988. DOI:10.1137/0217017.

[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.

[GMR06] Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. A method for making password-based key exchange resilient to server compromise. In Cynthia Dwork, editor, *Advances in Cryptology – Proc. CRYPTO 2006*, *LNCS*, volume 4117, pp. 142–159. Springer, 2006. DOI:10.1007/11818175_9.

[Jab07] David P. Jablon. Research papers on password-based cryptography, 2007. URL `http://www.jablon.org/passwordlinks.html`.

[KM04] Neal Koblitz and Alfred J. Menezes. Another look at "provable security", 2004. EPRINT `http://eprint.iacr.org/2004/152`. Published as [KM07].

[KM06] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". II, 2006. EPRINT `http://eprint.iacr.org/2006/229`.

[KM07] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". *Journal of Cryptology*, **20**(1):3–37, 2007. DOI:10.1007/s00145-005-0432-z. Earlier version appeared as [KM04].

[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`.

[PP04] Young Man Park and Sang Gyu Park. Two factor authenticated key exchange (TAKE) protocol in public wireless LANs. *IEICE Transactions on Communications*, **E87-B**(5):1382–1385, May 2004.

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

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

[SDOF07] Stuart Schecter, Rachna Dhamija, Andy Ozment, and Ian Fischer. 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*, pp. 51–65. IEEE Press, 2007. DOI:10.1109/SP.2007.35. EPRINT `http://usablesecurity.org/emperor/`.

[Sho99a] Victor Shoup. On formal models for secure key exchange. Report RZ 3120, IBM Research, April 1999.

[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].

[TWMP06] David Taylor, Thomas Wu, Nikos Mavrogiannopoulos, and Trevor Perrin. Using SRP for TLS authentication, December 2006. URL `http://www.ietf.org/internet-drafts/draft-ietf-tls-srp-13.txt`. Internet-Draft.

[YWWD06a] Guomin Yang, Duncan S. Wong, Huaxiong Wang, and Xiaotie Deng. Formal analysis and systematic construction of two-factor authentication scheme (short paper). In Peng Ning, Sihan Qing, and Ninghui Li, editors, *Information and Communications Security*, *LNCS*, volume 4307, pp. 82–91. Springer, 2006. DOI:10.1007/11935308_7. Full version available as [YWWD06b].

[YWWD06b] Guomin Yang, Duncan S. Wong, Huaxiong Wang, and Xiaotie Deng. Formal analysis and systematic construction of two-factor authentication scheme (short paper), 2006. EPRINT `http://eprint.iacr.org/2006/270`. Short version published as [YWWD06a].

[YY06] Eun-Jun Yoon and Kee-Young Yoo. An optimized two factor authenticated key exchange protocol in PWLANs. In Vassil N. Alexandrov, Geert Dick van Albada, Peter M.A. Sloot, and Jack Dongarra, editors, *Computational Science – ICCS 2006*, *LNCS*, volume 3992, pp. 1000–1007. Springer, 2006. DOI:10.1007/11758525. URL 10.1007/11758525_133.

A Other protocols

This section gives the specifications of the PAK [Mac02] and PAK-Z+ [GMR05] protocols and the formal security statements for these protocols. The line numbering is irregular so as to emphasize the relationship between each of the PAK and PAK-Z+ protocols and the MFPAK protocol into which they are combined. The notation has been adapted from the original papers to match the notation in this paper.

A.1 PAK

The user registration stage of the PAK protocol is given below. This stage should be completed over a private, authentic channel.

PAK User Registration			
	Client C		Server S
2.	$\mathsf{re}_{C,S} \in_R \mathsf{Responses}$		
7.	$\tau' = (H_4(C, S, \mathsf{re}_{C,S}))^{-1}$		
8.		$\xrightarrow{C,\tau'}$	
10.			Store $\mathsf{re}_S[C] = \tau'$

The login stage of the PAK protocol is given below. This stage can be completed over a public, untrusted channel.

PAK Login			
	Client C		Server S
1.	$x \in_R \mathbb{Z}_q$		
2.	$X = g^x$		
4.	$\tau = H_4(C, S, \mathsf{re}_{C,S})$		
5.	$m = X \cdot \tau$		
6.		$\xrightarrow{C,m}$	
7.			Abort if $\neg\mathsf{Acceptable}(m)$
8.			$y \in_R \mathbb{Z}_q$
9.			$Y = g^y$
11.			$\tau' = \mathsf{re}_S[C]$
12.			$X = m \cdot \tau'$
13.			$\sigma = X^y$
14.			$\mathsf{sid} = \langle C, S, m, Y \rangle$
15.			$k = H_5(\mathsf{sid}, \sigma, \tau')$
18.		$\xleftarrow{Y,k}$	
19.	$\sigma = Y^x$		
21.	$\tau' = \tau^{-1}$		
22.	$\mathsf{sid} = \langle C, S, m, Y \rangle$		
23.	Abort if $k \neq H_5(\mathsf{sid}, \sigma, \tau')$		
24.	$k' = H_7(\mathsf{sid}, \sigma, \tau')$		
30.		$\xrightarrow{k'}$	
31.			Abort if $k' \neq H_7(\mathsf{sid}, \sigma, \tau')$
33.	$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \tau')$		$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \tau')$

The formal security statement for PAK is as follows:

Theorem 2 (Theorem 6.9, [Mac02]) *Let G be a finite cyclic group generated by g. 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*, and at most q_{ro} queries to the random oracles. Then, for $t' = O(t + (q_{\mathsf{ro}}^2 + q_{\mathsf{se}} + q_{\mathsf{ex}})t_{\mathsf{exp}})$,*

$$\mathsf{Adv}_{\mathrm{PAK}}^{\mathrm{ake}}(\mathcal{A}) \le \frac{q_{\mathsf{se}}}{|\mathsf{Responses}|} + O\left(q_{\mathsf{se}}\mathsf{Adv}_{G,g}^{\mathrm{cdh}}\left(t', q_{\mathsf{ro}}^2\right) + \frac{(q_{\mathsf{se}} + q_{\mathsf{ex}})(q_{\mathsf{ro}} + q_{\mathsf{se}} + q_{\mathsf{ex}})}{|G|}\right) .$$

Moreover, the same bound applies for $\mathsf{Adv}_{\mathrm{PAK}}^{\mathrm{ma}}(\mathcal{A})$.

