

# Device-Enhanced Password Protocols with Optimal Online-Offline Protection

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

## 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 control 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 most online applications (and many offline systems). A plethora of sensitive information stored in many different contexts therefore depends 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 one-way mapping 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 recently, such as PayPal [1], LinkedIn [4], Blizzard [2] and Gmail [3]. The offline attacks are particularly devastating because a single server break-in may lead to 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.

---

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

†IBM Research. Email: hugo@ee.technion.ac.il. Research supported by ONR Contract N00014-14-C-0113.

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

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

In this paper, we present solutions directed to enhance password protocols against *both online and offline attacks* by an active man-in-the-middle attacker acting on *both* user-server and user-device links, and capable of compromising devices and servers by learning their full internal state (e.g., a server’s password file or the device’s secrets).

## 1.1 Our Contributions

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<sup>1</sup> settings. Moreover, 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, in the PKI setting the scheme can work with unmodified servers running the usual password-over-TLS authentication.

We introduce efficient DE-PAKE protocols with the following properties:

1. *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 and server in the role of  $U$  with password  $p$  and observing whether the server 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 the device still requires a full online guessing attack on the  $U$ - $S$  link and fully compromising the server 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  $q/|\text{Dict}|$ , where  $\text{Dict}$  is the passwords dictionary, the attacker needs to run  $q$  online interactions with  $D$  and  $q$  online interactions with  $S$ . Moreover, even when compromising the device and finding all its secrets, the attacker still needs to run  $q$  online interactions with  $S$ , and if the server is compromised,  $q$  online interactions with  $D$  are necessary. Finally, if both device and server are compromised the adversary must stage a full offline dictionary attack to learn any passwords. We formalize these properties through a security model that extends the traditional PAKE security setting, and then we use this model to prove the security of our schemes.
2. *PKI-agnostic:* The above security is 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 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 possible offline-online protection.

---

<sup>1</sup>We stress that PKI-freeness in password protocols refers to dispensing of PKI for user authentication but some form of secure channel is needed for password registration. Note that the latter process, being much less frequent than regular login, can use special safeguards, such as real-world physical interactions or out-of-band channels.

3. *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*. See Section 5.2.

## 1.2 Relevance 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  $S$ , and establishing a session key between the two, where the user has a password and a personal device  $D$  (e.g., a smartphone) that contains some secret auxiliary information. This secret information is used to increase the security of password authentication by preventing online attacks for an adversary that does not have access to  $D$ . Typically,  $D$  displays a short one-time PIN (OTP), received directly from the server or computed by  $D$  based on a key shared with the server, that the user manually copies over to the authentication terminal in addition to providing her password.

Traditionally, these TFA schemes have been used for increasing resistance to online dictionary attacks. However, as correctly argued by Shirvanian et al. [26], with the increasing vulnerabilities of servers to compromise [5, 6], TFA schemes should also be enhanced to strengthen security against offline attacks. This work [26] presented several schemes for achieving this goal. The main idea underlying all their TFA 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  $s$ . The authentication protocol checks whether the user types the correct password  $p$  and also that it can access the device that stores  $s$ .

There are, however, two important aspects in which the schemes of [26] can be improved, and we do so in this paper. First, all their schemes assume a PKI setting, namely, the user (through a client application) must be in possession of an *authentic* public key of the server which is used to establish a secure channel, e.g., via TLS. If such public key is not available or is compromised, the security of the scheme completely breaks down. Given the vulnerabilities of PKI to certification failures and man-in-the-middle (MitM) attacks (either due to programmatic errors or human mistakes), e.g., [11, 16, 27], reducing the dependency of authentication security on public keys is an increasingly important goal. Second, the schemes of [26] require the authentication server to run a different protocol than currently standard PKI-setting TFA schemes which can inhibit the deployment of such schemes. For example, an individual user (or an application) cannot adopt the schemes from [26] to protect her password stored by a web service without that service being modified to support the new TFA method.

The notion of DE-PAKE scheme we propose (and its efficient realization we construct) addresses many of the same concerns as TFA schemes, e.g. increased security against *on-line* attacks viz-a-viz solely-password based authentication, and it surpasses security offered by existing TFA schemes by offering maximal resistance to *off-line* attacks, i.e. making offline dictionary search impossible in case of server compromise. However, traditional TFA schemes offer also increased security against compromise of the client machine (compared to solely-password authentication). A generic DE-PAKE scheme does not provide this, although the specific DE-PAKE scheme we propose can be extended to offer additional defenses against attacks on the client (see Section 5.3). Moreover, it is possible to combine a DE-PAKE scheme with a traditional MFA mechanism (e.g. a one-time PIN delivered to or generated by the hand-held device) to extend the security properties of DE-PAKE to include the same level of resistance against client compromise as offered by traditional TFA schemes.

Table 1: Glossary

Acronym	Description
<b>PAKE</b>	Password-authenticated key exchange. We refer to password protocols by PAKE as well as to their security model (recalled in Section 3.1).
<b>PKI-Free PAKE</b>	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.
<b>DE-PAKE</b>	A new notion of <i>Device-Enhanced PAKE</i> 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).
<b>PTR</b>	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).
<b>FK-PTR</b>	A specific construction of a PTR scheme using the Ford-Kaliski password hardening technique (Section 4.1).
<b>OPRF</b>	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.
<b>PTR-PAKE</b>	A general name for DE-PAKE protocols built by composing a PTR and a PAKE protocols (their generic security is based on Theorem 3)
<b>FK-PTR-PAKE</b>	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 [18, 20].

