Device-Enhanced Password Protocols with Optimal Online-Offline Protection

Stanislaw Jarecki* Hugo Krawczyk† Maliheh Shirvanian‡ Nitesh Saxena§

March 29, 2017

Abstract

We introduce a setting that we call *Device-Enhanced PAKE (DE-PAKE)*, where PAKE (password-authenticated key exchange) protocols are strengthened against online and offline attacks through the use of an auxiliary device that aids the user in the authentication process. We build such schemes and show that their security, properly formalized, achieves *maximal-attainable* resistance to online and offline attacks in both PKI and PKI-free settings. In particular, an online attacker must guess the user's password *and* also corrupt the user's auxiliary device to authenticate, while an attacker who corrupts the server *cannot* learn the users' passwords via an offline dictionary attack. Notably, our solutions do not require secure channels, and *nothing* (in an information-theoretic sense) is learned about the password by the device (or a malicious software running on the device) or over the device-client channel, even without any external protection of this channel. An attacker taking over the device still requires a full *online* attack to impersonate the user. Importantly, our DE-PAKE scheme can be deployed at the user end without the need to modify the server and without the server having to be aware that the user is using a DE-PAKE scheme. In particular, the schemes can work with standard servers running the usual password-over-TLS authentication.

1 Introduction

Today, passwords constitute the prevalent authentication mechanism for bootstrapping security in online, personal and business applications, with a plethora of sensitive information depending on the security of password-based authentication. However, passwords are vulnerable to both online and offline dictionary attacks that build on password dictionaries from which a significant portion of passwords are chosen. Candidate passwords for authenticating a user to a server can be tested by an attacker through online interactions with the server. Furthermore, an attacker breaking into a server can mount an offline attack that uses information stored on the server (typically, a salted hash of the password) to test the different passwords in the dictionary. Such offline dictionary attacks are a serious concern, especially in light of frequent attacks against major commercial vendors such as PayPal [1], LinkedIn [4], Blizzard [2], Gmail [3], RSA [5, 6], and many more. The offline attacks are particularly devastating because a single server break-in may lead to

---

*U. California Irvine. Email: stasio@ics.uci.edu.

†IBM Research. Email: hugo@ee.technion.ac.il.

‡U. Alabama at Birmingham. Email: maliheh@uab.edu

§U. Alabama at Birmingham. Email: saxena@cis.uab.edu

A preliminary version of this paper appears in ACM Asia Conference on Computer and Communications Security (ASIA CCS'16).

compromising a huge number of user accounts [7]. Furthermore, since many users re-use their passwords across multiple services, compromising one service may compromise user accounts at other services.

In this paper we present solutions which enhance password protocols against *both online and offline attacks* through the use of an auxiliary device (e.g., a smartphone, smartwatch, USB token, etc.) owned by the user. We prove our solutions to be secure against an active man-in-the-middle attacker acting on user-server and user-device links, and capable of compromising devices and servers by learning their full internal state (e.g., server's password file and device's secrets).

1.1 Our Contributions

We introduce the setting of *Device-Enhanced PAKE (DE-PAKE)*, where PAKE (password-authenticated key exchange) protocols are strengthened against online and offline attacks through the use of an auxiliary device which aids the user in the authentication process. We build such schemes and show that their security, properly formalized, achieves *maximal-attainable* resistance to online and offline attacks in both PKI and PKI-free[1] settings. Most importantly, our DE-PAKE schemes can be deployed at the user end without the need to modify the server and without the server having to be aware that the user is using a DE-PAKE scheme. In particular, in the PKI setting the scheme can work with unmodified servers running the usual password-over-TLS authentication. We note that while we focus the presentation on a setting where the auxiliary device is a physical personal device, e.g. a phone, our protocols apply also to the setting where the device entity is implemented by an auxiliary on-line service.

Secure and Efficient DE-PAKE Protocols: We introduce efficient DE-PAKE protocols with the following properties:

• *Resistance to online and offline attacks:* Our DE-PAKE schemes provide *maximal-attainable* security in terms of resistance to both online and offline attacks. That is, the only attack allowed by the scheme is the *unavoidable* online guessing attack where the attacker tests if a given value p is the user U's password by interacting with *both* device D and server S in the role of U with password p and observing whether S accepts. In other words, each guess attempt requires the attacker to interact on both U-D and U-S links. No amount of attacking on one link helps without a corresponding attack on the other. Moreover, fully compromising D still requires a full online guessing attack on the U-S link, and fully compromising S requires a full online guessing attack on the U-D link. And even if both S and D are compromised, a full offline dictionary attack is required. More formally, to have an impersonation probability of $q/|\mathsf{Dict}|$, where Dict is a passwords dictionary from which U chooses its password, the attacker needs to either run q online interactions with D *and* q online interactions with S, or q online interactions with U impersonating both D and S. Moreover, even when compromising D and finding all its secrets, the attacker still needs to run q online interactions with either S or U, and if the server is compromised, q online interactions with D or U. Finally, if both D and S are compromised the adversary must stage a full offline dictionary attack to learn any passwords. We formalize these properties through a security model, DE-PAKE, that extends the traditional PAKE security setting, and then we use this model to prove the security of our schemes.

• *PKI-agnostic:* We show that the above strong security can be achieved even when the user-server channel is not protected by a server public key. On the other hand, when such protection is available one obtains the additional benefit that impersonating the server to the user is infeasible even if the user's password is

---

[1] In the client-server setting one considers both PKI-based (say, via TLS) password authentication [19] where the client possesses a server's certificate in addition to the password, and the PKI-free case where no such certificate or other form of secure channels is assumed for user login [18]. In either case, password registration (being much less frequent than regular login) can use special safeguards, such a PKI-based interactions or out-of-band channels.

disclosed. Luckily, our schemes can work, without modification and without having to be aware of it, with PKI-based and PKI-free authentication protocols, providing in each case the best attainable offline-online protection.

• *Modularity and server-transparency:* Our design is modular allowing for the use of independent device and server components, in particular enabling the use of our scheme with existing password protocols and *without the need to modify the server side.* Indeed, one of the core results in the paper is a *composition theorem* showing that the composition of our user-device protocol (we call it PTR) with *any* user-server PAKE scheme[2] results in a secure DE-PAKE protocol. A case of particular practical significance is when the user-server PAKE is implemented via the standard password-over-TLS protocol (this particular setting requires the communication from device to user to be authenticated - see Section 4.5[3]). Moreover, a single device and the same device-user protocol can serve passwords for different services, including services using different user-server PAKE protocols, with the user having to remember a single (master) password for all these services (see Section 5). As a particular application of these results, in [34] we show how to build, using the DE-PAKE framework, a smartphone-based password manager application with outstanding and unique security properties.

1.2 Related Work

Prior Work on Device-Enhanced Authentication. Our approach is closely related to the work of Acar, Belenkiy, and Kupcu [8], and Boyen [13]. A close variant of the key ingredient of our solution, a "Password-to-Random" (PTR) protocol, was proposed as a "Hidden Credential Retrieval" (HCR) scheme by [13]. Our DE-PAKE protocol has similar goals as the "Single Password Authentication" (SPA) scheme of [8], namely, strengthening password authentication by making offline dictionary attack against the server infeasible. Moreover, one of their schemes does so in roughly the same way as we do, following an approach first suggested by Ford-Kaliski [16], i.e. having the user "strengthen" her password into a strong secret via the HCR/PTR protocol with an auxiliary device (or service), and then use this strong secret to authenticate to the primary server. Despite these similarities, our work improves on [8, 13] in the following aspects: (1) Whereas [13] shows that HCR can be realized almost identically to our PTR, i.e. using the same variant of the Ford-Kaliski protocol [16], the SPA scheme of [8] did not show a general compiler from HCR, but assumed a stronger tool of unique blind signature which is costlier to realize (moreover, in their scheme the user-device blind signature protocol goes over TLS, while in ours the PTR protocol goes over insecure links); (2) The SPA scheme of [8] is weaker than our DE-PAKE, in that it achieves only entity authentication rather than authenticated key exchange as in our case and it assumes PKI while our DE-PAKE works in both PKI-free and PKI settings;[4] (3) Very importantly, our modular approach that treats PTR and PAKE as independent components and has no involvement of the server in the PTR execution, allows us to obtain DE-PAKE security without any modification to existing servers (e.g., those running password-over-TLS authentication). In contrast, the SPA scheme of [8] requires the server to perform additional public key operations.

Relation to T-PAKE and 2-PAKE. The DE-PAKE model that we introduce is closely related to 2-PAKE, the two-server case of Threshold PAKE (*T-PAKE*) [32]. Our model can be seen as a *main-server/auxiliary-server* version of 2-PAKE, where the client establishes an independent session key only with the single

---

[2] More precisely, any PAKE scheme that finding the user's password requires an offline dictionary attack even upon server compromise - see Section 3.2.

[3] In the previous version of this paper we overlooked this requirement that is specific to the password-over-TLS case.

[4] Our DE-PAKE security model is also more precise, e.g. letting adversary observe whether user authentication succeeds or not.

designated server. In DE-PAKE, the role of the second server is played by the device, and its role is solely to provide security against offline dictionary attacks in case the server is corrupted. The major advantage of the DE-PAKE model is that it enables modular realizations where the user-server authentication can use *any* password protocol, including standard password-over-TLS. In this case, one cannot just use 2-PAKE solutions as those would require a dedicated protocol on the server side too. See Section 3.3 for more details on modeling DE-PAKE as a case of T-PAKE.

Interestingly, the main-server/auxiliary-server 2-PAKE protocols implied by our work are more efficient than previously known 2-PAKE's (modified so that only one server establishes a session key). Indeed, assuming PKI and assuming that the user-server protocol proceeds over a pre-established TLS session, our DE-PAKE instantiation (see Section 5) requires 2 exponentiations for user and 1 for device, while to the best of our knowledge, the DE-PAKE implied by the most efficient existing 2-PAKE in that setting [14, 36], would take 2 exp's for server and 2 for device, assuming pre-estalished TLS sessions. The total costs *including* TLS session establishment would be even better in our case because in the 2-PAKEs of [14, 36] the client needs TLS sessions with both servers, while in our DE-PAKE the client needs a TLS session only with the main server. In the *PKI-Free* model, our DE-PAKE instantiation (see Section 5) requires 6 exponentiations for user, 1 for device, and 3 for the server, which to the best of our knowledge beats prior non-PKI 2-PAKE schemes[5]. Another point of comparison is the required cryptographic assumptions. While our protocol requires the One-More Diffie-Hellman (OMDH) interactive assumption in the Random Oracle Model (ROM), the 2-PAKE's of [14, 36] are secure under Decisional Diffie-Hellman (DDH) in ROM. We also note that there are 2-PAKE schemes secure in the standard model [9, 21, 28–30], but they are much less efficient than our ROM-based constructions.

Relation to Work on Two-Factor Authentication. One important and increasingly common line of defense against password attacks is the use of *two-factor password authentication (TFA) schemes.* TFA mechanisms are used for authenticating a user U to a server S where the user has a password and a personal device D that contains some secret auxiliary information. This secret information is used to provide security even in the case that the user's password leaks (e.g., by an attack on the client machine). Our DE-PAKE scheme offers several defenses against attacks on the client machine as we describe in Section 5 but does not add protection against the leakage of a user's password. For such protection, it is possible to combine a DE-PAKE scheme with a traditional two-factor mechanism that takes advantage of the presence of an auxiliary device (e.g. a one-time PIN delivered to or generated by the hand-held device).

In this context, it is worth recalling the work of Shirvanian et al. [35] that addresses the issues of two-factor authentication and defense against offline attacks in one combined solution. The main idea underlying their protocols is for the server to store a randomized hash of the password, $h = H(p,\ s)$, and for the device to store the corresponding random secret s. For authentication, the user sends both s and p to the server which makes the scheme crucially dependent on the PKI infrastructure over which the password is transmitted. The scheme also requires a special-purpose protocol on the server side and shared information between server and device. While we do not provide two-factor authentication, we improve on the latter two counts by accommodating PKI-free protocols as well as not requiring any changes on the server side.