A.2 PAK-Z+

The user registration stage of the PAK-Z+ protocol is given below. This stage should be completed over a private, authentic channel.

PAK-Z+ User Registration			
	Client C		Server S
1.	$\mathsf{pw}_{C,S} \in_R \mathsf{Passwords}$		
3.	$(V, W) \leftarrow \mathsf{Gen}(1^\kappa)$		
4.	$\gamma' = (H_1(C, S, \mathsf{pw}))^{-1}$		
5.	$V' = H_2(C, S, \mathsf{pw}) \oplus V$		
6.	$V'' = H_3(V)$		
8.		$\xrightarrow{C,\gamma',W,V',V''}$	
9.			Store $\mathsf{pw}_S[C] = \langle \gamma', W, V', V'' \rangle$

The login stage of the PAK-Z+ protocol is given below. This stage can be completed over a public, untrusted channel.

PAK-Z+ Login			
	Client C		Server S
1.	$x \in_R \mathbb{Z}_q$		
2.	$X = g^x$		
3.	$\gamma = H_1(C, S, \mathsf{pw})$		
5.	$m = X \cdot \gamma$		
6.		$\xrightarrow{C,m}$	
7.			Abort if $\neg\mathsf{Acceptable}(m)$
8.			$y \in_R \mathbb{Z}_q$
9.			$Y = g^y$
10.			$\langle \gamma', W, V', V'' \rangle = \mathsf{pw}_S[C]$
11.			$X = m \cdot \gamma'$
12.			$\sigma = X^y$
13.			$\mathsf{sid} = \langle C, S, m, Y \rangle$
14.			$k = H_5(\mathsf{sid}, \sigma, \gamma')$
15.			$a' = H_6(\mathsf{sid}, \sigma, \gamma')$
16.			$a = a' \oplus V'$
17.		$\xleftarrow{Y,k,a,V''}$	
18.	$\sigma = Y^x$		
19.	$\gamma' = \gamma^{-1}$		
21.	$\mathsf{sid} = \langle C, S, m, Y \rangle$		
22.	Abort if $k \neq H_5(\mathsf{sid}, \sigma, \gamma')$		
24.	$a' = H_6(\mathsf{sid}, \sigma, \gamma')$		
25.	$V' = a' \oplus a$		
26.	$V = H_2(C, S, \mathsf{pw}) \oplus V'$		
27.	Abort if $V'' \neq H_3(V)$		
28.	$s = \mathsf{Sign}_V(\mathsf{sid})$		
29.		$\xrightarrow{s}$	
31.			Abort if $\neg\mathsf{Verify}_W(\mathsf{sid}, s)$
32.	$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \gamma')$		$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \gamma')$

The formal security statement for PAK-Z+ is as follows:

Theorem 3 (Theorem 5.1, [GMR05]) *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*, and at most q_{ro} queries to the random oracles. Let $b_{\mathsf{co}} = 1$ if $\mathcal{A}$ makes a* CorruptPWS *query to a server, and 0 otherwise. Then, for $t' = O(t + (q_{\mathsf{ro}}^2 + q_{\mathsf{se}} + q_{\mathsf{ex}}) t_{\mathsf{exp}})$, and $\epsilon = O\left(q_{\mathsf{se}} \mathsf{Adv}^{\mathrm{cdh}}_{G,g}(t', q_{\mathsf{ro}}^2) + q_{\mathsf{se}} \mathsf{Succ}^{\text{eu-cma}}_{\mathbb{S},\kappa}(t', q_{\mathsf{se}}) + \frac{(q_{\mathsf{se}}+q_{\mathsf{ex}})(q_{\mathsf{ro}}+q_{\mathsf{se}}+q_{\mathsf{ex}})}{|G|}\right)$,*

$$\mathsf{Adv}^{\mathrm{ake}}_{\text{PAK-Z+}}(\mathcal{A}) \leq \frac{q_{\mathsf{se}}(1 - b_{\mathsf{co}}) + q_{\mathsf{ro}} b_{\mathsf{co}}}{|\mathsf{Passwords}|} + \epsilon \ .$$

Moreover,

$$\mathsf{Adv}^{\mathrm{c2s}}_{\text{PAK-Z+}}(\mathcal{A}) \leq \frac{q_{\mathsf{se}}(1 - b_{\mathsf{co}}) + q_{\mathsf{ro}} b_{\mathsf{co}}}{|\mathsf{Passwords}|} + \epsilon \ , \ \textit{and}$$

$$\mathsf{Adv}^{\mathrm{s2c}}_{\text{PAK-Z+}}(\mathcal{A}) \leq \frac{q_{\mathsf{se}}}{|\mathsf{Passwords}|} + \epsilon \ .$$

B Remainder of cases in formal analysis

This section includes the remaining three cases of the formal analysis in Section 4.

B.1 Case 2: Attacking a server instance, first factor uncompromised

This case addresses impersonation of the client when the session being attacked is a server instance and the first factor remains uncompromised.

The modifier $\mathcal{M}$ first uniformly at randomly guesses $U^* \in_R \mathsf{Servers}$ and $U'^* \in_R \mathsf{Clients}$ as its guess of who the adversary $\mathcal{A}$ will end up attacking. Let $\mathsf{GuessSC}$ be the event that the modifier $\mathcal{M}$ correctly guesses U^* and U'^*. We note that $\Pr(\mathsf{GuessSC}) = \Pr(\mathsf{GuessCS})$.

For this case, we assume that no $\mathsf{CorruptPWC}_{\text{MFPAK}}(U'^*, U^*)$ query is issued against $\mathcal{M}$: this case models client impersonation in the first factor, which is why this query is not allowed.