### 1.3 Organization and Glossary

We present an overview of our design and proof methodology in Section 2 with full details given in Section 4. Section 3 presents the 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 (or CRS) model; a description of our approach for armoring existing servers against online and offline attacks while keeping the servers unmodified; and a discussion of extensions to address issues related to client security (not covered in our DE-PAKE model). We close with conclusions and future directions.

In Table 1 we provide for easy reference a summary of the main terms (and corresponding acronyms) introduced and used throughout the paper.

## 2 Overview: DE-PAKE Design and Analysis

Our design follows the “password hardening” approach of Ford and Kaliski [13], 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 [13, 17].

The idea is simple: the user memorizes a regular password  $\text{pwd}$  but uses as her password with server  $S$  a value  $\text{rwd} = F_k(\text{pwd})$  where  $F$  is a pseudorandom function and  $k$  is a key held by device  $D$  ( $\text{rwd}$  is a mnemonics for “randomized password”). Before authenticating to  $S$ ,  $U$  contacts  $D$  (through a client application) and obtains  $\text{rwd}$  via a special protocol with  $D$  (in which  $D$  learns *nothing* about  $\text{pwd}$  or  $\text{rwd}$ !).  $U$  then authenticates to  $S$  via a (standard) PAKE protocol using  $\text{rwd}$  as a password. Note that without knowledge of  $k$ , the value  $\text{rwd}$  has full entropy (i.e., it is indistinguishable from uniform in the range set of function  $F$ ), hence dictionary attacks do not apply against  $\text{rwd}$  (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  $\text{pwd}$  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  $F_k(\text{pwd})$  but the protocol leaks no other information about protocol inputs, i.e.  $\text{pwd}$  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  $\text{pwd}$ -related information at  $D$  (the latter would open  $\text{pwd}$  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.

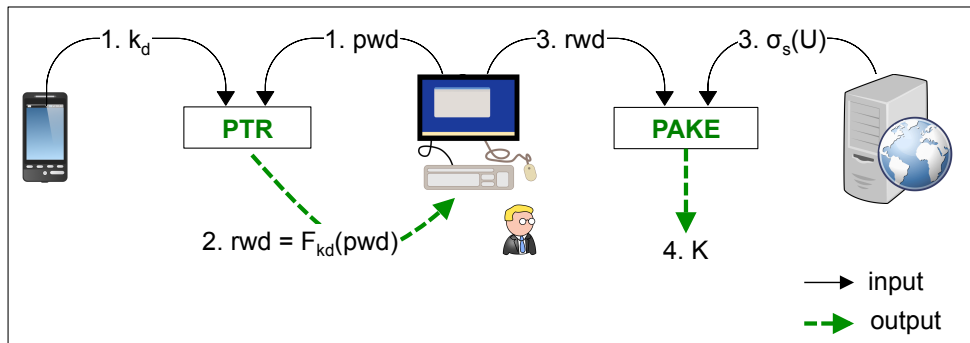


Figure 1: PTR-PAKE Authentication Phase

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  $\sigma_S(U)$  to the PAKE subprotocol denotes the user-specific information stored at  $S$  created in the PAKE initialization using  $\text{rwd}$  in the role of the password (e.g. in a common PKI-based PAKE  $\sigma_S(U)$  would be a salted hash of  $\text{rwd}$ ).

In order to prove the security of such scheme, we first extend (Section 3) the established security models for the PAKE functionality to the DE-PAKE setting. 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  $\text{pwd}$ ) with a secure PAKE protocol (run between  $U$  and  $S$  on the basis of the hardened password  $\text{rwd} = F_k(\text{pwd})$ ) results in a secure DE-PAKE scheme, provided that the PAKE protocol satisfies “security against server compromise” or the more precise notion of *wKCI* resistance introduced and discussed in Section 3.1.1 (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. In Section 5.1 we compose our *FK-PTR* scheme with a specific PAKE protocol - a *wKCI*-resistant version of the single-server variant of threshold PAKE from [18, 20]. Since this PAKE protocol does not require PKI, neither does the resulting DE-PAKE protocol. In addition, in Section 5.2 we expand on the fact that since our *PTR-PAKE* construction can be used with *any* existing

password protocol with resistance to wKCI attacks, one obtains DE-PAKE schemes without changing the server that implements the PAKE protocol.<sup>2</sup> In particular, this allows to use without any change any server that implements the standard PKI-based password-over-TLS protocol.

A PTR scheme is a close variant of OPRF, and the potential of OPRF to strengthen password authentication was recently and independently used in the Pythia system [12], 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 [9] that extends authenticated key exchange models to account for the inherent vulnerability of password protocols to online guessing attacks. We adapt the model to the client-server setting borrowing some of the formalism from [18] (we refer to the client as the “user”). In Section 3.1.1 we describe an extension of the PAKE model to security against server compromise.