---

[5] A recent construction of T-PAKE implied by the PPSS scheme from [24] would provide a more efficient non-PKI 2-PAKE than the non-PKI DE-PAKE construction presented here, but it would not satisfy the modularity and server transparency properties of the DE-PAKE protocol.

Table 1: Glossary

Acronym	Description
PAKE	Password-authenticated key exchange. We refer to password protocols by PAKE as well as to their security model (recalled in Section 3.1).
PKI-Free PAKE	A PAKE protocol that does not assume any secret or authenticated key carried by the user other than its own password. In particular, no PKI-based server-authentication is assumed. This is also known as the CRS or password-only model.
DE-PAKE	A new notion of *Device-Enhanced PAKE* protocols that guarantees optimal resilience to offline and online attacks upon compromise of device and/or server. We use DE-PAKE to refer to the security model (Section 3.2) as well as to the constructions satisfying this model (Sections 4 and 5) .
PTR	A security notion and model for password hardening protocols. We use PTR (password-to-random) to refer to the security model (Section 4.2) as well as to the constructions satisfying this model (Section 4.1).
FK-PTR	Specific PTR construction using the Ford-Kaliski password hardening technique (Section 4.1).
OPRF	Oblivious PRF (see Section 4.1) is the basis for the FK-PTR protocol. When implemented via the function $F_k(x) = H(x, (H'(x))^k)$ we obtain FK-PTR.
PTR-PAKE	A general name for DE-PAKE protocols built by composing a PTR and a PAKE protocols (their generic security is based on Theorem 3)
FK-PTR-PAKE	A PTR-PAKE scheme where the PTR part is implemented with the FK-PTR construction. It is also the name of the protocol in Fig 3 that combines FK-PTR with the PKI-free PAKE protocol of [22, 23].

1.3 Organization and Glossary

We overview our design and proof methodology in Section 2, with full details given in Section 4. Section 3 presents our formal DE-PAKE security model on which we base our analysis. Section 5 presents protocol instantiations and extensions: A fully specified DE-PAKE scheme secure in the PKI-free (i.e. CRS) model; a description of our approach for armoring existing servers against online and offline attacks while keeping the servers unmodified; and extensions which address issues related to client security (which are not covered in our DE-PAKE model). For easy reference, in Table 1 we provide a summary of the main terms and acronyms used throughout the paper.

2 Overview: DE-PAKE Design and Analysis

Our design follows the "password hardening" approach of Ford and Kaliski [16], but dispenses of authenticated channels (other than during a registration phase), multiple servers and/or other safeguards that were required for the secure use of these techniques in prior work [16, 20].

The idea is simple: the user memorizes a regular password pwd but uses as her password with server S a value $\mathsf{rwd} = F_k(\mathsf{pwd})$ where F is a pseudorandom function and k is a key held by device (or an auxiliary online service) D (rwd is a mnemonics for "randomized password"). Before authenticating to S, U contacts D (through a client application) and obtains rwd via a special protocol with D (in which D learns *nothing* about pwd or rwd). U then authenticates to S via a (standard) PAKE protocol using rwd as a password. Note that without knowledge of k, the value rwd has full entropy (i.e., it is indistinguishable from uniform in the range set of function F) hence dictionary attacks do not apply against rwd (neither online or offline attacks, not even if the server is compromised). Moreover, we will ensure that even if D (or S) is compromised, offline attacks against pwd are infeasible. Thus, the challenge is in implementing the protocol between U and D, to which we refer as *PTR (for Password-to-Random)*, so that U can compute $F_k(\mathsf{pwd})$ but the protocol leaks no other information about protocol inputs, i.e. pwd and k. Note that we cannot assume an authenticated or secret channel between the client and D since this would either require knowing a device public key or storing pwd-related information at D (the latter would open pwd to an offline dictionary attack upon compromising D). We show that in spite of the client-device link being unauthenticated, hence controlled by the attacker, the "blinded DH" approach of Ford-Kaliski, (see Section 4.1), can be used to implement the PTR protocol.

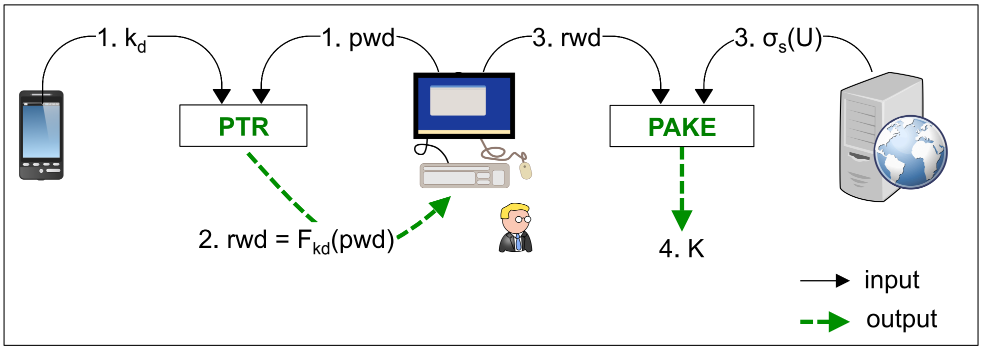

Figure 1: PTR-PAKE Authentication Phase

Hence, we obtain a DE-PAKE scheme, which we call *PTR-PAKE*, as the composition of a PTR protocol and a secure PAKE. We depict this generic construction of a DE-PAKE protocol in Figure 1. The output K in step 4 denotes the session key established between the user and the server, i.e. the output of the DE-PAKE protocol, which is equal to the output of the PAKE subprotocol. The server's input $\sigma_{\mathsf{S}}(\mathsf{U})$ to the PAKE subprotocol denotes the user-specific information stored at S created in the PAKE initialization using rwd in the role of the password (e.g. in a common PKI-based PAKE $\sigma_{\mathsf{S}}(\mathsf{U})$ would be a salted hash of rwd).

In order to prove the security of such scheme, we first extend the established security models for the PAKE functionality to the DE-PAKE setting (Section 3). For the DE-PAKE modeling we consider a fully capable man-in-the-middle attacker active on all the links between all parties and one who is allowed to compromise servers and devices at will. No external source of authentication is assumed other than the user's password (except for secure registration of a user with the server and device). Second, we define the security requirements from a PTR protocol and show a PTR instantiation, FK-PTR, that satisfies this definition in the random oracle model. Finally, we prove a *generic security composition theorem* showing that the composition of a secure PTR scheme (run between U and D on the basis of the user's password pwd) with a secure PAKE protocol (run between U and S on the basis of the hardened password rwd $= F_k(\mathsf{pwd})$) results in a secure DE-PAKE scheme, provided that the PAKE protocol satisfies "security against server compromise" or the more precise notion of KCI resistance introduced and discussed in Section 3.2 (we note that typical protocols that store a salted version of the password satisfy this property).

Since our FK-PTR scheme does not require PKI, using a PKI-free PAKE in the above composition results in a DE-PAKE protocol that does not rely on public keys or other secure channels.

In order to demonstrate full standalone solutions, in Section 5 we describe two instantiations of our PTR-PAKE construction of a DE-PAKE scheme. First, we compose our FK-PTR scheme with a specific PAKE protocol - a KCI-resistant version of the single-server variant of threshold PAKE from [22, 23]. Since this PAKE protocol does not require PKI, neither does the resulting DE-PAKE protocol. Secondly, we expand on the fact that since our PTR-PAKE construction can be used with *any* existing password protocol with resistance to KCI attacks, one obtains DE-PAKE schemes without changing the server that implements the

PAKE protocol. In particular, this allows us to use without any change a server that implements the standard PKI-based password-over-TLS protocol, but in this case the outgoing communication from the device needs to be authenticated (see Section 4.5).

A PTR scheme is a close variant of OPRF, and the potential of OPRF to strengthen password authentication was recently used in the Pythia system [15], but their proposal differs from ours in at least three ways: (1) Their OPRF scheme is significantly more costly than our PTR: it requires 6 exponentiations and a bilinear map compared to 3 exponentiations in our PTR scheme (and we can use elliptic curves without bilinear maps where exponentiations are cheaper); (2) Their solution relies on PKI, while ours does not; (3) Their solution does not offer a server-transparent instantiation.

3 Security Model

We introduce the *Device-Enhanced PAKE (DE-PAKE)* security model under which we prove the security of our schemes. The model extends the standard PAKE (Password Authenticated Key Exchange) formalisms to include user-specific devices and formulates a security definition that guarantees maximal online and offline security of password protocols. We start by recalling the PAKE security model (adapted to the client-server setting) and then we present the extension of the PAKE model to the DE-PAKE setting.

3.1 PAKE Security Model

We recall the security model for PAKE (Password-based Authenticated Key Exchange) protocols, based on the model of Bellare, Pointcheval and Rogaway [11] that extends authenticated key exchange models to account for the inherent vulnerability of password protocols to online guessing attacks. We adapt the PAKE model to the client-server setting borrowing some of the formalism from [22]. In Section 3.2 we will extend this standard PAKE model to security against server compromise.

Protocol participants. There are two types PAKE protocol participants, users and servers. Each user U is associated with a unique server S while servers may be associated with multiple users.

Protocol execution. A PAKE protocol has two phases: initialization and key exchange. In the initialization phase each user U chooses a random password pwd from a given dictionary Dict and interacts with its associated server S producing a user's state $\sigma_{\mathsf{S}}(\mathsf{U})$ that S stores while U only remembers its password pwd. *Initialization is assumed to be executed securely, e.g., over secure channels.* In the key exchange phase, users interact with servers over insecure (adversary-controlled) channels to establish session keys. Both users and servers may execute the protocol multiple times in a concurrent fashion. Each execution of the PAKE protocol by U or S defines a (user or server) protocol *instance*, also referred to as a protocol *session*, denoted respectively Π_i^{U} or Π_i^{S}, where integer pointer i serves to differentiates between multiple protocol instances executed by the same party. Each protocol session is associated with the following variables: a *session identifier* sid, which we equate with the message transcript observed by this instance (where both U and S order their interaction transcripts starting with U's message), a *peer identity* pid, and a *session key* sk. For a user instance the peer is always the user's server while for a server instance the peer is the user authenticated in the session. The output of an execution consists of the above three variables which can be set to $\perp$ if the party aborts the session (e.g., when authentication fails, a misformed message is received, etc.). When a session outputs $\mathsf{sk} \neq \perp$ we say that the session *accepts*.

PAKE Security. To define security we consider a probabilistic attacker A which schedules all actions in the protocol and controls all communication channels with full ability to transport, modify, inject, delay or drop messages. In addition, the attacker knows (or even chooses) the dictionaries used by users. The model defines the following queries or activations through which the adversary interacts with, and learns information from, the protocol's participants.

$\mathsf{send}(P, i, P', M)$: Delivers message M to instance Π_i^P purportedly coming from P'. In response to a send query the instance takes the actions specified by the protocol and outputs a message given to A. When a session accepts, a message indicating acceptance is given to A. A send message with a new value i (possibly with null M) creates a new instance at P with pid P'. For simplicity, we assume that the pair $\{P, P'\}$ in any send message contains a user and the server associated to that user (a non-compliant message causes the receiving instance to abort). The send query can also create a new instance of party P: If Π_i^{U} does not exist then query $\mathsf{send}(\mathsf{U}, i, \mathsf{S}, \mathsf{init})$ creates a new instance Π_i^{U} which executes with pid $= \mathsf{S}$ on U's chosen password pwd. Similarly, if Π_i^{S} does not exist then $\mathsf{send}(\mathsf{S}, i, \mathsf{U}, M)$ creates a new instance Π_i^{S} which executes with pid $= \mathsf{U}$ on S's input $\sigma_{\mathsf{S}}(\mathsf{U})$, with U's first message set to M. (This formalism assumes that protocol exchanges are initiated by users, which is the operational setting in PAKE.)