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

Password and response preparation. For each $(C, S) \in \mathsf{Clients} \times \mathsf{Servers}$, $\mathcal{M}$ sets $\mathsf{re}_{C,S} \in_R \mathsf{Responses}$ and constructs the corresponding $\mathsf{re}_S[C]$. In particular, $\mathcal{M}$ sets $\tau^* = H_4(U'^*, U^*, \mathsf{re}_{U^*,U'^*})$ and $\tau'^* = (\tau^*)^{-1}$. For each $(C, S) \in (\mathsf{Clients} \times \mathsf{Servers}) \setminus \{(U'^*, U^*)\}$, $\mathcal{M}$ sets $\mathsf{pw}_{C,S} = \mathsf{CorruptPWC}_{\text{PAK-Z+}}(C, S)$ and $\mathsf{pw}_S[C] = \mathsf{CorruptPWS}_{\text{PAK-Z+}}(S, C)$. Finally, $\mathcal{M}$ sets $\mathsf{pw}_{U^*}[U'^*] = \mathsf{CorruptPWS}_{\text{PAK-Z+}}(U^*, U'^*)$ (but only if $\mathcal{M}$ receives a $\mathsf{CorruptPWS}_{\text{MFPAK}}(U^*, U'^*)$ query). Of all the password and response values, only $\mathsf{pw}_{U'^*,U^*}$ remains unknown to $\mathcal{M}$.

Instantiation of PAK-Z+ simulator. We instantiate the PAK-Z+ simulator $\mathcal{S}_{\text{PAK-Z+}}$ with the following random oracles: $H^*_\ell = H_\ell$, for $\ell = 1, 2, 3$, and $H^*_\ell(\langle C, S, m, Y\rangle, \sigma, \gamma') = H_\ell(\langle C, S, m \cdot \tau^*, Y\rangle, \sigma, \gamma', \tau'^*)$, for $\ell = 5, 6, 8$. These 'starred' functions are independent random oracles if the corresponding unstarred functions are. The above construction is possible since τ^* and τ'^* are fixed and known to $\mathcal{M}$ because of the guesses made at the beginning of this case.

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

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

As before, we note that $(\mathsf{Gen}, \mathsf{Sign}^*, \mathsf{Verify}^*)$ is an eu-cma signature scheme if $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is.

$\mathcal{M}$'s handling of $\mathcal{A}$'s queries. The modifier $\mathcal{M}$ performs the following modifications to the queries of $\mathcal{A}$.

$\mathsf{CorruptPWC}(C, S)$:

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

$\mathsf{CorruptPWS}(S, C)$:

1. If $(C, S) \neq (U'^*, U^*)$:
   Return $\mathsf{pw}_S[C]$.
2. If $(C, S) = (U'^*, U^*)$:
   (a) Send a $\mathsf{CorruptPWS}(U^*, U'^*)$ query to $\mathcal{S}_{\text{PAK-Z+}}$ and receive $\mathsf{pw}_{U^*}[U'^*]$.
   (b) Return $\mathsf{pw}_{U^*}[U'^*]$.

$\mathsf{CorruptRe}(C, S)$: Return $\mathsf{re}_{C,S}$.

$\mathsf{Test}(U, i)$:

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

$\mathsf{Reveal}(U, i)$:

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

$\mathsf{Execute}(C, i, S, j)$:

1. If $(C, S) \neq (U'^*, U^*)$:
   $\mathcal{M}$ performs $\mathsf{Execute}_{\text{MFPAK}}(C, i, S, j)$

with all the values it has and returns the transcript.

2. If $(C, S) = (U'^*, U^*)$:
   $\mathcal{M}$ will use the PAK-Z+ simulator $\mathcal{S}_{\text{PAK-Z+}}$ to obtain a transcript for this query.
   (a) Send an $\mathsf{Execute}_{\text{PAK-Z+}}(C, i, S, j\rangle)$ query to $\mathcal{S}_{\text{PAK-Z+}}$ and receive $\langle C, m, Y, k, a, V'', s\rangle$.
   (b) Set $\hat{m} = m \cdot \tau^*$.
   (c) Set $\hat{k'} \in_R \text{range}(H_7)$.
   (d) Return $\langle C, \hat{m}, Y, k, a, V'', \hat{k'}, s\rangle$.

<u>$\mathsf{Send}(U, i, M)$:</u>

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

1. If $M = \langle$"start"$, S\rangle$:
   Perform $\textsc{ClientAction0}_{\text{MFPAK}}$ and return $\langle U, m\rangle$.
2. If $M = \langle C, m\rangle$ and $(C, U) \neq (U'^*, U^*)$:
   Perform $\textsc{ServerAction1}_{\text{MFPAK}}$ and return $\langle Y, k, a, V''\rangle$.
3. If $M = \langle C, m\rangle$ and $(C, U) = (U'^*, U^*)$:
   (a) Set $\hat{m} = m \cdot \gamma'^*$.
   (b) Send a $\mathsf{Send}_{\text{PAK-Z+}}(U, i, \langle C, \hat{m}\rangle)$ query to $\mathcal{S}_{\text{PAK-Z+}}$ and receive $\langle Y, k, a, V''\rangle$.
   (c) Return $\langle Y, k, a, V''\rangle$.
4. If $M = \langle Y, k, a, V''\rangle$ and $(U, U') \neq (U'^*, U^*)$, where U' is the partner of U:
   Perform $\textsc{ClientAction2}_{\text{MFPAK}}$ and return $\langle k', s\rangle$.
5. If $M = \langle Y, k, a, V''\rangle$ and $(U, U') = (U'^*, U^*)$, where U' is the partner of U:
   (a) Send a $\mathsf{Send}_{\text{PAK-Z+}}(U, i, \langle Y, k, a, V''\rangle)$ query to $\mathcal{S}_{\text{PAK-Z+}}$ and receive $\langle s\rangle$.
   (b) Set $\hat{k'} \in_R \text{range}(H_7)$ and store.
   (c) Return $\langle \hat{k'}, s\rangle$.
6. If $M = \langle k', s\rangle$ and $(U', U) \neq (U'^*, U^*)$, where U' is the partner of U:
   Perform $\textsc{ServerAction3}_{\text{MFPAK}}$.
7. If $M = \langle k', s\rangle$ and $(U', U) = (U'^*, U^*)$, where U' is the partner of U:
   (a) Abort if k' is not the same as the $\hat{k'}$ generated in Case 5 above.
   (b) Send a $\mathsf{Send}_{\text{PAK-Z+}}(U, i, \langle s\rangle)$ query to $\mathcal{S}_{\text{PAK-Z+}}$.