**Protocol participants.** There are two types of parties participating in a PAKE protocol, 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  $\text{pwd}$  from a given dictionary  $\text{Dict}$  and interacts with its associated server  $S$  producing a user’s state  $\sigma_S(U)$  that  $S$  stores while  $U$  only remembers its password  $\text{pwd}$ . *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  $\Pi_i^U$  or  $\Pi_i^S$ , where integer pointer  $i$  serves to differentiate between multiple protocol instances executed by the same party. Each protocol session is associated with the following variables: a *session identifier*  $\text{sid}$ , 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*  $\text{pid}$ , and a *session key*  $\text{sk}$ . 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 malformed message is received, etc.). When a session outputs  $\text{sk} \neq \perp$  we say that the session *accepts*.

---

<sup>2</sup>If the PAKE protocol is secure but does not offer wKCI resistance, the resulting scheme does not offer all security properties of DE-PAKE but it still significantly improves upon the security of the PAKE protocol.

**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.

$\text{send}(P, i, P', M)$ : causes message  $M$  to be delivered to instance  $\Pi_i^P$  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  $\Pi_i^U$  does not exist then query  $\text{send}(U, i, S, \text{init})$  creates a new instance  $\Pi_i^U$  which executes with pid =  $S$  on  $U$ ’s chosen password  $\text{pwd}$ . Similarly, if  $\Pi_i^S$  does not exist then  $\text{send}(S, i, U, M)$  creates a new instance  $\Pi_i^S$  which executes with pid =  $U$  on  $S$ ’s input  $\sigma_S(U)$ , 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.)

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

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

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

$\text{test}(P, i)$ : If instance  $\Pi_i^P$  has accepted, this query causes  $\Pi_i^P$  to flip a random bit  $b$ . If  $b = 1$  the instance’s session key  $\text{sk}$  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 [18] 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 [9], 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  $\text{send}(P, i, P', M)$  query is *rogue* if it was not generated and/or delivered according to the specification of the protocol. That is, the 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  $\text{send}(P, i, P', M)$  query where  $P$  is uncorrupted and  $P'$  is corrupted. Throughout the paper we will refer to messages delivered through rogue send queries as *rogue activations* by  $A$ .

*Matching sessions.* A session in instance  $\Pi_i^P$  and a session in instance  $\Pi_j^{P'}$  are said to be *matching* if both have the same session identifier  $\text{sid}$  (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  $\Pi_i^P$  with peer  $P'$  s.t.  $\{P, P'\} = \{U, S\}$  is called *fresh* if none of the queries  $\text{corrupt}(U)$ ,  $\text{corrupt}(S)$ ,  $\text{compromise}(S, U)$ ,  $\text{reveal}(P, i)$  or  $\text{reveal}(P', i')$  were issued, where  $\Pi_{i'}^{P'}$  is an instance whose session matches  $\Pi_i^P$  (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  $\text{Adv}_A^{\text{PAKE}} = 2 \cdot \Pr [A \text{ wins against PAKE}] - 1$ .

**Definition 1.** A PAKE protocol PAKE is  $(q_S, q_U, T, \epsilon)$ -secure if it is correct and for any password dictionary Dict and any attacker  $A$  that runs in time  $T$ , it holds that  $\text{Adv}_A^{\text{PAKE}} \leq \frac{q_U + q_S}{|\text{Dict}|} + \epsilon$  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  $2^d$ .* For notational convenience, we will use  $2^d$  to denote the size of password dictionaries, although our treatment does not depend on these sizes being a power of 2.

### 3.1.1 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* [15], 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.<sup>3</sup> 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* [10]. 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.

Here we extend the above indistinguishability-based PAKE formalism to capture resistance to weak KCI attacks (*wKCI-resistance*) through the following game (this formalism is well suited to typical ROM-based implementations that hash the password). We consider a PAKE setting as before except for the following changes. A 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 PAKE attacker  $A$  is only given a random subset of Dict of size  $q$  as well as the server's state  $\sigma_S(U)$ . In addition,  $A$  is required, to choose the test session at an instance of  $S$  with peer  $U$  (in the regular PAKE case this is not allowed since  $S$  is U-compromised). We call a PAKE scheme  $\epsilon$ -wKCI-resistant if for any  $q \leq 2^d$ , the attacker's advantage in the above 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.

## 3.2 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. 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  $\Pi_i^D$  are created similarly to user and server instances, and are activated by  $A$  via send

<sup>3</sup>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  $U$ ).



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.

The attacker’s goal remains the same as before, namely, win the test experiment at a user or server instance exactly 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 [23] 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  $\Pi_i^S$  at server S with peer U is considered fresh *even if*  $\text{corrupt}(S)$  or  $\text{compromise}(S, U)$  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  $\Pi_i^S$* . 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  $\Pi_i^S$ . 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  $\text{Adv}_A^{\text{DPK}} = 2 \cdot \text{Pr} [A \text{ wins against DPK}] - 1$ .