$\mathsf{reveal}(P, i)$: If instance Π_i^P has accepted, outputs the respective session key sk; otherwise outputs $\perp$.

$\mathsf{corrupt}(P)$: Outputs all data held by party P and A gains full control of P. We say that P is *corrupted.*

$\mathsf{compromise}(\mathsf{S}, \mathsf{U})$: Outputs state $\sigma_{\mathsf{S}}(\mathsf{U})$ at S. We say that S is U-*compromised.*

$\mathsf{test}(P, i)$: If instance Π_i^P has accepted, this query causes Π_i^P to flip a random bit b. If $b = 1$ the instance's session key sk is output and if $b = 0$ a string drawn uniformly from the space of session keys is output. A test query may be asked at any time during the execution of the protocol, but may only be asked once. We will refer to the party P against which a test query was issued and to its peer as the *target parties*.

The following notion taken from [22] is used in the security definition below to ensure that legitimate messages exchanged between honest parties do not help the attacker in online password guessing attempts (only adversarially-generated messages count towards such online attacks). It has similar motivation as the execute query in [11], but the latter fails to capture the ability of the attacker to delay and interleave messages from different sessions.

Rogue send *queries/activations:* We say that a $\mathsf{send}(P, i, P', M)$ query is *rogue* if it was not generated and/or delivered according to the specification of the protocol, i.e. message M has been changed or injected by the attacker, or the delivery order differs from what is stipulated by the protocol (delaying message delivery or interleaving messages from different sessions is not considered a rogue operation as long as internal session ordering is preserved). We also consider as rogue any $\mathsf{send}(P, i, P', M)$ query where P is uncorrupted and P' is corrupted. We refer to messages delivered through rogue send queries as *rogue activations* by A.

Matching sessions. A session in instance Π_i^P and a session in instance $\Pi_j^{P'}$ are said to be *matching* if both have the same session identifier sid (i.e., their transcripts match), the first has pid $= P'$, the second has pid $= P$, and both have accepted.

Fresh sessions. A session at instance Π_i^P with peer P' s.t. $\{P, P'\} = \{\mathsf{U}, \mathsf{S}\}$ is called *fresh* if none of the queries $\mathsf{corrupt}(\mathsf{U})$, $\mathsf{corrupt}(\mathsf{S})$, $\mathsf{compromise}(\mathsf{S}, \mathsf{U})$, $\mathsf{reveal}(P, i)$ or $\mathsf{reveal}(P', i')$ were issued, where $\Pi_{i'}^{P'}$ is an instance whose session matches Π_i^P (if such $\Pi_{i'}^{P'}$ exists).

Correctness. Matching sessions between uncorrupted peers output the same session key.

Attacker's advantage. Let PAKE be a PAKE protocol and A be an attacker with the above capabilities running against PAKE. Assume that A issues a single test query against a fresh session at a user or server and ends its run with an output bit b'. We say that A *wins* if $b' = b$ where b is the bit chosen internally by the test session. The *advantage of* A *against* PAKE is defined as $\mathsf{Adv}_{\mathsf{A}}^{\mathsf{PAKE}} = 2 \cdot Pr\,[\mathsf{A} \text{ wins against PAKE}] - 1$.

Definition 1. *A PAKE protocol* PAKE *is* (q_S, q_U, T, ϵ)-secure *if it is correct and for any password dictionary* Dict *and any attacker* A *that runs in time T, it holds that* $\mathsf{Adv}_{\mathsf{A}}^{\mathsf{PAKE}} \le \frac{q_U + q_S}{|\mathsf{Dict}|} + \epsilon$ *where q_U is the number of* *rogue* send *queries having the target user* U *as recipient and* q_S *is the number of rogue* send *queries having the target* S *as recipient.*

Dictionary size 2^d. Our treatment works for any dictionary size, but for notational convenience we denote it as 2^d.

3.2 Security against server compromise and KCI-resistance

In the asymmetric client-server setting of password authentication that concerns us in this paper, plain passwords should not be stored at the server, so as to prevent the leakage of the password in case of server compromise. Instead, the server should store some other verification information corresponding to this password, such as the salted password hash. The security requirement in this case, often referred to as *security against server compromise* [18], is that access to the server's state for a particular user (i.e, U-compromise in our terminology) does not allow the attacker to authenticate that user to the server except after running an offline dictionary attack that recovers the password given the server's state.[6] In the key-exchange literature an attack in which the compromise of a party P allows the attacker to falsely authenticate another party P' to P is called a *Key-Compromise Impersonation (KCI) attack* [12]. Therefore, the above notion of security against server compromise can be seen as a *weak form of KCI resistance,* where impersonation of U to S is possible but only after running an offline dictionary attack.

We extend the above PAKE formalism to capture resistance to weak KCI attacks (*wKCI-resistance*) through the following game (which is well-suited to ROM-based implementations that hash the password). The security experiment is as before except for the following changes. User U (associated with server S) chooses its password at random from a dictionary Dict, where Dict is a random subset of $\{0,1\}^\tau$ of size 2^d (for integers $d<\tau$). The attacker A is given a random subset of Dict of size q as well as the server's state $\sigma_{\mathsf{S}}(\mathsf{U})$, and it must choose the test session at an instance of S with peer U (in the regular case this is not allowed since S is U-compromised). We call a PAKE scheme *ϵ-wKCI-resistant* if for any $q \leq 2^d$, the attacker's advantage in this game is at most $q/2^d + \epsilon$.

A strong notion of KCI resistance is achieved in the DE-PAKE model as we will see next.

3.3 DE-PAKE Security Model

We extend the PAKE model to the DE-PAKE setting. Besides servers and users in the PAKE model, each user is associated with a device D with which it communicates over a two-way link. (We stress that the role of D can be played by any data-connected entity, including a hand-held device or an auxiliary web service.) The initialization phase of PAKE is extended to include the user-device communication that establishes the state stored at D. As before, users only remember their passwords. As in the PAKE case, initialization (including the user-device interaction) is assumed to run over secure channels. After initialization, the links between users and devices are subject to the same man-in-the-middle adversarial activity as in the links between users and servers. Device instances Π_i^{D} are created similarly to user and server instances, and are activated by A via send queries that include users and devices as senders and receivers. However, device instances do not produce output other than the outgoing messages. In particular, reveal queries do not apply to them, but corrupt queries can be issued against devices, in which case the internal state of the device is revealed to A who then controls the device. The session-related notions, including the test query, do not apply to devices.

---

[6] Recovering the password via an offline dictionary attack is unavoidable in the PAKE model. Also unavoidable is impersonating S to U when S is U-compromised (except if one assumes, as in the PKI model, an independent authenticated channel from S to U).

The attacker's goal is the same as before, i.e. to win the test experiment at a user or server instance, as in the PAKE setting. Also the correctness property is unchanged. However, to the attacker resources we add the number of *rogue* send queries (see Section 3.1) where the target user is the recipient and the device the sender (denoted q'_U) and the number of *rogue* send queries where the target user is the sender and the device the recipient (denoted q_D). We refer to this more powerful adversary as a DE-PAKE attacker.

Strong KCI resistance. The DE-PAKE model is intended to provide a much stronger notion of security in case of server compromise than achievable in the PAKE case. While in the latter, impersonating U to S in case of U-compromise is possible (and unavoidable) through an offline dictionary attack, in DE-PAKE protocols this is prohibited. In order to formalize this requirement we follow the treatment of KCI resistance from [31] and we strengthen the capabilities of a DE-PAKE attacker through a more liberal notion of fresh sessions at a server S. All sessions considered *fresh* in the PAKE model are also considered fresh in the DE-PAKE model; in addition, in the DE-PAKE model, a session Π_i^{S} at server S with peer U is considered fresh *even if* corrupt(S) *or* compromise(S, U) *were issued* as long as all other requirements for freshness are satisfied and *the attacker* A *does not have access to the temporary state information created by session* Π_i^{S}. This relaxation of the notion of freshness captures the case where the attacker A might have corrupted S and gained access to S's secrets (including long-term ones), yet A is not actively controlling S during the generation of session Π_i^{S}. In this case we would still want to prevent A from authenticating as U to S on that session. Definition 2 (item 2) below ensures that this is the case for DE-PAKE secure protocols even when *unbounded* offline attacks against S are allowed.

The following security definition captures the maximal-attainable online and offline security from a DE-PAKE protocol as informally discussed in the introduction. Let DPK be a DE-PAKE protocol and A be an attacker with the above capabilities running against DPK. As in the PAKE model, we assume that A issues a single test query against some U or S session, that A output bit b', and we say that A wins if $b' = b$ where b is the bit chosen by the test session. We define $\mathsf{Adv}_{\mathsf{A}}^{\mathsf{DPK}} = 2 \cdot Pr\,[\mathsf{A} \text{ wins against } \mathrm{DPK}] - 1$.

Definition 2. *A DE-PAKE protocol is called* $(q_S, q_U, q'_U, q_D, T, \epsilon)$*-secure if it is correct, and for any password dictionary* Dict *of size* 2^d *and any attacker that runs in time* T*, the following properties hold (for* q_S, q_U, q'_U, q_D *as defined above):*

1. *If* S *and* D *are uncorrupted, the following bound holds:*

$$\mathsf{Adv}_{\mathsf{A}}^{\mathsf{DPK}} \leq \frac{\min\{q_U + q_S, q'_U + q_D\}}{2^d} + \epsilon. \tag{1}$$

2. *If* D *is corrupted then* $\mathsf{Adv}_{\mathsf{A}}^{\mathsf{DPK}} \leq (q_U + q_S)/2^d + \epsilon$.
3. *If* S *is corrupted then* $\mathsf{Adv}_{\mathsf{A}}^{\mathsf{DPK}} \leq (q'_U + q_D)/2^d + \epsilon$.
4. *When both* D *and* S *are corrupted, expression (1) holds but* q_D *and* q_S *are replaced by the number of offline operations performed based on* D*'s and* S*'s state, respectively.*

Note that the bounds in items 3 and 4 hold also when S is U-compromised (since being corrupted implies U-compromise for all users U associated with S).

Note *(more general bounds).* One can define the above bounds more generally by replacing the expression $\min\{q_U + q_S, q'_U + q_D\}/2^d + \epsilon$ with a tighter bound (as in equation (3) in the proof of Theorem 3). We choose the simpler and more natural expression (1) that can be achieved by the adversary via generic attacks (e.g., $q_U + q_S$ is achievable when A plays man-in-the-middle between S and D on a guessed password, and $q'_U + q_D$ is achievable when A acts between U and D on the guessed password). Finally, we note that item 2 (resp. 3) could be covered by (1) if one replaced q_D (resp. q_S) in this expression with the number of offline operations performed based on D's (resp. S's) state.

Note on modeling DE-PAKE via a T-PAKE. In a (t, n)-Threshold-PAKE (T-PAKE) (cf. [22]), a user holding a single password can securely establish authenticated keys with a subset of n servers as long as no more than t of them are corrupted (and the user interacts with at least $t + 1$ well-behaving servers). One can implement DE-PAKE on the basis of a T-PAKE for $(t, n)=(1, 2)$, otherwise known as 2-PAKE, by letting D and S act as the two servers in the 2-PAKE scheme. This would imply the first three conditions of Definition 2, but the last condition should be added as an additional requirement. Moreover, one can use 2-PAKE as the basis for the definition of DE-PAKE where the user only authenticates to one of the servers. However, the dedicated DE-PAKE definition we present, and its instantiations, provide several advantages: (1) It makes the security goals for the DE-PAKE notion clearer; (2) It allows for a more precise specification of the (strict) upper bounds on attacker's advantage depending on the attack setting; and (3) It allows for more efficient implementations, in particular enabling a server-transparent DE-PAKE implementation, which cannot be done using 2-PAKE. (A 2-PAKE cannot be server-transparent because if S runs the code as in PAKE then S's presence cannot help U to authenticate to D.)