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 and responses 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 $\mathcal{S}_{\text{PAK-Z+}}$ can by inspection be seen to match what the MFPAK simulator would do because $\mathcal{S}_{\text{PAK-Z+}}$ is using the specially-constructed random oracles H^*. The differences between $\mathcal{M}$ and what a true MFPAK simulator would are as follows:

- $\mathsf{CorruptPWC}(C, S)$ when $(C, S) = (U'^*, U^*)$ and $\mathsf{Test}(U, i)$ when $U \neq U^*$:
  The modifier $\mathcal{M}$ aborts here, while a true MFPAK simulator should not. If $\mathcal{M}$ correctly guessed U^* and U'^* at the beginning of this case, then this query would never occur, for if it did then the session in which a Test query is directed to $\Pi_i^{U^*}$ would not be fresh.
- $\mathsf{Execute}(C, i, S, j)$ when $(C, S) = (U'^*, U^*)$, $\mathsf{Send}(U, i, M)$ when $M = \langle Y, k, a, V''\rangle$ and $(U, U') = (U'^*, U^*)$ where U' is the partner of U, and $\mathsf{Send}(U, i, M)$ 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 session instead of generating $k' = H_7(\mathsf{sid}, \sigma, \gamma', \tau')$. Since H_7 is a random oracle, this substitution is distinguishable by the adversary $\mathcal{A}$ if and only if $\mathcal{A}$ queries H_7 on the arguments $\mathsf{sid}, \sigma, \gamma', \tau'$. But if that occurs, then $\mathcal{A}$ must know γ'. These are the same inputs to the H_8^* oracle used to compute the session key in the PAK-Z+ simulation $\mathcal{S}_{\text{PAK-Z+}}$, so the same adversary could distinguish the output of $\mathsf{Test}(U^*, i)$ received from $\mathcal{S}_{\text{PAK-Z+}}$. The latter event corresponds to the

event $\mathsf{Succ}^{\text{ake}}_{\text{PAK-Z+}}$, and so the substitution is distinguishable with probability at most $\Pr(\mathsf{Succ}^{\text{ake}}_{\text{PAK-Z+}})$.

Let $\mathsf{Dist}_2 \wedge \mathsf{GuessSC}$ 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 $\Pr(\mathsf{Dist}_2 \wedge \mathsf{GuessSC}) \le 3\Pr(\mathsf{Succ}^{\text{ake}}_{\text{PAK-Z+}})$ by the argument above.

Result for case 2. Let $U^* \in \mathsf{Servers}$, $U'^* \in \mathsf{Clients}$ and let E_2 be the event that query $\mathsf{CorruptPWC}_{\text{MFPAK}}(U'^*, U^*)$ does not occur. The session involving U'^*, U^* in $\mathcal{S}_{\text{PAK-Z+}}$ is fresh if and only if the corresponding session in $\mathcal{M}$ is fresh in the first factor. Thus, if event E_2 occurs and event $\mathsf{GuessSC}$ occurs, then, whenever $\mathcal{A}$ wins against $\mathcal{M}$, $\mathcal{A}^*$ wins against $\mathcal{S}_{\text{PAK-Z+}}$, except with probability at most $\Pr(\mathsf{Dist}_2 \wedge \mathsf{GuessSC})$, since. Therefore,

$$\Pr(\mathsf{Succ}^{\text{ake-f1}}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_2 \wedge \mathsf{GuessSC}) \le \Pr(\mathsf{Succ}^{\text{ake}}_{\text{PAK-Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ ,$$

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

$$\left| \Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_2 \wedge \mathsf{GuessSC}) - \Pr(\mathsf{Succ}^{\text{ake-f1}}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_2 \wedge \mathsf{GuessSC}) \right|$$
$$\le \Pr(\mathsf{Dist}_2 \wedge \mathsf{GuessSC}) \ .$$

Combining these two expressions yields the following result:

Lemma 2 *Let $U^* \in \mathsf{Servers}$, $U'^* \in \mathsf{Clients}$, and suppose that $\mathsf{CorruptPWC}_{\text{MFPAK}}(U'^*, U^*)$ does not occur (which is event E_2). Then*

$$\Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_2 \wedge \mathsf{GuessSC}) \le 4\Pr(\mathsf{Succ}^{\text{ake}}_{\text{PAK-Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ ,$$

where $q'_{\mathsf{ro}} \le q_{\mathsf{ro}} + 1 + z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, $t' \le t + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}})$, and $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$, and a similar bound exists for $\mathsf{Adv}^{\text{c2s-f1}}_{\text{MFPAK}}$.

B.2 Case 3: Attacking a client instance, second factor uncompromised

This case addresses impersonation of the server when the session being attacked is a client instance and the second factor remains uncompromised.

The modifier $\mathcal{M}$ first uniformly at randomly 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. The event that the modifier correctly guesses these values is $\mathsf{GuessCS}$ and $\Pr(\mathsf{GuessCS})$ is given in (1).

For this case, we assume that no $\mathsf{CorruptRe}_{\text{MFPAK}}$ query is issued against $\mathcal{M}$: this case models server impersonation in the second factor, which is why this query is not allowed.

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

Password and response preparation. For each $(C, S) \in \mathsf{Clients} \times \mathsf{Servers}$, $\mathcal{M}$ sets $\mathsf{pw}_{C,S} \in_R \mathsf{Passwords}$ and $(V, W) \overset{R}{\leftarrow} \mathsf{Gen}(1^\kappa)$, and constructs the corresponding $\mathsf{pw}_S[C]$. In particular, $\mathcal{M}$ sets $\gamma^* = H_1(U^*, U'^*, \mathsf{pw}_{U^*,U'^*})$ and $\gamma'^* = (\gamma^*)^{-1}$. For each $(C, S) \in (\mathsf{Clients} \times \mathsf{Servers}) \setminus \{(U^*, U'^*)\}$, $\mathcal{M}$ sets $\mathsf{re}_{C,S} \in_R \mathsf{Responses}$. Of all the password and response values, only re_{U^*,U'^*} remains unknown to $\mathcal{M}$ at this point.

Instantiation of PAK simulator. We instantiate the PAK simulator $\mathcal{S}_{\text{PAK}}$ with the following random oracles: $H_4^* = H_4$, $H_5^*(\langle C, S, m, Y\rangle, \sigma, \tau') = H_5(\langle C, S, m\cdot\gamma^*, Y\rangle, \sigma, \gamma'^*, \tau') \,||\, H_6(\langle C, S, m\cdot \gamma^*, Y\rangle, \sigma, \gamma'^*, \tau')$, and $H_\ell^*(\langle C, S, m, Y\rangle, \sigma, \tau') = H_\ell(\langle C, S, m\cdot\gamma^*, Y\rangle, \sigma, \gamma'^*, \tau')$ for $\ell = 7, 8$. These 'starred' functions are independent random oracles if the component unstarred functions are. The above construction is possible since γ^* and γ'^* are fixed and known to $\mathcal{M}$ because of the guesses made at the beginning of this case.