**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:

$$\text{Adv}_A^{\text{DPK}} \leq \frac{\min\{q_U + q_S, q'_U + q_D\}}{2^d} + \epsilon. \quad (1)$$

2. If D is corrupted then  $\text{Adv}_A^{\text{DPK}} \leq (q_U + q_S)/2^d + \epsilon$ .
3. If S is corrupted then  $\text{Adv}_A^{\text{DPK}} \leq (q'_U + q_D)/2^d + \epsilon$ .
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 could define the above bounds more generally by replacing the expression  $\min\{q_U + q_S, q'_U + q_D\}/2^d + \epsilon$  with some more general function of the  $q$  parameters but we choose the simpler and more natural case. Also, note that the expression (1) 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 (modeling and implementing DE-PAKE via a (2,2)-TPAKE).** In a  $(t + 1, n)$  threshold-PAKE (TPAKE) (cf. [18]), 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 (2,2)-TPAKE by letting D and S act as the two servers in a (2,2)-TPAKE scheme (this would imply the first three conditions of Definition 2 but the last one should be added as an additional requirement). Moreover, one can use (2,2)-TPAKE as the basis for the definition of DE-PAKE where the user only authenticates to one of the parties. 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 TPAKE. (A TPAKE 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 5.3).

## 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 [13] (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., [15, 18]). Moreover, if the PAKE scheme is in the password-only model<sup>4</sup> then the DE-PAKE scheme is also secure in this model.

<sup>4</sup>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

### 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  $\tau$  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  $\Pi$ .

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

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

### Login Phase

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

- **User-Server Interaction (PAKE)**

Follows the specification of the PAKE protocol  $\Pi$  where U uses  $\text{rwd}$  as its password.

Figure 2: The FK-PTR-PAKE Scheme

## 4.1 The FK-PTR Scheme

The particular instantiation of a PTR scheme we call FK-PTR is based on Ford-Kaliski’s “password hardening” [13] or its more general interpretation as an Oblivious PRF (OPRF)<sup>5</sup> [14, 21, 22]. 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: It is shown as the interaction between U and D by which U retrieves a random value  $\text{rwd}$  with the help of its password  $\text{pwd}$ . At initialization, U chooses and remembers password  $\text{pwd}$  while D chooses and stores  $k \leftarrow Z_q$ . To retrieve  $\text{rwd}$ , U first blinds  $\text{pwd}$  by raising the hashed value  $H'(\text{pwd})$  to a random exponent  $\rho$ , and send it to D. This perfectly hides  $\text{pwd}$  from D and from any

run by a user; they require the same integrity guarantees as the program itself.

<sup>5</sup>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 the computation, and one that holds an input and learns the output of the function on that input and nothing else. 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. [14, 21], and their properties do not seem sufficient for the PTR protocol. On the other hand, the recent definition of UC OPRF [19] involves verifiability which we do not provide (hence the FK-PTR construction shown here uses fewer exponentiations than the OPRF scheme of [19] 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.

eavesdropper on the  $U - D$  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  $H'(\text{pwd})^k$ . Finally,  $U$  hashes this value with  $\text{pwd}$  to obtain the randomized password  $\text{rwd}$ .

Note that  $D$  contains no information related to  $\text{pwd}$  hence an attacker interacting with  $D$  or even breaking into it learns nothing about  $\text{pwd}$ . Also,  $U$  does not run any test on the value reconstructed in the FK-PTR protocol. Hence, an attacker that interacts with  $U$  in the role of  $D$  does not learn anything about  $\text{pwd}$  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  $\text{Dict}$  of size  $2^d$  for some  $d$  (the power of two is chosen for notational convenience only). User  $U$  is initialized with password  $\text{pwd} \leftarrow \text{Dict}$  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  $\text{pwd}$ ,  $U$  outputs the secret  $\text{rwd} = F_k(\text{pwd})$ . 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.

We use the notation  $\text{RDict} = \{F_k(p) : p \in \text{Dict}\}$  where  $k$  is  $D$ 's secret key.

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

To capture this property we define the following experiment referred to as the *distinguishing test*. Let  $\text{rwd} = F_k(\text{pwd})$  and choose  $r \leftarrow \text{RDict} \setminus \{\text{rwd}\}$ .  $A$  is given both  $\text{rwd}, r$  (in random order) and it needs to guess which one equals  $F_k(\text{pwd})$ .

*Attack avenue 2: Learning values in  $\text{RDict}$ .* Since  $A$  can learn values in  $\text{RDict}$  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  $\text{RDict}$  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  $\text{RDict}$ , 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  $F_k(\text{pwd}')$  where  $\text{pwd}' \in \text{Dict}$  is known to A. In this case, A can observe the PAKE run of U with S and see if  $\text{pwd}'$  is the correct password. This allows A to test passwords in  $\text{Dict}$  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  $\text{RDict}$  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  $\text{RDict} \setminus \{\text{pwd}\}$ , we choose the stronger notion by adding an explicit requirement against such possibility.

On the basis of this discussion we will require the probability that the output from U in a PTR run is  $F_k(\text{pwd}')$  for  $\text{pwd}' \in \text{Dict} \setminus \{\text{pwd}\}$  to be negligible.