Note on client security. The DE-PAKE model is designed to capture (maximal) security against online and offline attacks where the attacker fully controls all communication channels and can compromise servers and devices. However, as it is customary in the PAKE setting, the model does not consider the security of the machine (the "client") into which the user enters the password. Yet, our solutions, while vulnerable to some forms of attack by an attacker controlling the client machine, also provide defenses to common attacks such as keyloggers or phishing attacks (see Section 5).

4 A modular DE-PAKE Scheme

In this section we present and analyze our generic DE-PAKE scheme, i.e. the PTR-PAKE shown in Figure 1, which results from the composition of two independent cryptographic primitives, a PTR protocol and a PAKE protocol with resistance to wKCI attacks (see section 3.1). For a high-level description of the functionality of a PTR (password-to-random) scheme and its use for obtaining a DE-PAKE scheme see Section 2. We start by describing a specific efficient PTR implementation we call FK-PTR, with is based on the "password hardening" protocol of Ford-Kaliski [16] (Section 4.1). We then use this protocol example to formalize the PTR notion and its security requirements (Section 4.2), and we prove that the FK-PTR protocol satisfies the PTR security notion (Section 4.3). Finally, we prove that the *generic* composition of any secure PTR scheme and any PAKE scheme with resistance to wKCI attacks results in a secure DE-PAKE scheme (Section 4.4). Thus, our scheme can be instantiated with the FK-PTR scheme as the PTR part and any secure wKCI-resistant PAKE protocol (e.g., [18, 22]). Moreover, if the PAKE scheme is in the password-only model[7] then the DE-PAKE scheme is also secure in this model.

4.1 The FK-PTR Scheme

The instantiation of a PTR scheme we call FK-PTR is based on Ford-Kaliski's "password hardening" [16] or its more general interpretation as an Oblivious PRF (OPRF) [17, 26, 27]. Roughly, an OPRF is a pseudorandom function that is computed by two parties, one that holds the key to the function and learns nothing from

---

[7] This model assumes that user/password registration is implemented over secure channels but user authentication after registration does not assume public keys or secure channels for any party in the system - only the existence of public parameters, e.g., for defining an elliptic curve, is assumed. These parameters are common to all users of the system and are part of the client program run by a user; they require the same integrity guarantees as the program itself.

Setup

- *Group G.* The scheme works over a cyclic group G of prime order q, $|q| = \ell$, with generator g.
- *Hash functions H, H'* map arbitrary-length strings into elements of $\{0,1\}^\tau$ and G, respectively, where τ is a security parameter.
- *OPRF.* For a key $k \leftarrow Z_q$, we define function F_k as $F_k(x) = H(x, (H'(x))^k)$.
- *Parties.* User U, Device D, Server S.
- *Dictionary* Dict of size 2^d (a power of 2 is used for notational convenience only).
- Any PAKE protocol Π.

Initialization Phase *(assumed to be executed over secure links)*

- FK-PTR Initialization: U chooses password $\mathsf{pwd} \leftarrow \mathsf{Dict}$; D chooses and stores OPRF key $k \leftarrow Z_q$; U interacts with D to compute $\mathsf{rwd} = F_k(\mathsf{pwd})$.
- PAKE Initialization: User U and server S are initialized with value rwd used as a password according to the specification of PAKE protocol Π.

Login Phase

- **User-Device Interaction (FK-PTR)**
  1. U chooses $\rho \leftarrow Z_q$; sends $\alpha = (H'(\mathsf{pwd}))^\rho$ to D.
  2. D checks that the received $\alpha \in G$ and if so it responds with $\beta = \alpha^k$.
  3. U sets $\mathsf{rwd} = H(\mathsf{pwd}, \beta^{1/\rho})$.
- **User-Server Interaction (PAKE)**

  Follows the specification of the PAKE protocol Π where U uses rwd as its password.

Figure 2: The FK-PTR-PAKE Scheme

the computation, and one that holds an input and learns the output of the function on that input and nothing else.[8] In Figure 2 we present a particular instantiation of the PTR-PAKE protocol, which we call FK-PTR-PAKE, that results from a composition of FK-PTR, which is a specific instantiation of a PTR scheme, with a PAKE scheme. Figure 2 fully specifies the FK-PTR protocol, which is an interaction between U and D by which U retrieves a random value rwd with the help of its password pwd. At initialization, U chooses and remembers password pwd while D chooses and stores $k \leftarrow Z_q$. To retrieve rwd, U first blinds pwd by raising the hashed value $H'(\mathsf{pwd})$ to a random exponent ρ, and send it to D. This perfectly hides pwd from D and from any eavesdropper on the $\mathsf{U} - \mathsf{D}$ link. D checks that the received value is in the group G and if so it raises it to the secret exponent k. Now, U can de-blind this value by raising it to the power $1/\rho$ to obtain $H'(\mathsf{pwd})^k$. Finally, U hashes this value with pwd to obtain the randomized password rwd.

Note that D contains no information related to pwd hence an attacker interacting with D or even breaking into it learns nothing about pwd. Also, U does not run any test on the value reconstructed in the FK-PTR

---

[8] The reason that we define our notion of PTR instead of referring to some existing definition of OPRF is that that most existing definitions of OPRF are game-based, e.g. [17, 26], and their properties do not seem sufficient for the PTR protocol. On the other hand, the recent definition of UC OPRF [22] involves verifiability which we do not provide (hence the FK-PTR construction shown here uses fewer exponentiations than the OPRF scheme of [22] based on the same OMG-DH assumption in ROM), but even that UC OPRF notion does not imply some of the PTR properties we require, namely the 1-1 property listed as #4 in definition 3 below.

protocol. Hence, an attacker that interacts with U in the role of D does not learn anything about pwd from watching the behavior of U. These "obliviousness" and minimality properties of FK-PTR are essential to achieve PTR security and make the security analysis challenging. We will use this scheme to motivate the security requirements from a PTR scheme as needed for composing it with a PAKE protocol and obtain a secure DE-PAKE protocol. We establish these requirements in the next subsection and then prove the security of FK-PTR.

4.2 PTR Security Model

Here we present the security model for (generic) PTR schemes. We first define the adversarial game underlying this model and then use the FK-PTR scheme and explicit potential attacks against it to motivate the security definition.

PTR adversarial game. The game is parameterized by a function family F and a password dictionary Dict of size 2^d for some d (the power of two is chosen for notational convenience only). User U is initialized with password $\mathsf{pwd} \leftarrow \mathsf{Dict}$ and device D with a key k defining function F_k. Later, the parties interact so that in an undisturbed interaction between U and D, where U runs with input pwd, U outputs the secret $\mathsf{rwd} = F_k(\mathsf{pwd})$. Attacker A has oracle access to U and D, calling these parties with any message of its choice and receiving the corresponding response as defined by the scheme depending on the internal secrets and state of the responding party. The security requirements are defined below in Definition 3 but we first motivate them as follows.

Attack avenues and PTR security requirements. We define security of a PTR scheme in a way that guarantees that the generic composition of PTR and PAKE protocols results in a secure DE-PAKE scheme. The definition consists of several requirements that we motivate next via concrete attacks showing these requirements to be *necessary* (and by virtue of Thm. 3 also sufficient). Reducing the PTR requirements to the minimum necessary is pivotal for obtaining our *very* efficient FK-PTR implementation that would not be possible otherwise.

Attack avenue 1: Leakage on $\mathsf{rwd} = F_k(\mathsf{pwd})$. Given that A can obtain values in RDict by interacting with D on input any password in Dict we need to assure that nothing in the scheme leaks information on the specific value of $\mathsf{rwd} = F_k(\mathsf{pwd})$ or otherwise the attacker can use this information to gain advantage on guessing which of the RDict values is more plausible to be the correct rwd (e.g., it shouldn't be possible for A to test a possible value p as a candidate for pwd or to test a value r as a candidate for rwd). More generally, to apply PAKE we need to ensure that the view of the attacker at the end of the PTR run is independent, computationally or statistically, from rwd.

To capture this property we define the following experiment referred to as the *distinguishing test*. Define RDict as $\{F_k(p) : p \in \mathsf{Dict}\}$ where k is D's secret key. Let $\mathsf{rwd} = F_k(\mathsf{pwd})$ and choose $r \leftarrow \mathsf{RDict} \setminus \{\mathsf{rwd}\}$. A is given both rwd, r (in random order) and it needs to guess which one equals $F_k(\mathsf{pwd})$.

Attack avenue 2: Learning values in RDict. Since A can learn values in RDict by interacting with D, A can later interact with S in the PAKE protocol using these values. Thus, the best we can do is to require the PTR protocol not to leak to A more than one value in RDict for each interaction with D. We formalize this by defining a game where the attacker, at the end of its run, outputs a set of candidate values in RDict, and requiring that this set does not contain more than q_D correct values where q_D is the number of rogue activations of D by A.

Attack avenue 3: Using U *to test passwords.* Since the attacker can influence the values output by U in the PTR protocol, the possibility exists, at least in principle, that A makes U output a value $F_k(\mathsf{pwd}')$ where

$\mathsf{pwd}' \in \mathsf{Dict}$ is known to A. In this case, A can observe the PAKE run of U with S and see if pwd' is the correct password. This allows A to test passwords in Dict without having to act as an active MitM in the PAKE protocol between U and S. While this attack is not possible against FK-PTR (as we will prove later), one can show PTR schemes where this attack is feasible. There are two ways of dealing with this issue. We either show that any such "dictated password" requires a specific rogue activation of D (as in Attack 2 above) hence treating it as any other password in RDict that A may learn by interacting with D or we require that a secure PTR scheme does not allow for such attack. The latter is better as it prevents A from testing passwords without a rogue activation of U but the former can be acceptable in a protocol that allows the attack. Given that our FK-PTR protocol does not allow the attacker to use U as an oracle for testing passwords in $\mathsf{RDict} \setminus \{\mathsf{rwd}\}$, we choose the stronger notion by adding an explicit requirement against such possibility.

To prevent this we require that U's PTR output equals $F_k(\mathsf{pwd}')$ for $\mathsf{pwd}' \in \mathsf{Dict} \setminus \{\mathsf{pwd}\}$ with at most negligible probability.

Attack avenue 4: Running U *on passwords outside* RDict. The PTR-PAKE composition presents an attack avenue not present in regular PAKE protocols: A can make U run the PAKE protocol on a password from a dictionary RDict^* different than RDict (note that this is different from attack scenario 3). To see this, consider an attack in which A impersonates D to U running the protocol with a key k' chosen by A. As a consequence, U will run the PAKE protocol with the value $F_{k'}(\mathsf{pwd})$, i.e., with a value uniformly distributed over $\mathsf{RDict}^* = \{F_{k'}(p) : p \in \mathsf{Dict}\}$ where RDict^* is known to A. This allows A to attack the PAKE protocol as follows. It impersonates S to U as if the server's state was initialized with password $F_{k'}(p)$ for $p \in \mathsf{Dict}$. If $p = \mathsf{pwd}$, A succeeds in the impersonation and learns pwd.[9] This attack is not contemplated in standard PAKE models where the user is assumed to run with a password from the specified dictionary and without adversarial choice of the password. To illustrate the dangers of such attack, imagine that the family F has a key k^* such that $F_{k^*}(\cdot)$ is a constant function (with an output known to A). This is a real possibility against FK-PTR if we define $F_k(p)$ to be $H((H'(p)^k)$ in which case $k^* = 0$ has exactly this effect. Similarly, if there is a key k^* for which F_{k^*} is a t-to-1 function, A could discard t passwords with each S-impersonation attempt against U. Again, this is possible against FK-PTR with the modified F_k where A can choose β', the response returned to U, to be in a group of small-order. (Such an implementation of FK-PTR would require to test $\beta' \in G \setminus \{1\}$.)