$\mathcal{M}$'s handling of $\mathcal{A}$'s queries. The modifier $\mathcal{M}$ performs the following modifications to the queries of $\mathcal{A}$.

<u>$\mathsf{CorruptPWC}(C, S)$</u>: Return $\mathsf{pw}_{C,S}$.

<u>$\mathsf{CorruptPWS}(S, C)$</u>: Return $\mathsf{pw}_S[C]$.

<u>$\mathsf{CorruptRe}(C, S)$</u>: Abort; this query cannot occur in this case.

<u>$\mathsf{Test}(U, i)$</u>:

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

<u>$\mathsf{Reveal}(U, i)$</u>:

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

<u>$\mathsf{Execute}(C, i, S, j)$</u>:

1. If $(C, S) \neq (U^*, U'^*)$:
   $\mathcal{M}$ performs $\mathsf{Execute}_{\text{MFPAK}}(C, i, S, j)$ with all the values it has and returns the transcript.
2. If $(C, S) = (U^*, U'^*)$:
   $\mathcal{M}$ will use the PAK simulator $\mathcal{S}_{\text{PAK}}$ to help construct a full transcript by performing the following sequence of operations:
   (a) Send an $\mathsf{Execute}_{\text{PAK}}(C, i, S, j)$ query to $\mathcal{S}_{\text{PAK}}$ and receive $\langle C, m, Y, k, k' \rangle$.
   (b) Set
   $$\begin{aligned} \hat{m} &= m \cdot \gamma^* \\ \hat{k} &= \text{substring}_1(k) \\ \hat{a'} &= \text{substring}_2(k) \\ \hat{a} &= \hat{a'} \oplus V' \\ \hat{s} &= \mathsf{Sign}_V(\langle C, S, \hat{m}, Y \rangle) \ . \end{aligned}$$
   (c) Return $\langle C, \hat{m}, Y, \hat{k}, \hat{a}, V'', k', s \rangle$ to $\mathcal{A}$.

<u>$\mathsf{Send}(U, i, M)$</u>:

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

1. If $M = \langle \text{"start"}, S \rangle$ and $(U, S) \neq (U^*, U'^*)$:
   Perform ClientAction0$_{\text{MFPAK}}$ and return $\langle U, m \rangle$.
2. If $M = \langle \text{"start"}, S \rangle$ and $(U, S) = (U^*, U'^*)$:
   (a) Send a $\mathsf{Send}_{\text{PAK}}(U, i, \langle \text{"start"}, S \rangle)$ query to $\mathcal{S}_{\text{PAK}}$ and receive $\langle U, m \rangle$.
   (b) Set $\hat{m} = m \cdot \gamma^*$ and store.
   (c) Return $\langle U, \hat{m} \rangle$.
3. If $M = \langle C, m \rangle$ and $(C, U) \neq (U^*, U'^*)$:
   Perform ServerAction1$_{\text{MFPAK}}$ and return $\langle Y, k, a, V'' \rangle$.
4. If $M = \langle C, m \rangle$ and $(C, U) = (U^*, U'^*)$:
   (a) Set $\hat{m} = m \cdot \gamma'^*$ and store.
   (b) Send a $\mathsf{Send}_{\text{PAK}}(U, i, \langle C, \hat{m} \rangle)$ query to $\mathcal{S}_{\text{PAK}}$ and receive $\langle Y, k \rangle$.
   (c) Set
   $$\begin{aligned} \hat{k} &= \text{substring}_1(k) \\ \hat{a'} &= \text{substring}_2(k) \\ \hat{a} &= \hat{a'} \oplus V' \ . \end{aligned}$$
   (d) Return $\langle Y, \hat{k}, \hat{a}, V'' \rangle$.
5. If $M = \langle Y, k, a, V'' \rangle$ and $(U, U') \neq (U^*, U'^*)$, where U' is the partner of U:
   Perform ClientAction2$_{\text{MFPAK}}$ and return $\langle k', s \rangle$.
6. If $M = \langle Y, k, a, V'' \rangle$ and $(U, U') = (U^*, U'^*)$, where U' is the partner of U:
   (a) Set $\hat{a'} = a \oplus V'$ and $\hat{k} = k \ || \ \hat{a'}$.
   (b) Send a $\mathsf{Send}_{\text{PAK}}(U, i, \langle Y, \hat{k} \rangle)$ query to $\mathcal{S}_{\text{PAK}}$ and receive $\langle k' \rangle$ or abort.
   (c) Set $\hat{s} = \mathsf{Sign}_V(\langle U^*, U'^*, \hat{m}, Y \rangle)$ where $\hat{m}$ is the value generated in step 2.
   (d) Return $\langle k', \hat{s} \rangle$.
7. If $M = \langle k', s \rangle$ and $(U', U) \neq (U^*, U'^*)$, where U' is the partner of U:
   Perform ServerAction3$_{\text{MFPAK}}$.
8. If $M = \langle k', s \rangle$ and $(U', U) = (U^*, U'^*)$, where U' is the partner of U:
   (a) Abort if $\neg\mathsf{Verify}_W(\langle U^*, U'^*, \hat{m}, Y \rangle, s)$ where $\hat{m}$ is the value generated in step 4.
   (b) Send a $\mathsf{Send}_{\text{PAK}}(U, i, \langle k' \rangle)$ query to $\mathcal{S}_{\text{PAK}}$.

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 and responses exactly match the MFPAK specifications. Furthermore, all the generated responses exactly match the PAK 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 simulator $\mathcal{S}_{\mathrm{PAK}}$ can by inspection be seen to match what the MFPAK simulator would do because $\mathcal{S}_{\mathrm{PAK}}$ is using the specially-constructed random oracles H^*. The differences between $\mathcal{M}$ and what a true MFPAK simulator would do are as follows:

- $\mathsf{CorruptRe}(C, S)$:
  The modifier $\mathcal{M}$ aborts here, while a true MFPAK simulator should not. However, if this query did occur, then no session could be fresh in the second factor.
- $\mathsf{Test}(U, i)$ when $U \neq U^*$:
  The modifier $\mathcal{M}$ aborts here, while a true MFPAK simulator should not. If $\mathcal{M}$ correctly guessed U^* at the beginning of this case, then this query would never occur, for if it did then the session in which a Test query is directed to $\Pi_i^{U^*}$ would not be fresh.