*Attack avenue 4: Running U on passwords outside  $\text{RDict}$ .* 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  $\text{RDict}^*$  different than  $\text{RDict}$  (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  $F_{k'}(\text{pwd})$ , i.e., with a value uniformly distributed over  $\text{RDict}^* = \{F_{k'}(p) : p \in \text{Dict}\}$  where  $\text{RDict}^*$  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  $p \in \text{Dict}$ . If  $p = \text{pwd}$ , A succeeds in the impersonation and learns  $\text{pwd}$ .<sup>6</sup> 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  $\beta'$ , 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 will require that *any* attack strategy by A for generating a dictionary  $\text{RDict}^*$  will induce 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

<sup>6</sup>This attack recovers  $\text{pwd}$  with  $2^d$  impersonation attempts (of S) against U and it only requires one value  $F_{k'}(\text{pwd})$  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.

dictionary  $\text{RDict}^* = \{f_c(p) : p \in \text{Dict}\}$  of size  $|\text{Dict}|$ .)

We are now ready to define PTR security.

**Definition 3.** We say that a PTR scheme is  $(q_D, q_U, T, \epsilon)$ -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,  $\epsilon$  is an upper bound on the values  $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$  defined as follows (these  $\epsilon_i$  are functions of  $q_D, q_U, T$  and they correspond to the above attack avenues):

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  $\text{RDict}$  following attack avenue 2 is at most  $\epsilon_2$ ;
3. the probability that  $U$  outputs  $F_k(\text{pwd}')$  for  $\text{pwd}' \in \text{Dict} \setminus \{\text{pwd}\}$  is at most  $\epsilon_3$ ;
4. the probability that  $f_c$ , as defined in attack avenue 4, is not 1-1 is less than  $\epsilon_4$ .

where in all four cases the probability goes over random PTR key  $k$  and random  $\text{pwd}$  in  $\text{Dict}$ .

### 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 [8, 22]:** Let  $G$  be a group of prime order  $q$  and  $k$  a random value in  $Z_q$ . Let  $\text{DH}_k$  be an oracle<sup>7</sup> that on input  $g \in G$  outputs  $g^k$ , and let  $\text{DDH}_k$  be an oracle that on input a pair  $(a, b)$  answers whether  $b = a^k$ . We say that  $G$  satisfies the  $\epsilon_{\text{omg}}$ -OMG-DH assumption for function  $\epsilon_{\text{omg}}$  if any attacker  $A$  that runs in time  $T$  has probability at most  $\epsilon_{\text{omg}}(T, q_{dh}, q_{ddh})$  to win the following game:  $A$  is given access to the  $\text{DH}_k$  and  $\text{DDH}_k$  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  $\epsilon_{\text{omg}}(\cdot)$ -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  $\text{Dict} \subset \{0, 1\}^\tau$  is  $(q_D, q_U, T, \epsilon)$ -secure where  $\epsilon = \max\{\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4\}$  with  $\epsilon_1 = 0, \epsilon_2 \leq T/2^\tau + \epsilon_{\text{omg}}(T, q_D, q_H), \epsilon_3 \leq 1/2^\tau, \epsilon_4 \leq |\text{Dict}|^2/2^\tau$ .

*Proof.* To show that  $\epsilon_1 = 0$ , note that the only information  $A$  sees related to the particular value  $\text{pwd}$  is  $\alpha = (H'(\text{pwd}))^\rho$  but since  $\rho$  is chosen by  $U$  uniformly in  $Z_q$ ,  $\alpha$  is uniformly distributed in  $G$  independently of  $\text{pwd}$ . Thus, the view of  $A$  is independent of  $U$ 's password  $\text{pwd}$  and the probability of  $A$  to win the distinguishing test is  $1/2$ . The bound on  $\epsilon_2$  follows from Lemma 2 below proven based on the OMG-DH assumption. The bound on  $\epsilon_3$  follows from the fact that  $\text{pwd}$  is included under  $H$ , hence the probability that for some  $\text{pwd}' \in \text{Dict} \setminus \{\text{pwd}\}$ , we have  $H(\text{pwd}', (H'(\text{pwd}'))^k) = H(\text{pwd}, (H'(\text{pwd}))^k)$  is  $2^{-\tau}$  where  $\tau$  is the length of the output from  $H$ . Similarly, the bound on  $\epsilon_4$  follows from the collision resistance properties of the (random)  $H$ .  $\square$

Note: The bound  $|\text{Dict}|^2/2^\tau$  on  $\epsilon_4$  can be reduced significantly if one relaxes requirement 4 of PTR security to allow for some deviation from injectiveness, e.g., allowing  $\text{RDict}$  to be of size  $\alpha \cdot |\text{Dict}|$  for some  $\alpha$ , say  $\alpha = 1/2$ .

<sup>7</sup> $\text{DH}_k$  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.