To prevent this attack avenue we require that *any* attack strategy by A for generating a dictionary RDict^* induces a 1-1 function. We formalize this as follows. Let c denote a set of coins for parties U, D, A in a PTR run. For any such c define $f_c(p)$ as the output from U if its password was p. We require that except for negligible probability over the choice of c, f_c is 1-1. (Note that each such c defines a dictionary $\mathsf{RDict}^* = \{f_c(p) : p \in \mathsf{Dict}\}$ of size $|\mathsf{Dict}|$.)

We are now ready to define PTR security.

Definition 3. *We say that a PTR scheme is* (q_D, q_U, T, ϵ)-secure *if for any PTR attacker* A *that runs time* T *and performs* q_D *and* q_U *rogue activations of* D *and* U, *respectively,* ϵ *is an upper bound on the values* $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$ *defined as follows (these* ϵ_i *are functions of* q_D, q_U, T *and they correspond to the above attack avenues):*

---

[9] This attack recovers pwd with 2^d impersonation attempts (of S) against U and it only requires one value $F_{k'}(\mathsf{pwd})$ used by U as its PAKE password. This does not imply a break of the DE-PAKE scheme, since for each impersonation attempt against U, A needs to perform a rogue activation of U in PTR. If q'_U is the number of rogue activation of U in PTR and q_U is the number of rogue calls to U in PAKE, then the probability of successful impersonation is at most $\min\{q_U, q'_U\}/2^d$. This implies that *it is insecure for* U *to cache the value retrieved from* D *for use in multiple sessions* - doing so allows the above attack without A having to act as a MitM between U and D in each DE-PAKE session.

1. *the probability that* A *passes the distinguishing test of attack avenue 1 is at most* $1/2 + \epsilon_1$*;*
2. *the probability that* A *outputs more than* q_D *values in* RDict *following attack avenue 2 is at most* ϵ_2*;*
3. *the probability that* U *outputs* $F_k(\mathsf{pwd}')$ *for* $\mathsf{pwd}' \in \mathsf{Dict} \setminus \{\mathsf{pwd}\}$ *is at most* ϵ_3*;*
4. *the probability that* f_c*, as defined in attack avenue 4, is not 1-1 is less than* ϵ_4*.*

where in all four cases the probability goes over random PTR key k *and random* pwd *in* Dict*.*

4.3 Security of the FK-PTR Scheme

Theorem 1 below summarizes the security of the FK-PTR scheme in terms of Definition 3. It uses the One-More Gap Diffie-Hellman assumption defined next.

The One-More Gap DH (OMG-DH) Assumption [10, 27]: Let G be a group of prime order q and k a random value in Z_q. Let DH_k be an oracle[10] that on input $g \in G$ outputs g^k, and let DDH_k be an oracle that on input a pair (a, b) answers whether $b = a^k$. We say that G satisfies the ϵ_{omg}-OMG-DH assumption for function ϵ_{omg} if any attacker A that runs in time T has probability at most $\epsilon_{omg}(T, q_{dh}, q_{ddh})$ to win the following game: A is given access to the DH_k and DDH_k oracles, which it queries q_{dh} and q_{ddh} times, resp., and is given a set R of random elements in G. It wins if it outputs $q_{dh} + 1$ different pairs $(g, g^k), g \in R$.

Theorem 1. *Let* G *be a group where the* $\epsilon_{omg}(\cdot)$*-One-More Gap DH holds. Let the hash functions* H, H' *be modeled as random oracles and* q_H *be the number of invocations to* H*. Then, the FK-PTR scheme run over group* G *with a dictionary* $\mathsf{Dict} \subset \{0,1\}^\tau$ *is* (q_D, q_U, T, ϵ)-secure *where* $\epsilon = \max\{\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4\}$ *with* $\epsilon_1 = 0$*,* $\epsilon_2 \leq T/2^\tau + \epsilon_{omg}(T, q_D, q_H)$*,* $\epsilon_3 \leq 1/2^\tau$*,* $\epsilon_4 \leq |\mathsf{Dict}|^2/2^\tau$*.*

Proof. To show that $\epsilon_1 = 0$, note that the only information A sees related to the particular value pwd is $\alpha = (H'(\mathsf{pwd}))^\rho$ but since ρ is chosen by U uniformly in Z_q, α is uniformly distributed in G independently of pwd. Thus, the view of A is independent of U's password pwd and the probability of A to win the distinguishing test is $1/2$. The bound on ϵ_2 follows from Lemma 2 below proven based on the OMG-DH assumption. The bound on ϵ_3 follows from the fact that pwd is included under H, hence the probability that for some $\mathsf{pwd}' \in \mathsf{Dict} \setminus \{\mathsf{pwd}\}$, we have $H(\mathsf{pwd}', (H'(\mathsf{pwd}'))^k) = H(\mathsf{pwd}, (H'(\mathsf{pwd}))^k)$ is $2^{-\tau}$ where τ is the length of the output from H. Similarly, the bound on ϵ_4 follows from the collision resistance properties of the (random) H. ☐

Note: The bound $|\mathsf{Dict}|^2/2^\tau$ on ϵ_4 can be reduced significantly if one relaxes requirement 4 of PTR security to allow for some deviation from injectiveness, e.g., allowing RDict to be of size $\alpha \cdot |\mathsf{Dict}|$ for some α, say $\alpha = 1/2$.

Lemma 2. *Let* G *be a group where the* ϵ_{omg}*-One-More Gap DH assumption holds and model hash functions* H, H' *as random oracles. Let* A *be a PTR-attacker against the FK-PTR scheme that runs time* T *and activates* D q_D *times with values chosen by* A *(i.e., rogue activations). Then, the probability that* A *outputs more than* q_D *values in* RDict *(as in attack avenue 2) is at most* $\epsilon_2 = T/2^\tau + \epsilon_{omg}(T', q_D, q_H)$ *where* q_H *is the number of invocations of* H *by* A *and* $T' \approx T$*.*

Proof. Given a PTR attacker A against the FK-PTR scheme over a ϵ_{omg}-OMG-DH group G, we build an attacker A' against OMG-DH in group G. A' gets access to a DH_k and DDH_k oracles and an input in the form of an ordered set $R = g_1, ..., g_N$ of random elements in G. A' runs A on a simulated run of FK-PTR. A' uses DH_k to answer queries to D (i.e., A' simulates an instance of D under key k) and uses the set R to

---

[10] DH_k is not defined over elements outside G hence one needs to check the input to the oracle - it can be done by an explicit group membership check or by co-factor exponentiation.

answer H' queries. Namely, if $R = g_1, ..., g_N$ then the first H' query is answered with g_1, the second with g_2, etc. Queries (x, y) to H are answered with random values in $\{0,1\}^\tau$. A' keeps a table of defined inputs-outputs for these oracles and answers to repeated queries with the corresponding values in these tables. A' chooses pwd, sets $H'(\mathsf{pwd}) = g_1$ (i.e., this is done prior to answering any H' query from A), and queries g_1 to DH_k obtaining g_1^k which we denote by y^*.

A' simulates the actions of U faithfully, namely, it outputs messages of the form $\alpha = g_1^\rho$ for random $\rho \leftarrow Z_q$ and upon receiving a response β from A it computes $r' = H(\mathsf{pwd}, \beta^{1/\rho})$. When A delivers a message α' to D, A' responds to it as follows. If α' was output by U, in which case A' knows ρ such that $\alpha' = g_1^\rho$, A' responds with y_1^ρ; otherwise A' queries α' from DH_k. In addition, for inputs (x, y) to H, A' checks that $H'(x)$ was queried and if so it checks, using the DDH_k oracle, whether the $g_j \in R$ returned as the result of $H'(x)$ satisfies $y = g_j^k$. If all checks pass, A' stores the pair (g_j, y) in a list L.

Note that the above simulation of A is perfect hence the output from A is the same as in a real execution of FK-PTR. As claimed in the proof of Theorem 1, the view of A is independent of pwd and g_1 (A only sees values of the form g_1^ρ for one-time random ρ's) hence it is independent of rwd (which is computed as $H(\mathsf{pwd}, (H'(\mathsf{pwd}))^k)$).

Denote by R' the set of values in RDict output by A. Let E_1 denote the event that R' contains a value $r' = H(p, (H'(p))^k)$ for $p \in \mathsf{Dict}$ and that A did not query H on $(p, (H'(p))^k)$ or H' on p; the probability of E_1 is at most $T/2^\tau$. Assuming E_1 does not happen we have that if $r' = H(p, (H'(p))^k) \in R'$ then A queried H' on p and H on $(p, (H'(p))^k)$. Since A' chooses its responses to H' queries from elements in the set R, we have that $r' = H(p, {g_j}^k)$ for $g_j \in R$ so the query $H(p, (H'(p))^k)$ resulted in the pair $(g_j, {g_j}^k)$ being included by A' into the list L. Thus, $|R'| \leq |L|$.

Note that A' has obtained pairs (g_j, g_j^k) for all pairs in L as well as for (g_1, g_1^k). Let $L' = L \cup \{(g_1, g_1^k)\}$. By the DH-OMG assumption we have that the probability that L' contains more than q_{dh} elements is at most $\epsilon_{omg}(T', q_{dh}, q_{ddh})$ where T' is the running time of A'. So, except for this probability, we can assume $|L'| \leq q_{dh}$. We show $|L| \leq q_{\mathsf{D}}$ and therefore $|R'| \leq q_{\mathsf{D}}$ as claimed by the lemma.

A' queries DH_k in two cases: Upon rogue activations of D by A and for obtaining g_1^k. Thus, if $(g_1, g_1^k) \notin L$ we have $q_{dh} = q_{\mathsf{D}} + 1$, and together with the assumption $q_{dh} \geq |L'| = |L| + 1$ we obtain that $|L| \leq q_{dh} - 1 = q_{\mathsf{D}}$. If $(g_1, g_1^k) \in L$ (meaning that g_1 was also queried by A through a rogue activation) then $q_{dh} = q_{\mathsf{D}}$ and together with $q_{dh} \geq |L'| = |L|$ we obtain that $|L| \leq q_{dh} = q_{\mathsf{D}}$.

In all, we have that except for probability $\epsilon_2 = \epsilon_{omg}(T', q_D, q_H) + T/2^\tau$, $|R'| \leq |L| \leq q_D$. The running time of A' is essentially that of A if we count U activations as part of A running time and equate the cost of a call to DH_k to that of a D activation and the cost of a H' call to that of a DDH_k invocation. $\square$

4.4 PTR-PAKE Composition Theorem

We are now ready to prove the composition theorem showing that composing a secure PTR with a PAKE that offers the wKCI-resistance property, results in a secure DE-PAKE scheme (with security definitions as presented in Section 3). As noted earlier, if the PTR and PAKE schemes dispense of PKI so does our DE-PAKE protocol: An example of such composed scheme free of PKI (except for initialization) is presented in Section 5.

Theorem 3. *Let P be a $(q_D, q'_U, T_P, \epsilon_P)$-secure PTR scheme and Π be a $(q_S, q_U, T_\Pi, \epsilon_\Pi)$-secure PAKE protocol that is also ϵ_{KC}-wKCI-resistant, then the DE-PAKE scheme C that uses the composition of both protocols is a $(q_S, q_U, q'_U, q_D, T_C, \epsilon_C)$-secure DE-PAKE protocol where $\epsilon_C = \epsilon_\Pi + (3q'_U + 1)\epsilon_P + \epsilon_{KC} + \frac{q_U + q_S}{2^{\tau-1}}$.*

Proof. The proof of the Theorem is presented in Appendix A. $\square$

4.5 Adaptation to Standard Password-over-TLS

The most widely deployed password protocol in practice is the web-based "password-over-TLS" scheme, namely, one where server and browser establish a TLS channel authenticated by the server over which the user's password is encrypted and transmitted to the server. At reception the server decrypts the password and passes it to the application that requested the authentication. Typically, such application will not keep passwords in plaintext form but will rather store a (possibly salted) hash image of the password so that upon server compromise an attacker still has to mount a dictionary attack. However, in this setting an attacker that compromises or controls a server can potentially learn the *cleartext password* upon user authentication, namely, when the password is decrypted and sent to the application.