In particular, we note that, when the event $\mathsf{GuessCS}$ occurs, the handling of $\mathcal{A}$'s $\mathsf{Execute}$ and Send queries exactly matches the behaviour and distributions of a true MFPAK simulator.

Result for case 3. Let $U^* \in \mathsf{Clients}$, $U'^* \in \mathsf{Servers}$ and let E_3 be the event that no $\mathsf{CorruptRe}_{\mathrm{MFPAK}}$ query occurs. If event E_3 occurs, the session involving U^*, U'^* in $\mathcal{S}_{\mathrm{PAK}}$ is fresh if and only if the corresponding session in $\mathcal{M}$ is fresh in the second factor. Thus, if event E_3 occurs and event $\mathsf{GuessCS}$ occurs, then, whenever $\mathcal{A}$ wins against $\mathcal{M}$, $\mathcal{A}^*$ wins against $\mathcal{S}_{\mathrm{PAK}}$. Therefore,

$$\Pr(\mathsf{Succ}_{\mathcal{M}}^{\text{ake-f2}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3 \wedge \mathsf{GuessCS}) \leq \Pr(\mathsf{Succ}_{\mathrm{PAK}}^{\text{ake}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ ,$$

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

$$\Pr(\mathsf{Succ}_{\mathrm{MFPAK}}^{\text{ake-f2}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3 \wedge \mathsf{GuessCS}) = \Pr(\mathsf{Succ}_{\mathcal{M}}^{\text{ake-f2}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3 \wedge \mathsf{GuessCS}) \ .$$

Combining these two expressions yields the following result:

Lemma 3 *Let $U^* \in \mathsf{Clients}$, $U'^* \in \mathsf{Servers}$, and suppose that no $\mathsf{CorruptRe}_{\mathrm{MFPAK}}$ query occurs (which is event E_3). 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 $\mathsf{Execute}$, respectively, and at most q_{ro} random oracle queries. Then*

$$\Pr(\mathsf{Succ}_{\mathrm{MFPAK}}^{\text{ake-f2}}(\mathcal{A}) | \mathsf{E}_3 \wedge \mathsf{GuessCS}) \leq \Pr(\mathsf{Succ}_{\mathrm{PAK}}^{\text{ake}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ ,$$

where $q'_{\mathsf{ro}} \leq 2q_{\mathsf{ro}} + 1 + 4z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, $t' \leq t + z \cdot t_{\mathsf{Gen}} + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}})$, and $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$, and a similar bound exists for $\mathsf{Adv}_{\mathrm{MFPAK}}^{\text{s2c-f2}}$.

B.3 Case 4: Attacking a server instance, second factor uncompromised

This case addresses impersonation of the client when the session being attacked is a server instance and the second factor remains uncompromised.

The modifier $\mathcal{M}$ first uniformly at randomly guesses $U^* \in_R \mathsf{Servers}$ and $U'^* \in_R \mathsf{Clients}$ as its guess of who the adversary $\mathcal{A}$ will end up attacking. The event that the modifier correctly guesses these values is $\mathsf{GuessSC}$ which has the same probability as $\mathsf{GuessCS}$.

For this case, we assume that no $\mathsf{CorruptRe}_{\mathrm{MFPAK}}$ query is issued against $\mathcal{M}$: this case models client impersonation in the second factor, which is why this query is not allowed.

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

Password and response preparation. For each $(C,S) \in \mathsf{Clients} \times \mathsf{Servers}$, $\mathcal{M}$ sets $\mathsf{pw}_{C,S} \in_R \mathsf{Passwords}$ and $(V,W) \overset{R}{\leftarrow} \mathsf{Gen}(1^\kappa)$, and constructs the corresponding $\mathsf{pw}_S[C]$. In particular, $\mathcal{M}$ sets $\gamma^* = H_1(U'^*, U^*, \mathsf{pw}_{U'^*,U^*})$ and $\gamma'^* = (\gamma^*)^{-1}$. For each $(C,S) \in (\mathsf{Clients} \times \mathsf{Servers}) \setminus \{(U'^*,U^*)\}$, $\mathcal{M}$ sets $\mathsf{re}_{C,S} \in_R \mathsf{Responses}$. Of all the password and response values, only $\mathsf{re}_{U'^*,U^*}$ remains unknown to $\mathcal{M}$ at this point.

Instantiation of PAK simulator. $\mathcal{M}$ instantiates the PAK simulator $\mathcal{S}_{\mathrm{PAK}}$ with the following random oracles: $H_4^* = H_4$, $H_5^*(\langle C,S,m,Y\rangle,\sigma,\tau') = H_5(\langle C,S,m{\cdot}\gamma^*,Y\rangle,\sigma,\gamma'^*,\tau') \,||\, H_6(\langle C,S,m{\cdot}\gamma^*,Y\rangle,\sigma,\gamma'^*,\tau')$, and $H_\ell^*(\langle C,S,m,Y\rangle,\sigma,\tau') = H_\ell(\langle C,S,m{\cdot}\gamma^*,Y\rangle,\sigma,\gamma'^*,\tau')$, for $\ell = 7,8$. These 'starred' functions are independent random oracles if the component unstarred functions are. The above construction is possible since γ^* and γ'^* are fixed and known to $\mathcal{M}$ because of the guesses made at the beginning of this case.

$\mathcal{M}$'s handling of $\mathcal{A}$'s queries. The modifier $\mathcal{M}$ performs the following modifications to the queries of $\mathcal{A}$.

$\underline{\mathsf{CorruptPWC}(C,S)}$: Return $\mathsf{pw}_{C,S}$.

$\underline{\mathsf{CorruptPWS}(S,C)}$: Return $\mathsf{pw}_S[C]$.

$\underline{\mathsf{CorruptRe}(C,S)}$: Abort; this query cannot occur in this case.