**Lemma 2.** *Let  $G$  be a group where the  $\epsilon_{omg}$ -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  $\text{RDict}$  (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  $\epsilon_{omg}$ -OMG-DH group  $G$ , we build an attacker  $A'$  against OMG-DH in group  $G$ .  $A'$  gets access to a  $\text{DH}_k$  and  $\text{DDH}_k$  oracles and an input in the form of an ordered set  $R = g_1, \dots, g_N$  of random elements in  $G$ .  $A'$  runs  $A$  on a simulated run of FK-PTR.  $A'$  uses  $\text{DH}_k$  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  $R = g_1, \dots, g_N$  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  $\text{pwd}$ , sets  $H'(\text{pwd}) = g_1$  (i.e., this is done prior to answering any  $H'$  query from  $A$ ), and queries  $g_1$  to  $\text{DH}_k$  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  $\beta$  from  $A$  it computes  $r' = H(\text{pwd}, \beta^{1/\rho})$ . When  $A$  delivers a message  $\alpha'$  to  $D$ ,  $A'$  responds to it as follows. If  $\alpha'$  was output by  $U$ , in which case  $A'$  knows  $\rho$  such that  $\alpha' = g_1^\rho$ ,  $A'$  responds with  $y_1^\rho$ ; otherwise  $A'$  queries  $\alpha'$  from  $\text{DH}_k$ . In addition, for inputs  $(x, y)$  to  $H$ ,  $A'$  checks that  $H'(x)$  was queried and if so it checks, using the  $\text{DDH}_k$  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  $\text{pwd}$  and  $g_1$  ( $A$  only sees values of the form  $g_1^\rho$  for one-time random  $\rho$ 's) hence it is independent of  $\text{pwd}$  (which is computed as  $H(\text{pwd}, (H'(\text{pwd}))^k)$ ).

Denote by  $R'$  the set of values in  $\text{RDict}$  output by  $A$ . Let  $E_1$  denote the event that  $R'$  contains a value  $r' = H(p, (H'(p))^k)$  for  $p \in \text{Dict}$  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  $|L| \leq q_D$  and therefore  $|R'| \leq q_D$  as claimed by the lemma.

$A'$  queries  $\text{DH}_k$  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  $q_{dh} = q_D + 1$ , and together with the assumption  $q_{dh} \geq |L'| = |L| + 1$  we obtain that  $|L| \leq q_{dh} - 1 = q_D$ . If  $(g_1, g_1^k) \in L$  (meaning that  $g_1$  was also queried by  $A$  through a rogue activation) then  $q_{dh} = q_D$  and together with  $q_{dh} \geq |L'| = |L|$  we obtain that  $|L| \leq q_{dh} = q_D$ .

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  $\text{DH}_k$  to that of a  $D$  activation and the cost of a  $H'$  call to that of a  $\text{DDH}_k$  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 5.1.

**Theorem 3.** *Let  $P$  be a  $(q_D, q'_U, T_P, \epsilon_P)$ -secure PTR scheme and  $\Pi$  be a  $(q_S, q_U, T_\Pi, \epsilon_\Pi)$ -secure PAKE protocol that is also  $\epsilon_{KC}$ -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. □

## 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.

### 5.1 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 [18, 20]. More precisely, the PAKE protocol we use is an adaptation of the variant of the TPAKE protocol from [18, 20] 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 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 next the key computation corresponding to the HMQV protocol [24] in terms of the KE formula used in the protocol:

- Set  $e_u = H(X_u, S)$ ,  $e_s = H(X_s, U)$ , where  $S, U$  represent the parties’ identities.
- $S$  computes  $K = H((X_u P_u^{e_u})^{x_s + e_s p_s})$ ;  $U$  computes  $K = H((X_s P_s^{e_s})^{x_u + e_u p_u})$ .



**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  $\tau$  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  $\text{rwd} = F_{k_d}(\text{pwd})$ .
- PAKE Initialization:
  1. S chooses  $p_s \in_{\mathbb{R}} Z_q$  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  $z \in_{\mathbb{R}} \{0, 1\}^\tau, k_s \in_{\mathbb{R}} Z_q$ ;  
sets values  $c = z \oplus F_{k_s}(\text{rwd}), r = f_z(0), C = H(r, \text{rwd}, c), p_u = f_z(1) \bmod q$ ;  
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  $\alpha = (H'(\text{rwd}))^\rho$  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  $x_s \in_{\mathbb{R}} Z_q$  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  $K = KE(p_s, x_s, P_u, X_u)$  and outputs session key  $SK = f_K(0)$ .
3. U proceeds as follows:
  - (a) Sets  $z = c \oplus H(\text{rwd}, \beta^{1/\rho}), r = f_z(0), p_u = f_z(1) \bmod q$ .
  - (b) Aborts unless the following conditions hold:  $\beta \in G, C = H(r, \text{rwd}, c), m_u = f_z(2, P_u, P_s)$ .
  - (c) Computes  $K = KE(p_u, x_u, P_s, X_s)$  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 [18, 20]

## 5.2 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. Since the password authentication protocol is independent of the hardening procedure, it can be the standard password-over-TLS protocol used in the defacto standard PKI setting. (Note that if the server stores a salted password hash then the password-over-TLS authentication is wKCI-resistant.)<sup>8</sup> 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 refer to this mechanism as Server-Transparent DE-PAKE.

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.

We are currently studying the application of this server-transparent FK-PTR-PAKE scheme to building a secure password manager.

## 5.3 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.

Moreover, resistance to phishing attacks can be achieved if in the computation of  $rwd$ , one concatenates the URL being accessed to  $pwd$  (i.e.,  $rwd$  being computed as  $F_{K_d}(pwd|url)$ ). This is similar to the PwdHash approach [25] 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.