This capability of the attacker can be exploited in our setting as follows. Assume the attacker A corrupts server S and impersonates the device to the user choosing its own OPRF key k'. When the user U authenticates to S, it obtains from the (impersonated) device the value $\mathsf{rwd}' = f_{k'}(\mathsf{pwd})$ which U then transmits to S over TLS. If we assume that A learns the cleartext rwd' by corrupting S, then A can now mount a dictionary attack on pwd on the basis of the values k' and rwd' that it knows. Therefore, to recover DE-PAKE security in this case one has to assume that the communication from the device D to the user's client machine is authenticated. This requires a one-time setup procedure where a device's key is installed in the user's machine (one can also leverage the PKI setting in this case) .

Revisiting the security analysis in this case, one notes that the ability to learn the plaintext rwd', violates PTR security, particularly via *attack avenue 4*. Indeed, the attacker A can now induce the computation of a password rwd' outside RDict which it learns via S compromise. Requiring the communication from the device to the client machine to be authenticated solves the issue as it prevents the attacker from choosing its own OPRF key k' for computing non-RDict passwords. Thus, with device authentication the proof of Theorem 3 remains valid. It is worth noting that in this setting the attacker A can learn $\mathsf{rwd} = f_k(\mathsf{pwd})$ computed under the real device's key k. Yet, this is of no value to A as it does not allow to mount an attack on pwd or on other rwd values except if A also compromises the device. In the latter case, A can mount an offline dictionary attack but as stated before such an attack upon compromise of both device and server is unavoidable.

5 Instantiations and Extensions

In this section, we discuss several instantiations and extensions of our PTR-PAKE scheme showing the practicality and flexibility of our approach. We first present a full and detailed instantiation of PTR-PAKE that is secure in the PKI-free setting . Then, we show how to provide transparent DE-PAKE support to currently deployed web services, namely, armoring an existing service against online and offline attacks *without changing the server*. Finally, we comment on extensions that provide defenses against client-side and phishing attacks.

PKI-Free DE-PAKE. Figure 3 describes a full instantiation of a PKI-free PTR-PAKE protocol using the FK-PTR scheme from Figure 2 and a PKI-free PAKE protocol with resistance against wKCI attacks adapted from the threshold PAKE (TPAKE) protocol of [22, 23]. More precisely, the PAKE protocol we use is an adaptation of the variant of the TPAKE protocol from [22, 23] with resistance against wKCI attacks, proven secure in the PKI-free (or CRS) model, to the single-server case (i.e., a (1,1)-TPAKE).

The protocol as described in Figure 3 also requires a key-exchange mechanism (the "KE formula") to set a session key between server and user (in particular for the sake of mutual authentication). Different protocols can be used here, for example, based on shared keys or public keys, with or without forward

Parties: User U, Device D, Server S.

Public Parameters and Components

- *Group* G of prime order q with generator g.
- *Hash functions* H, H' with ranges $\{0,1\}^{2\tau}, G$ and Z_q, respectively, for τ a security parameter.
- *Pseudorandom function (PRF)* f with range $\{0,1\}^{2\tau}$.
- *OPRF function* $F_k(x) = H(x, (H'(x))^k)$ for key $k \in Z_q$.
- *Key exchange formula* KE: on input long-term and ephemeral private-public keys outputs shared key $K \in \{0,1\}^{\tau}$.

Initialization Phase *(assumed to be executed over secure links)*

- FK-PTR Initialization: Run FK-PTR initialization of Fig. 2 to choose pwd, device's OPRF key k_d, and compute $\mathsf{rwd} = F_{k_d}(\mathsf{pwd})$.
- PAKE Initialization:
  1. S chooses $p_s \in_{\mathrm{R}} Z_q$ and sends to U the public key $P_s = g^{p_s}$ (P_s can be used with all of S's users).
  2. U chooses $z \in_{\mathrm{R}} \{0,1\}^{\tau}, k_s \in_{\mathrm{R}} Z_q$;
     sets values $c = z \oplus F_{k_s}(\mathsf{rwd}),\ r = f_z(0),\ C = H(r, \mathsf{rwd}, c),\ p_u = f_z(1) \bmod q$;
     computes $P_u = g^{p_u}$ and $m_u = f_z(2, P_u, P_s)$; and sends to S the values $c,\ C,\ k_s,\ P_u,\ m_u$.
  3. S stores $c,\ C,\ k_s, P_u, m_u$ in its U-associated storage (if P_s is user-specific, it also stores P_s and p_s).

Login Phase

- **User-Device Interaction (FK-PTR)**

  Follows the FK-PTR protocol as per Fig. 2 to obtain rwd on input pwd from U and input k_d from D.
- **User-Server Interaction (PAKE)**
  1. U chooses $\rho, x_u \leftarrow Z_q$; initiates a key exchange session with S by sending its identity U, the value $\alpha = (H'(\mathsf{rwd}))^{\rho}$ and $X_u = g^{x_u}$.
  2. S proceeds as follows:
     (a) Checks that $\alpha \in G$;
     (b) Retrieves (c, C, k_s, P_u, m_u) from its U-associated storage;
     (c) Picks $x_s \in_{\mathrm{R}} Z_q$ and computes $\beta = \alpha^{k_s},\ X_s = g^{x_s}$.
     (d) Sends to U: $\beta, c, C, P_u, m_u, P_s, X_s$.
     (e) Computes $K = \mathsf{KE}(p_s, x_s, P_u, X_u)$ and outputs session key $SK = f_K(0)$.
  3. U proceeds as follows:
     (a) Sets $z = c \oplus H(\mathsf{rwd}, \beta^{1/\rho}),\ r = f_z(0),\ p_u = f_z(1) \bmod q$.
     (b) Aborts unless the following conditions hold: $\beta \in G,\ C = H(r, \mathsf{rwd}, c),\ m_u = f_z(2, P_u, P_s)$.
     (c) Computes $K = \mathsf{KE}(p_u, x_u, P_s, X_s)$ and outputs session key $SK = f_K(0)$.
- **Explicit Authentication**

  If explicit authentication of the parties is required then S adds the value $f_K(1)$ to its message and U adds a third message with value $f_K(2)$. Each party verifies the value received from the other party.

Figure 3: Instantiation of FK-PTR-PAKE with PKI-free PAKE protocol from [22, 23]

secrecy, etc. However, we note that in order to achieve security against server compromise (needed to provide the maximal security of a PTR-PAKE scheme) one must use a public key mechanism. Otherwise, the server would be storing a secret authentication key for the user which would allow an attacker to impersonate the user to the server in case of server compromise. Thus, while we allow for different key exchange mechanisms through a general KE formula, we do require these to be based on public keys for both parties (we also accommodate ephemeral keys if forward secrecy is desired). For illustration, and as a concrete and efficient instantiation that preserves a minimal number of messages and provides forward secrecy, we define the key computation formulas corresponding to the HMQV protocol [31], where $e_u = H(X_u, \mathsf{S}), e_s = H(X_s, \mathsf{U})$:

$$\begin{aligned} \text{For } \mathsf{S}: \quad & K = \mathsf{KE}(p_s, x_s, P_u, X_u) = H\left((X_u P_u^{e_u})^{x_s + e_s p_s}\right) \\ \text{For } \mathsf{U}: \quad & K = \mathsf{KE}(p_u, x_u, P_S, X_S) = H\left((X_s P_s^{e_s})^{x_u + e_u p_u}\right) \end{aligned}$$

Server-Transparent DE-PAKE. An important implication of the modularity of our PTR-PAKE scheme is that the user can use any secure PTR protocol to derive a hardened password rwd from her nominal password pwd, and then register rwd as her actual password with an existing server, where the latter implements *any* wKCI-resistant password authentication protocol acting as PAKE. In particular, such authentication protocol can be the standard PKI-based password-over-TLS protocol widely used as the basis for web authentication. In such case the login phase of the PTR-PAKE protocol consists of the user typing her password pwd, the client terminal and the device executing the PTR protocol to compute the hardened password rwd, and the client terminal sending rwd to the server over a TLS session. In this setting, no modification to an existing service is required. We note, however, that as stated in Section 4.5, providing DE-PAKE security in this TLS setting requires authentication of the communication from device to the user's machine. Once this authentication is ensured (via a setup procedure where a device's key is installed in the user's machine) the resultant composition of PTR and password-over-TLS is DE-PAKE secure.

There are several advantages of this setting: (1) the user can simply remember the short nominal pwd but register with a strong high-entropy password that significantly increases resistance to online and offline guessing attacks (in particular, offline-only attacks on a compromised server are not possible); (2) nominal password pwd can be the same or reused among multiple services, but the OPRF key associated with each service stored on the device can be different (hence also rwd would be different), and therefore the compromise of the password rwd at one server will not reveal the actual password pwd and will not compromise the user's accounts with other services; (3) rather than asking the user to frequently change the password and memorize the updated password, only the key on the device can be changed, which improves the usability.

In a companion work [34], we have developed a smartphone-based password manager application built on top of the DE-PAKE framework and investigated its usability. The results of our study demonstrate the promising feasibility of our approach to a real-world practical problem.

Resisting Client-Side & Phishing Attacks. Malicious code and keyloggers remain a threat to browsers in spite of browser security enhancements. Because we use a keyed password hardening scheme, an attacker who learns pwd by a key-logger or shoulder surfing can *not* authenticate to the service without interacting with the device. However, an attacker who compromises a client terminal can obtain rwd. By using service-specific keys at the device we guarantee that an attacker who obtains rwd can only compromise the particular service associated with it; even if pwd is used for multiple services, the rwd values derived for each service are random and independent .

Still, one can reduce the threat of the malware attack to the by combining our scheme with the traditional two-factor authentication (TFA) mechanism, i.e., having D generate a PRF on a time value or a nonce under a key that D shares with S. Note that in a traditional TFA mechanism, compromising the client allows the attacker to hijack the current login session of the user, but does not allow the attacker to login in future sessions (due to the use of "one-time" PIN codes). Integrating our DE-PAKE protocol with traditional TFA could provide the same level of security in the event of client compromise, while providing all the other security properties of our DE-PAKE scheme.

Resistance to phishing attacks can be achieved if rwd is computed on a concatenatation of pwd and the URL being accessed, i.e., if $\mathsf{rwd} = F_{K_d}(\mathsf{pwd}|url)$. This is similar to the PwdHash approach [33] except that in PwdHash, the attacker that obtains the randomized password through phishing can mount a dictionary attack to find the user's password while in our case this is not feasible.

Integrating our DE-PAKE techniques with existing two-factor authentication mechanisms (e.g., PIN based) to simultaneously enhance security against offline-online attacks and client compromise is left as a future work item.

6 Conclusions and Future Work

In this paper, we considered the problem of armoring password protocols against online guessing attacks as well as offline dictionary attacks in the event of server or device compromise. We proposed a novel, efficient and modular device-enhanced password protocol (DE-PAKE) and formally analyzed its security. In contrast to previous work on this subject, our protocol does not require the presence of a public key infrastructure or the availability of authenticated public keys (except, possibly, for initial password registration) thus relaxing the concerns regarding PKI failures or compromises. At the same time, when an authentic and uncompromised public key of the server is available, our protocol further guarantees resilience to server impersonation even when the user's password is disclosed. Remarkably, we can achieve these benefits without necessitating service-side changes.

Finally, we note that, thanks to our modular architecture, one can further increase the resistance to server compromise by using a threshold-PAKE protocol (e.g., [22]), in which case an attacker needs to compromise a threshold of servers *in addition to* the device before being able to mount an offline dictionary attack.