$\underline{\mathsf{Test}(U,i)}$:

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

$\underline{\mathsf{Reveal}(U,i)}$:

1. If $U = U^*$ or $U = U'^*$:
   Send a $\mathsf{Reveal}_{\mathrm{PAK}}(U,i)$ query to PAK simulator $\mathcal{S}_{\mathrm{PAK}}$ and return the result to $\mathcal{A}$.
2. Otherwise:
   Return sk for instance Π_i^U.

$\underline{\mathsf{Execute}(C,i,S,j)}$:

1. If $(C,S) \neq (U'^*,U^*)$:
   $\mathcal{M}$ performs $\mathsf{Execute}_{\mathrm{MFPAK}}(C,i,S,j)$ with all the values it has and returns the transcript.
2. If $(C,S) = (U'^*,U^*)$:
   $\mathcal{M}$ will use the PAK simulator $\mathcal{S}_{\mathrm{PAK}}$ to help construct a full transcript by performing the following sequence of operations:
   (a) Send an $\mathsf{Execute}_{\mathrm{PAK}}(C,i,S,j)$ query to $\mathcal{S}_{\mathrm{PAK}}$ and receive $\langle C,m,Y,k,k'\rangle$.
   (b) Set
   $$\begin{aligned}\hat{m} &= m \cdot \gamma^* \\ \hat{k} &= \mathrm{substring}_1(k) \\ \hat{a'} &= \mathrm{substring}_2(k) \\ \hat{a} &= \hat{a'} \oplus V' \\ \hat{s} &= \mathsf{Sign}_V(\langle C,S,\hat{m},Y\rangle)\ .\end{aligned}$$
   (c) Return $\langle C,\hat{m},Y,\hat{k},\hat{a},V'',k',s\rangle$ to $\mathcal{A}$.

$\underline{\mathsf{Send}(U,i,M)}$

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

1. If $M = \langle\text{“start”},S\rangle$ and $(U,S) \neq (U'^*,U^*)$:
   Perform $\textsc{ClientAction0}_{\mathrm{MFPAK}}$ and return $\langle U,m\rangle$.
2. If $M = \langle\text{“start”},S\rangle$ and $(U,S) = (U'^*,U^*)$:
   (a) Send a $\mathsf{Send}_{\mathrm{PAK}}(U,i,\langle\text{“start”},S\rangle)$ query to $\mathcal{S}_{\mathrm{PAK}}$ and receive $\langle U,m\rangle$.
   (b) Set $\hat{m} = m \cdot \gamma^*$ and store.
   (c) Return $\langle U,\hat{m}\rangle$.
3. If $M = \langle C,m\rangle$ and $(C,U) \neq (U'^*,U^*)$:
   Perform $\textsc{ServerAction1}_{\mathrm{MFPAK}}$ and return $\langle Y,k,a,V''\rangle$.
4. If $M = \langle C,m\rangle$ and $(C,U) = (U'^*,U^*)$:
   (a) Set $\hat{m} = m \cdot \tau'^*$ and store.
   (b) Send a $\mathsf{Send}_{\mathrm{PAK}}(U,i,\langle C,\hat{m}\rangle)$ query to $\mathcal{S}_{\mathrm{PAK}}$ and receive $\langle Y,k\rangle$.

(c) Set

$$\hat{k} = \text{substring}_1(k)$$
$$\hat{a'} = \text{substring}_2(k)$$
$$\hat{a} = \hat{a'} \oplus V' \ .$$

(d) Return $\langle Y, \hat{k}, \hat{a}, V'' \rangle$.

5. If $M = \langle Y, k, a, V'' \rangle$ and $(U, U') \neq (U'^*, U^*)$, where U' is the partner of U: Perform ClientAction2$_{\text{MFPAK}}$ and return $\langle k', s \rangle$.
6. If $M = \langle Y, k, a, V'' \rangle$ and $(U, U') = (U'^*, U^*)$, where U' is the partner of U:
   (a) Set $\hat{a'} = a \oplus V'$ and $\hat{k} = k \mid\mid \hat{a'}$.
   (b) Send a $\mathsf{Send}_{\text{PAK}}(U, i, \langle Y, \hat{k} \rangle)$ query to $\mathcal{S}_{\text{PAK}}$ and receive $\langle k' \rangle$ or abort.
   (c) Set $\hat{s} = \mathsf{Sign}_V(\langle U'^*, U^*, \hat{m}, Y \rangle)$ where $\hat{m}$ is the value generated in step 2.
   (d) Return $\langle \hat{k'}, \hat{s} \rangle$.
7. If $M = \langle k', s \rangle$ and $(U, U') \neq (U^*, U'^*)$, where U' is the partner of U: Perform ServerAction3$_{\text{MFPAK}}$.
8. If $M = \langle k', s \rangle$ and $(U, U') = (U^*, U'^*)$, where U' is the partner of U:
   (a) Abort if $\neg\mathsf{Verify}_W(\langle U'^*, U^*, \hat{m}, Y \rangle, s)$ where $\hat{m}$ is the value generated in step 4.
   (b) Send a $\mathsf{Send}_{\text{PAK}}(U, i, \langle k' \rangle)$ query to $\mathcal{S}_{\text{PAK}}$.

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 and responses exactly match the MFPAK specifications. Furthermore, all the generated passwords exactly match the PAK 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 simulator $\mathcal{S}_{\text{PAK}}$ can by inspection be seen to match what the MFPAK simulator would do because $\mathcal{S}_{\text{PAK}}$ is using the specially-constructed random oracles H^*. The differences between $\mathcal{M}$ and what a true MFPAK simulator would do are as follows:

- $\mathsf{CorruptRe}(C, S)$:
  The modifier $\mathcal{M}$ aborts here, while a true MFPAK simulator should not. However, if this query did occur, then no session could be fresh in the second factor.
- $\mathsf{Test}(U, i)$ when $U \neq U^*$:
  The modifier $\mathcal{M}$ aborts here, while a true MFPAK simulator should not. If $\mathcal{M}$ correctly guessed U^* at the beginning of this case, then these queries would never occur, for if they did then the session in which a Test query is directed to $\Pi_i^{U^*}$ would not be fresh.

In particular, we note that, when the event $\mathsf{GuessSC}$ occurs, the handling of $\mathcal{A}$'s $\mathsf{Execute}$ and Send queries exactly matches the behaviour and distributions of a true MFPAK simulator.