---

<sup>8</sup>Moreover, if the server implements password authentication badly and does not use salted hash, the resulting PTR+PAKE instance will not have strong KCI-resistance, but it will not endanger any other instances where the user re-uses the same password  $pwd$ . This is because learning  $rwd_i = F_{k_i}(pwd)$  where  $k_i$  is the OPRF key which  $D$  associates with server  $S_i$ , does not help in finding either  $pwd$  or  $rwd_j = F_{k_j}(pwd)$  for  $k_j$  independent from  $k_i$  (assuming device  $D$  remains uncorrupted).

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., [18]), 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.

## 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] M. Bellare, C. Namprempe, 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.
- [9] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology – Eurocrypt*, 2000.
- [10] 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.
- [11] I. Dacosta, M. Ahamad, and P. Traynor. Trust no one else: Detecting MITM attacks against SSL/TLS without third-parties. In *European Symposium on Research in Computer Security*, 2012.

- [12] A. Everspaugh, R. Chaterjee, S. Scott, A. Juels, and T. Ristenpart. The pythia prf service. In *24th USENIX Security Symposium (USENIX Security 15)*, 2015.
- [13] 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.
- [14] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography*. 2005.
- [15] 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.
- [16] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov. The most dangerous code in the world: Validating ssl certificates in non-browser software. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2012.
- [17] D. P. Jablon. Password authentication using multiple servers. In *The Cryptographer’s Track at RSA Conference (CT-RSA)*. 2001.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] S. Jarecki and X. Liu. Fast secure computation of set intersection. In *Security and Cryptography for Networks*. 2010.
- [23] H. Krawczyk. HMQV: A high-performance secure diffie-hellman protocol. pages 546–566, 2005.
- [24] H. Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In *Advances in Cryptology-CRYPTO*, 2005.
- [25] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *Usenix security Symposium*, 2005.
- [26] 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.
- [27] J. Sunshine, S. Egelman, H. Almuhiemedi, N. Atri, and L. F. Cranor. Crying wolf: An empirical study of ssl warning effectiveness. In *USENIX Security Symposium*, 2009.

## A Proofs

### A.1 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  $\Pi$  be a  $(q_S, q_U, T, \epsilon)$ -secure PAKE protocol. Then the following holds for any PAKE-adversary  $A$  against  $\Pi$ . Let  $\text{Dict} \subset \{0, 1\}^\tau$  be a dictionary composed of the union of two disjoint sets  $\text{Dict}_1$  and  $\text{Dict}_2$ , where  $\text{Dict}_1$  is known to  $A$  and  $\text{Dict}_2$  is chosen as a random subset of  $\{0, 1\}^\tau \setminus \text{Dict}_1$  and unknown to  $A$ . Then the advantage of  $A$  against  $\Pi$  running with dictionary  $\text{Dict}$  is at most*

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

*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):

$$\begin{aligned} & Pr[A \text{ wins} : p \in \text{Dict}_1] \cdot Pr[p \in \text{Dict}_1] + \\ & Pr[A \text{ wins} : p \in \text{Dict}_2] \cdot Pr[p \in \text{Dict}_2] \\ & \leq \frac{1}{2} + \frac{\min\{q_A, |\text{Dict}_1|\}}{|\text{Dict}_1|} \frac{|\text{Dict}_1|}{|\text{Dict}|} + \frac{q_A}{|\text{Dict}_1|} \frac{|\text{Dict}_2|}{|\text{Dict}|} + \epsilon \\ & \leq \frac{1}{2} + \frac{\min\{q_A, |\text{Dict}_1|\}}{|\text{Dict}|} + \frac{2q_A}{2^\tau} + \epsilon \end{aligned}$$

as claimed in the lemma.

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

### A.2 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  $\text{pwd}$  as in attack avenue 4. Since  $A$  has no information about  $\text{pwd}$  (first PTR requirement), this is equivalent to  $U$  choosing a random independent password  $\text{rwd}^*$  from the dictionary  $\text{RDict}^* = f_c(\text{Dict})$

(which by the 1-1 requirement is of the same size as user  $U$ 's dictionary  $\text{Dict}$ ). Also, since by the third requirement of PTR security,  $\text{rwd}^*$  is different than  $U$ 's real password  $\text{rwd} = F_k(\text{pwd})$ , then the runs of  $U$  with  $\text{rwd}^*$  are independent from those with  $\text{rwd}$  (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  $\text{rwd}^*$  as a separate user from  $U$ , one created by the attacker with dictionary  $\text{RDict}^* = f_c(\text{Dict})$ . Note that  $U$  does not have  $\text{rwd}^*$  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.<sup>9</sup>