Acknowledgments

This research is partially supported by ONR Contract N00014-14-C-0113, NSF CICI Award #1547435 and #1547350, and NSF CNS Award #1209280.

References

[1] Anonymous hackers claim to leak 28,000 PayPal passwords on global protest day. Available at: `http://goo.gl/oPv2h`.

[2] Blizzard servers hacked; emails, hashed passwords stolen. Available at: `http://goo.gl/OTNWJC`.

[3] Hackers compromised nearly 5M Gmail passwords. Available at: `http://goo.gl/IRu07u`.

[4] LinkedIn Confirms Account Passwords Hacked. Available at: `http://goo.gl/UBWuY0`.

[5] RSA breach leaks data for hacking securid tokens. Available at: `http://goo.gl/tcEoS`.

[6] RSA SecurID software token cloning: a new how-to. Available at: `http://goo.gl/qkSFY`.

[7] Russian Hackers Amass Over a Billion Internet Passwords. Available at: http://goo.gl/aXzqj8.

[8] T. Acar, M. Belenkiy, and A. Kupcu. Single password authentication. *Computer Networks*, 57(13):2597 – 2614, 2013.

[9] A. Bagherzandi, S. Jarecki, N. Saxena, and Y. Lu. Password-protected secret sharing. In *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, pages 433–444, 2011.

[10] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. 16(3):185–215, June 2003.

[11] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology – Eurocrypt*, 2000.

[12] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In *Proceedings of the 6th IMA International Conference on Cryptography and Coding*, pages 30–45, London, UK, UK, 1997. Springer-Verlag.

[13] X. Boyen. Hidden credential retrieval from a reusable password. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, pages 228–238, New York, NY, USA, 2009. ACM.

[14] J. Brainard, A. Juels, B. Kaliski, and M. Szydlo. A new two-server approach for authentication with short secrets. In *Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12*, SSYM'03, pages 14–14, Berkeley, CA, USA, 2003. USENIX Association.

[15] A. Everspaugh, R. Chaterjee, S. Scott, A. Juels, and T. Ristenpart. The pythia prf service. In *24th USENIX Security Symposium (USENIX Security 15)*, 2015.

[16] W. Ford and B. S. Kaliski Jr. Server-assisted generation of a strong secret from a password. In *Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2000.

[17] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography*. 2005.

[18] C. Gentry, P. MacKenzie, and Z. Ramzan. A method for making password-based key exchange resilient to server compromise. In *Advances in Cryptology-CRYPTO*. 2006.

[19] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. 2(3):230–268, Aug. 1999.

[20] D. P. Jablon. Password authentication using multiple servers. In *The Cryptographer's Track at RSA Conference (CT-RSA)*. 2001.

[21] S. Jarecki, A. Kiayias, and H. Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, pages 233–253, 2014.

[22] S. Jarecki, A. Kiayias, and H. Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In *Advances in Cryptology–ASIACRYPT*. 2014.

[23] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu. Highly Efficient and Composable Password-Protected Secret Sharing. In *1st IEEE European Symposium on Security and Privacy (EuroS&P)*. 2015.

[24] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu. Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online). In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 276–291, 2016.

[25] S. Jarecki, H. Krawczyk, M. Shirvanian, and N. Saxena. Device-enhanced password protocols with optimal online-offline protection. IACR Cryptology ePrint Archive: Report 2015/1099 available at `http://eprint.iacr.org/2015/1099`, December 2015.

[26] S. Jarecki and X. Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In *Theory of Cryptography*. 2009.

[27] S. Jarecki and X. Liu. Fast secure computation of set intersection. In *Security and Cryptography for Networks*. 2010.

[28] J. Katz, P. MacKenzie, G. Taban, and V. Gligor. Two-server password-only authenticated key exchange. *Journal of Computer and System Sciences*, 78(2):651 – 669, 2012. Games in Verification.

[29] F. Kiefer and M. Manulis. *Distributed Smooth Projective Hashing and Its Application to Two-Server Password Authenticated Key Exchange*, pages 199–216. Springer International Publishing, Cham, 2014.

[30] F. Kiefer and M. Manulis. *Universally Composable Two-Server PAKE*, pages 147–166. Springer International Publishing, Cham, 2016.

[31] H. Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In *Advances in Cryptology–CRYPTO*, 2005.

[32] P. D. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold password-authenticated key exchange. In *Advances in Cryptology - CRYPTO 2002, International Cryptology Conference*, 2002.

[33] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *Usenix security Symposium*, 2005.

[34] M. Shirvanian, S. Jarecki, H. Krawczyk, and N. Saxena. SPHINX: A password store that perfectly hides passwords from itself. In *37th IEEE International Conference on Distributed Computing Systems (ICDCS 2017), to appear*, June 2017.

[35] M. Shirvanian, S. Jarecki, N. Saxena, and N. Nathan. Two-factor authentication resilient to server compromise using mix-bandwidth devices. In *Network & Distributed System Security Symposium*, 2014.

[36] M. Szydlo and B. Kaliski. Proofs for two-server password authentication. In A. Menezes, editor, *Topics in Cryptology – CT-RSA 2005: The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005. Proceedings*, pages 227–244, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

A Proofs

A.1 Proof of Lemma 2

Proof. Given a PTR attacker A against the FK-PTR scheme over a ϵ_{omg}-OMG-DH group G, we build an attacker A' against OMG-DH in group G. A' gets access to a DH_k and DDH_k oracles and an input in the form of an ordered set $R = g_1, ..., g_N$ of random elements in G. A' runs A on a simulated run of FK-PTR. A' uses DH_k to answer queries to D (i.e., A' simulates an instance of D under key k) and uses the set R to answer H' queries. Namely, if $R = g_1, ..., g_N$ then the first H' query is answered with g_1, the second with g_2, etc. Queries (x, y) to H are answered with random values in $\{0, 1\}^\tau$. A' keeps a table of defined inputs-outputs for these oracles and answers to repeated queries with the corresponding values in these tables. A' chooses pwd, sets $H'(\mathsf{pwd}) = g_1$ (i.e., this is done prior to answering any H' query from A), and queries g_1 to DH_k obtaining g_1^k which we denote by y^*.

A' simulates the actions of U faithfully, namely, it outputs messages of the form $\alpha = g_1^\rho$ for random $\rho \leftarrow Z_q$ and upon receiving a response β from A it computes $r' = H(\mathsf{pwd}, \beta^{1/\rho})$. When A delivers a message α' to D, A' responds to it as follows. If α' was output by U, in which case A' knows ρ such that $\alpha' = g_1^\rho$, A' responds with y_1^ρ; otherwise A' queries α' from DH_k. In addition, for inputs (x, y) to H, A' checks that $H'(x)$ was queried and if so it checks, using the DDH_k oracle, whether the $g_j \in R$ returned as the result of $H'(x)$ satisfies $y = g_j^k$. If all checks pass, A' stores the pair (g_j, y) in a list L.

Note that the above simulation of A is perfect hence the output from A is the same as in a real execution of FK-PTR. As claimed in the proof of Theorem 1, the view of A is independent of pwd and g_1 (A only sees values of the form g_1^ρ for one-time random ρ's) hence it is independent of rwd (which is computed as $H(\mathsf{pwd}, (H'(\mathsf{pwd}))^k)$).

Denote by R' the set of values in RDict output by A. Let E_1 denote the event that R' contains a value $r' = H(p, (H'(p))^k)$ for $p \in \mathsf{Dict}$ and that A did not query H on $(p, (H'(p))^k)$ or H' on p; the probability of E_1 is at most $T/2^\tau$. Assuming E_1 does not happen we have that if $r' = H(p, (H'(p))^k) \in R'$ then A queried H' on p and H on $(p, (H'(p))^k)$. Since A' chooses its responses to H' queries from elements in the set R, we have that $r' = H(p, {g_j}^k)$ for $g_j \in R$ so the query $H(p, (H'(p))^k)$ resulted in the pair $(g_j, {g_j}^k)$ being included by A' into the list L. Thus, $|R'| \le |L|$.

Note that A' has obtained pairs (g_j, g_j^k) for all pairs in L as well as for (g_1, g_1^k). Let $L' = L \cup \{(g_1, g_1^k)\}$. By the DH-OMG assumption we have that the probability that L' contains more than q_{dh} elements is at most $\epsilon_{omg}(T', q_{dh}, q_{ddh})$ where T' is the running time of A'. So, except for this probability, we can assume $|L'| \le q_{dh}$. We show $|L| \le q_\mathsf{D}$ and therefore $|R'| \le q_\mathsf{D}$ as claimed by the lemma.

A' queries DH_k in two cases: Upon rogue activations of D by A and for obtaining g_1^k. Thus, if $(g_1, g_1^k) \notin L$ we have $q_{dh} = q_\mathsf{D} + 1$, and together with the assumption $q_{dh} \ge |L'| = |L| + 1$ we obtain that $|L| \le q_{dh} - 1 = q_\mathsf{D}$. If $(g_1, g_1^k) \in L$ (meaning that g_1 was also queried by A through a rogue activation) then $q_{dh} = q_\mathsf{D}$ and together with $q_{dh} \ge |L'| = |L|$ we obtain that $|L| \le q_{dh} = q_\mathsf{D}$.

In all, we have that except for probability $\epsilon_2 = \epsilon_{omg}(T', q_D, q_H) + T/2^\tau$, $|R'| \le |L| \le q_D$. The running time of A' is essentially that of A if we count U activations as part of A running time and equate the cost of a call to DH_k to that of a D activation and the cost of a H' call to that of a DDH_k invocation. ☐

A.2 Lemma 4

We formulate and prove Lemma 4 that we use as a main component in the proof of the composition theorem, Theorem 3.

Lemma 4. *Let* Π *be a* (q_S, q_U, T, ϵ)*-secure PAKE protocol. Then the following holds for any PAKE-adversary* A *against* Π*. Let* $\mathsf{Dict} \subset \{0,1\}^\tau$ *be a dictionary composed of the union of two disjoints sets* Dict_1 *and* Dict_2*, where* Dict_1 *is known to* A *and* Dict_2 *is chosen as a random subset of* $\{0,1\}^\tau \setminus \mathsf{Dict}_1$ *and unknown to* A*. Then the advantage of* A *against* Π *running with dictionary* Dict *is at most*

$$\frac{\min\{q_S + q_U, |\mathsf{Dict}_1|\}}{|\mathsf{Dict}|} + \frac{q_S + q_U}{2^{\tau-1}} + \epsilon.$$

Proof. Denote $q_A = q_S + q_U$. The winning probability of A when the password is selected at random in D satisfies (see below for explanations):

$$\begin{aligned}
&Pr\left[\mathsf{A\ wins} : \mathsf{p} \in \mathsf{Dict}_1\right] \cdot Pr\left[\mathsf{p} \in \mathsf{Dict}_1\right] + \\
&\qquad Pr\left[\mathsf{A\ wins} : \mathsf{p} \in \mathsf{Dict}_2\right] \cdot Pr\left[\mathsf{p} \in \mathsf{Dict}_2\right] \\
\le\;& \frac{1}{2} + \frac{\min\{q_A, |\mathsf{Dict}_1|\}}{|\mathsf{Dict}_1|}\frac{|\mathsf{Dict}_1|}{|\mathsf{Dict}|} + \frac{q_A}{|\overline{\mathsf{Dict}_1}|}\frac{|\mathsf{Dict}_2|}{|\mathsf{Dict}|} + \epsilon \\
\le\;& \frac{1}{2} + \frac{\min\{q_A, |\mathsf{Dict}_1|\}}{|\mathsf{Dict}|} + \frac{2q_A}{2^\tau} + \epsilon
\end{aligned}$$

as claimed in the lemma.