Result for case 4. Let $U^* \in \mathsf{Servers}$, $U'^* \in \mathsf{Clients}$ and let E_3 be (as before) the event that no $\mathsf{CorruptRe}_{\text{MFPAK}}$ query occurs. If event E_3 occurs, then the session involving U'^*, U^* in $\mathcal{S}_{\text{PAK}}$ is fresh if and only if the corresponding session in $\mathcal{M}$ is fresh in the second factor. Thus, if event E_3 occurs and event $\mathsf{GuessSC}$ occurs, then, whenever $\mathcal{A}$ wins against $\mathcal{M}$, $\mathcal{A}^*$ wins against $\mathcal{S}_{\text{PAK}}$. Therefore,

$$\Pr(\mathsf{Succ}_{\mathcal{M}}^{\text{ake-f2}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3 \wedge \mathsf{GuessSC}) \leq \Pr(\mathsf{Succ}_{\text{PAK}}^{\text{ake}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ ,$$

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

$$\Pr(\mathsf{Succ}_{\text{MFPAK}}^{\text{ake-f2}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3 \wedge \mathsf{GuessSC}) = \Pr(\mathsf{Succ}_{\mathcal{M}}^{\text{ake-f2}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3 \wedge \mathsf{GuessSC}) \ .$$

Combining these two expressions yields the following result:

Lemma 4 *Let $U^* \in \mathsf{Servers}$, $U'^* \in \mathsf{Clients}$, and suppose that no $\mathsf{CorruptRe}_{\mathrm{MFPAK}}$ query occurs (which is event E_3). 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} random oracle queries. Then*

$$\Pr(\mathsf{Succ}^{\text{ake-f2}}_{\mathrm{MFPAK}}(\mathcal{A})|\mathsf{E}_3 \wedge \mathsf{GuessSC}) \leq \Pr(\mathsf{Succ}^{\text{ake}}_{\mathrm{PAK}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ ,$$

where $q'_{\mathsf{ro}} \leq 2q_{\mathsf{ro}} + 1 + 4z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, $t' \leq t + z \cdot t_{\mathsf{Gen}} + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}})$, and $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$, and a similar bound exists for $\mathsf{Adv}^{\text{c2s-f2}}_{\mathrm{MFPAK}}$.

C Example instantiation

By careful accounting of the constants in our various lemmas and in the proofs for PAK [Mac02, Theorem 6.9] and PAK-Z+ [GMR05, Theorem 5.1], we can restate our overall result in Theorem 1 more precisely as follows.

Theorem 4 *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. Then* MFPAK *is a secure multi-factor password-authenticated key exchange protocol, with*

$$\mathsf{Adv}^{\text{ake-f1}}_{\mathrm{MFPAK}}(\mathcal{A}) \leq \frac{16\delta q_{\mathsf{se}}}{|\mathsf{Passwords}|} + \epsilon \quad \text{and} \quad \mathsf{Adv}^{\text{ake-f2}}_{\mathrm{MFPAK}}(\mathcal{A}) \leq \frac{4\delta q_{\mathsf{se}}}{|\mathsf{Responses}|} + \epsilon \ ,$$

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

To give an example instantiation, we have to pick appropriate values for the various parameters in the statement of the theorem. We choose

$$\begin{aligned} |\mathsf{Clients}| &= 2^{15} & |\mathsf{Servers}| &= 2^{5} & & \\ q_{\mathsf{se}} &= 2^{10} & q_{\mathsf{ex}} &= 2^{20} & q_{\mathsf{ro}} &= 2^{40} \\ t &= 2^{80} & t_{\mathsf{exp}} &= 2^{20} \ . & & \end{aligned}$$

With this choice of parameters, we find that $z \doteq 2^{20}$, $q'_{\mathsf{ro}} \doteq 2^{41}$, $t' \doteq 2^{105}$, and $\epsilon \doteq 2^{13} \cdot \mathsf{Adv}^{\text{cdh}}_{G,g}(2^{105}, 2^{82}) + 2^{13} \cdot \mathsf{Succ}^{\text{eu-cma}}_{\mathcal{S},\kappa}(2^{105}, 2^{10}) + \frac{2^{63}}{|G|}$.

We want $\epsilon \leq 2^{25}$. To achieve this, we need $|\mathsf{Passwords}| = |\mathsf{Responses}| = 2^{59}$, which, on the author's keyboard with 94 distinct printable characters on it, is achieved by having passwords and responses as strings of length 9.

Furthermore, we need $2^{13} \cdot \mathsf{Adv}^{\text{cdh}}_{G,g}(2^{105}, 2^{82}) \leq 2^{-26}$, $2^{13} \cdot \mathsf{Succ}^{\text{eu-cma}}_{\mathcal{S},\kappa}(2^{105}, 2^{10}) \leq 2^{-26}$, and $|G| \geq 2^{90}$. We assume that the the best technique for solving CDH is to find discrete logarithms and that, for an n-bit elliptic curve group, it takes $2^{n/2}$ time to find discrete logarithms (as in [BCC$^+$06]); in other words, $\mathsf{Adv}^{\text{cdh}}_{G,g}(t, 1) \leq \frac{t}{\sqrt{|G|}}$. Noting that $\mathsf{Adv}^{\text{cdh}}_{G,g}(t, q) \leq q\mathsf{Adv}^{\text{cdh}}_{G,g}(t, 1)$, we need $|G| \geq (2^{39}2^{82}2^{105})^2 = 2^{452}$; in other words, we need a 452-bit elliptic curve group. Furthermore, need $2^{13} \cdot \mathsf{Succ}^{\text{eu-cma}}_{\mathcal{S},\kappa}(2^{105}, 2^{21}) \leq 2^{-25}$, which is again satisfied by ECDSA with a 452-bit elliptic curve group. Finally, we can instantiate the hash function with SHA-512.

The security reduction in our theorem is not tight, meaning that in the formal security argument there is a gap between the hardness of solving the underlying cryptography problem and the hardness of breaking our protocol. The example instantiation above provides a elliptic curve group size (452 bits) that follows from the values in the formal security arguments, but is substantially larger than one might hope to use in a real-world protocol (where a 160-bit or 256-bit curve might be desirable). MFPAK instantiated with a smaller curve size may still be secure and indeed has no obvious weakness, but the formal security argument does not make any meaningful statement about the hardness of breaking our protocol with the smaller group size.[3]

[3]Interpretations of formal security arguments are the main subject of [KM07] and [KM06].