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  $\text{RDict}^*$ , 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  $\text{rwd}^*$ ). Thus, in what follows, we assume a setting (or game) where each split user runs with a password  $\text{pwd}^*$  that is independent of  $\text{pwd}$  and independent of other users'  $\text{pwd}^*$ , and that these passwords are chosen from dictionaries  $\text{RDict}^*$  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  $\Pi^*$  to denote the PAKE protocol  $\Pi$  when run against an attacker that can create split users with independent passwords as above. The PAKE security of  $\Pi$  implies the PAKE security of  $\Pi^*$  (the PAKE model requires  $\Pi$  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  $\Pi$ . That is, we build a PAKE attacker  $\text{SIM}$  against  $\Pi$  given a DE-PAKE attacker  $A$  against the composed DE-PAKE scheme  $C$ . For this  $\text{SIM}$  simulates the PTR part of the protocol as follows. Let  $\text{Dict}$  be the dictionary used by the target user  $U$ .  $\text{SIM}$  chooses  $k$  for  $D$  and  $\text{pwd} \leftarrow \text{Dict}$  for  $U$  in the PTR game. It defines the dictionary on which  $\Pi^*$  runs as  $\text{RDict} = \{F_k(p) : p \in \text{Dict}\}$ . By the 1-1 property of  $F_k$ ,  $\text{RDict}$  is of the same size as  $\text{Dict}$  (note that  $\Pi$  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 ( $\text{pwd}$  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  $\epsilon_1$ , of  $\text{pwd}$  and of the password  $\text{rwd} = F_k(\text{pwd})$  used by  $U$  in protocol  $\Pi$ .

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  $\text{rwd}$  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  $\text{RDict}$  from which  $U$ 's password  $\text{rwd}$  is chosen. Specifically, by requirement 2 of PTR security, we can assume (up to an advantage difference of  $\epsilon_2$ ), that  $A$  knows at most  $q_D$  elements in  $\text{RDict}$ , where  $q_D$  is the number of rogue activations of  $D$  by  $A$ . In the view of  $A$ , the rest of  $\text{RDict}$  is distributed uniformly (or pseudorandomly) in  $\{0, 1\}^\tau$ . Thus, we are in the setting of Lemma 4, hence the probability of  $A$  winning the DE-PAKE game in a session at  $U$  or  $S$  is at most

$$\frac{\min\{q_U + q_S, q_D\}}{|\text{Dict}|} + \frac{q_U + q_S}{2^{\tau-1}} + \epsilon_{\Pi}. \quad (2)$$

<sup>9</sup>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.

We now consider  $\Pi$  activations of  $U$  running with a password from an attacker-induced dictionary  $\text{RDict}^*$ , or the equivalent  $\Pi^*$ -activation by  $A$  of a split user  $U^*$ . Such user runs with a password  $\text{pwd}^*$  from a dictionary  $\text{RDict}^*$  of the same size as  $\text{Dict}$  and where  $\text{pwd}^* \neq \text{pwd}$ ; in particular,  $\text{pwd}^*$  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  $\Pi^*$ , we have that its success is at most  $q_U^*/|\text{RDict}^*| = q_U^*/|\text{Dict}|$  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  $\Pi$  of user  $U$  thus the total success probability of  $A$  against split users (i.e., against  $U$  running on an induced password  $\text{pwd}^*$ ) is bounded by  $q_U/|\text{Dict}|$ .

However, note that *each activation* of  $U$  in  $C$  with an induced password other than  $\text{pwd}$  (equivalently, the activation of a split user  $U^*$  in  $\Pi^*$ ) 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  $\min\{q_U, q'_U\}/|\text{Dict}|$  (i.e., this form of attack can be exploited only if the activation of  $U$  as a  $\Pi$  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  $\min\{q_U, q'_U\}/|\text{Dict}|$  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  $\text{pwd}$  while the  $q_U$  in  $\min\{q_U, q'_U\}/|\text{Dict}|$  counts rogue activations running on an unregistered password  $\text{pwd}^*$ . 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

$$\begin{aligned} & \frac{\min\{p_U + q_S, q_D\}}{|\text{Dict}|} + \frac{\min\{p_U^*, q'_U\}}{|\text{Dict}|} + \frac{p_U + q_S}{2^{\tau-1}} + \epsilon_\Pi \\ & \leq \frac{\min\{p_U + q_S + p_U^*, q_D + q'_U\}}{|\text{Dict}|} + \frac{p_U + q_S}{2^{\tau-1}} + \epsilon_\Pi. \end{aligned}$$

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

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

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  $\text{RDict}$  (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),  $A$ 's advantage (even with  $|\text{Dict}|$  offline  $F_k$  computations) is at most  $(q_U + q_S)/|\text{Dict}| + \epsilon'_C$ .

*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  $\Pi$  being  $\epsilon_{KC}$ -wKCI-resistant, we have that an attacker against  $\Pi$  that knows  $q$  passwords from the dictionary  $\text{RDict}$  has advantage at most  $\min\{q_S, q\}/|\text{Dict}| + \epsilon_{KC}$ , 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  $\epsilon_2$ ) that  $q \leq q_D$ , and the advantage of the attacker (without U queries) is at most  $\min\{q_S, q_D\}/|\text{Dict}| + \epsilon_{KC}$ . When adding attacks via U we get this expression to be  $\min\{q_S + q_U, q_D + q'_U\}/|\text{Dict}| + \epsilon_{KC} + \epsilon'_C$  and, regardless of the value of  $q_S$ , this is at most  $(q_D + q'_U)/|\text{Dict}| + \epsilon_{KC} + \epsilon'_C$ .

*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. □