To see why the above inequalities hold, first note that the case $\mathsf{p} \in \mathsf{Dict}_1$ corresponds to a regular PAKE game with known dictionary Dict_1 hence the attacker's winning probability in this case is at most $1/2 + \min\{q_A, |\mathsf{Dict}_1|\}/|\mathsf{Dict}_1| + \epsilon$; on the other hand, $Pr\left[\mathsf{p} \in \mathsf{Dict}_1\right] = |\mathsf{Dict}_1|/|\mathsf{Dict}|$ from which the first term in the final expression follows (ϵ and $1/2$ are separated as they are common to both terms). Note that the minimum in $\min\{q_A, |\mathsf{Dict}_1|\}$ simply means that the advantage $q_A/|\mathsf{Dict}_1|$ is capped at 1 even for larger values of q_A. As for the second term, the case $\mathsf{p} \in \mathsf{Dict}_2$ is, from the point of view of A, equivalent to a dictionary $\overline{\mathsf{Dict}_1} = 2^\tau \setminus \mathsf{Dict}_1$ since all elements outside Dict_1 are equiprobable. Hence, the winning probability in this case is at most $\frac{1}{2} + \frac{q_A}{|\overline{\mathsf{Dict}_1}|} + \epsilon$ while $Pr\left[\mathsf{p} \in \mathsf{Dict}_2\right] = |\mathsf{Dict}_2|/|\mathsf{Dict}|$, resulting in the second term. Finally, we observe that when $|\mathsf{Dict}_1| \le 2^{\tau-1}$, then $\frac{q_A}{|\overline{\mathsf{Dict}_1}|}\frac{|\mathsf{Dict}_2|}{|\mathsf{Dict}|} \le \frac{q_A}{|\overline{\mathsf{Dict}_1}|} \le \frac{q_A}{2^{\tau-1}}$, and when $|\mathsf{Dict}_1| \ge 2^{\tau-1}$ then $\frac{q_A}{|\overline{\mathsf{Dict}_1}|}\frac{|\mathsf{Dict}_2|}{|\mathsf{Dict}|} \le \frac{q_A}{|\mathsf{Dict}|} \le \frac{q_A}{|\mathsf{Dict}_1|} \le \frac{q_A}{2^{\tau-1}}$. ☐

A.3 Proof of Theorem 3

Proof. We consider 4 cases as in Definition 2 according to whether D and S are corrupted or not. We focus on the main ideas of the proof - a formal presentation would represent the arguments in the proof below as a sequence of game transitions.

Case 1: No corruption. We start by addressing the following modeling issue. In the CRS setting, a PTR-PAKE attacker A can make the user U run with a password generated via a function f_c applied to the U's password pwd as in attack avenue 4. Since A has no information about pwd (first PTR requirement), this is equivalent to U choosing a random independent password rwd^* from the dictionary $\mathsf{RDict}^* = f_c(\mathsf{Dict})$ (which by the 1-1 requirement is of the same size as user U's dictionary Dict). Also, since by the third requirement of PTR security, rwd^* is different than U's real password $\mathsf{rwd} = F_k(\mathsf{pwd})$, then the runs of U with rwd^* are independent from those with rwd (runs of a user with different passwords are independent of each other since the only shared state between runs, or sessions, is the password). Thus, we can treat U running with rwd^* as a separate user from U, one created by the attacker with dictionary $\mathsf{RDict}^* = f_c(\mathsf{Dict})$. Note that U does not have rwd^* registered with S or any other server (the attacker is allowed to create such unregistered users). We will refer to these derived users as "split users" and consider them as additional regular users in a PAKE protocol.[11]

In summary, thanks to requirements 1, 3, 4 of PTR security, the ability of the PTR attacker to induce different password outputs from U translates into the ability of the PAKE adversary to create independent "split users". Note that user U can run with different dictionaries RDict^*, corresponding to different functions f_c, and each such run generates a new split user. On the other hand, the PTR attacker may choose to use the same f_c multiple times which we model as repeated runs of the same split user (since in all these runs the user will use the same password rwd^*). Thus, in what follows, we assume a setting (or game) where each split user runs with a password pwd^* that is independent of pwd and independent of other users' pwd^*, and that these passwords are chosen from dictionaries RDict^* of size 2^d. Formally, we need to apply a standard sequence-of-games argument to quantify the increase in the advantage of the attacker in this game transition. Specifically, each split user activation adds a $\epsilon_1 + \epsilon_3 + \epsilon_4$ advantage to the attacker success for a total of $q'_U(\epsilon_1 + \epsilon_3 + \epsilon_4)$.

For clarity, we will use Π^* to denote the PAKE protocol Π when run against an attacker that can create split users with independent passwords as above. The PAKE security of Π implies the PAKE security of Π^* (the PAKE model requires Π to be secure with any number of adversarially generated users).

Having established this correspondence between PTR-induced passwords and split users we can now reduce the DE-PAKE security of a PTR-PAKE scheme to the PAKE security of Π. That is, we build a PAKE attacker SIM against Π given a DE-PAKE attacker A against the composed DE-PAKE scheme C. For this SIM simulates the PTR part of the protocol as follows. Let Dict be the dictionary used by the target user U. SIM chooses k for D and $\mathsf{pwd} \leftarrow \mathsf{Dict}$ for U in the PTR game. It defines the dictionary on which Π^* runs as $\mathsf{RDict} = \{F_k(p) : p \in \mathsf{Dict}\}$. By the 1-1 property of F_k, RDict is of the same size as Dict (note that Π needs to be secure against any dictionary, even an adversarially chosen one).

This simulation of P is perfect as it uses full information on the parties' secrets (pwd and k). Then, by virtue of PTR security (requirement 1), we have that the view of the DE-PAKE attacker A is independent, up to an advantage loss of ϵ_1, of pwd and of the password $\mathsf{rwd} = F_k(\mathsf{pwd})$ used by U in protocol Π.

Now consider the PAKE activations by the DE-PAKE attacker A of the target pair (U, S), i.e., the activations of S as well as of U running with password rwd and of U running with passwords induced by A in the PTR activation of U. We start by considering the activations of S and U according to the regular PAKE model and then consider the activations related to split users.

The attacker has partial knowledge of the dictionary RDict from which U's password rwd is chosen. Specifically, by requirement 2 of PTR security, we can assume (up to an advantage difference of ϵ_2), that A knows at most q_D elements in RDict, where q_D is the number of rogue activations of D by A. In the view of A, the rest of RDict is distributed uniformly (or pseudorandomly) in $\{0,1\}^\tau$. Thus, we are in the setting of Lemma 4, hence the probability

---

[11] The only difference with traditional users is that they are not registered with any server, although we could define a special server S with which the attacker register split users but S is never activated by the attacker.

of A winning the DE-PAKE game in a session at U or S is at most

$$\frac{\min\{q_U + q_S, q_D\}}{|\mathsf{Dict}|} + \frac{q_U + q_S}{2^{\tau-1}} + \epsilon_\Pi. \tag{2}$$

We now consider Π activations of U running with a password from an attacker-induced dictionary RDict^*, or the equivalent Π^*-activation by A of a split user U^*. Such user runs with a password rwd^* from a dictionary RDict^* of the same size as Dict and where $\mathsf{rwd}^* \neq \mathsf{rwd}$; in particular, rwd^* is not registered with S. Thus, activations of S are irrelevant to this case but A may activate U^* (with rogue send messages purportedly coming from S) in order to attempt at winning a test session at U^*. Since this attack is a legitimate attack against the PAKE protocol Π^*, we have that its success is at most $q_U^*/|\mathsf{RDict}^*| = q_U^*/|\mathsf{Dict}|$ where q_U^* is the number of activations of U^*. The sum of all activations of all split users U^* derived from U is bounded by the number of activations in Π of user U thus the total success probability of A against split users (i.e., against U running on an induced password pwd^*) is bounded by $q_U/|\mathsf{Dict}|$.

However, note that *each activation* of U in C with an induced password other than rwd (equivalently, the activation of a split user U^* in Π^*) requires a rogue activation of U by A in the PTR protocol. Thus, if we denote by q'_U the number of rogue U activations in the PTR protocol, we need to adjust the above bound to $\min\{q_U, q'_U\}/|\mathsf{Dict}|$ (i.e., this form of attack can be exploited only if the activation of U as a Π user is matched by a rogue activation of U as a PTR user).

The final bound on A's advantage is obtained by adding together the above term $\min\{q_U, q'_U\}/|\mathsf{Dict}|$ and the one in (2). Before doing so we note that the value q_U in (2) only counts rogue activations of U running on the correct U's password rwd while the q_U in $\min\{q_U, q'_U\}/|\mathsf{Dict}|$ counts rogue activations running on an unregistered password rwd^*. If we denote the number of the first type of activations by p_U and the latter type by p_U^*, we have that the total advantage of the attacker is

$$\frac{\min\{p_U + q_S, q_D\}}{|\mathsf{Dict}|} + \frac{\min\{p_U^*, q'_U\}}{|\mathsf{Dict}|} + \frac{p_U + q_S}{2^{\tau-1}} + \epsilon_\Pi \tag{3}$$

$$\leq \quad \frac{\min\{p_U + q_S + p_U^*, q_D + q'_U\}}{|\mathsf{Dict}|} + \frac{p_U + q_S}{2^{\tau-1}} + \epsilon_\Pi. \tag{4}$$

Noting that $q_U = p_U + p_U^*$ and adding to the above expression the attacker's advantage from PTR game transitions ($\epsilon_2 + q'_U(\epsilon_1 + \epsilon_3 + \epsilon_4)$), we get that the total advantage of the DE-PAKE attacker A is bounded by

$$\frac{\min\{q_U + q_S, q_D + q'_U\}}{|\mathsf{Dict}|} + \epsilon'_C$$

where $\epsilon'_C = \frac{p_U + q_S}{2^{\tau-1}} + \epsilon_\Pi + (3q'_U + 1)\epsilon_P$.

We now consider the cases where server or device are corrupted. In all these cases the above analysis of case 1 holds except that some of the online operations can now be performed offline.

Case 2: D *corrupted.* In this case the attacker learns k, hence it does not need to access D via online activations. By the same argument in Case 1 based on Lemma 4, we have that if A computes q_D values from the dictionary RDict (by offline computation using k), its advantage in the DE-PAKE game where it activates S and U for q_S and q_U times, respectively, is bounded by equation (2). If, in addition, A attacks U in PTR with q'_U queries, q_D in (2) becomes $q_D + q'_U$. But in any case, given the $\min$ in (2), $\mathcal{A}$'s advantage (even with $|\mathsf{Dict}|$ offline F_k computations) is at most $(q_U + q_S)/|\mathsf{Dict}| + \epsilon'_C$.

Case 3: S *corrupted (or* U*-compromised).* Consider first attacks that do not involve rogue queries to U. In this case, by virtue of the PAKE protocol Π being ϵ_{KC}-wKCI-resistant, we have that an attacker against Π that knows q passwords from the dictionary RDict has advantage at most $\min\{q_S, q\}/|\mathsf{Dict}| + \epsilon_{KC}$, where q_S counts offline operations based on S's state. On the other hand, as in the argument of case 1, by requirement 2 of PTR security, we can assume that A knows at most q_D elements in RDict, where q_D is the number of rogue activations of D by A. Thus, we have (up to probability ϵ_2) that $q \leq q_D$, and the advantage of the attacker (without U queries) is at most $\min\{q_S, q_D\}/|\mathsf{Dict}| + \epsilon_{KC}$.

When adding attacks via U we get this expression to be $\min\{q_S + q_U, q_D + q'_U\}/|\mathsf{Dict}| + \epsilon_{KC} + \epsilon'_C$ and, regardless of the value of q_S, this is at most $(q_D + q'_U)/|\mathsf{Dict}| + \epsilon_{KC} + \epsilon'_C$.

Case 4: D *and* S *corrupted.* The combination of the arguments in cases 2 and 3 implies that an attack when both S and D are corrupted achieves equation (2) where q_D is the number of outputs of F_k computed by A using its knowledge of k and q_S is the number of passwords run by A in its offline dictionary attack based on S's state. ☐