Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only Model

Stanislaw Jarecki[1], Aggelos Kiayias[2], and Hugo Krawczyk[3]

[1] University of California Irvine, stasio@ics.uci.edu
[2] National and Kapodistrian University of Athens, aggelos@kiayias.com
[3] IBM Research, hugo@ee.technion.ac.il

Abstract. In a *Password-Protected Secret Sharing (PPSS)* scheme with parameters (t, n) (formalized by Bagherzandi et al. [2]), a user Alice stores secret information among n servers so that she can later recover the information solely on the basis of her password. The security requirement is similar to a (t, n)-threshold secret sharing, i.e., Alice can recover her secret as long as she can communicate with $t+1$ honest servers but an attacker gaining access to t servers cannot learn any information about the secret. In particular, the system is secure against offline password attacks by an attacker controlling up to t servers. On the other hand, accounting for inevitable on-line attacks one allows the attacker an advantage proportional to the fraction of dictionary passwords tested in on-line interactions with the user and servers.

We present the first round-optimal PPSS scheme, requiring just one message from user to server and from server to user, and prove its security in the challenging password-only setting where users do not have access to an authenticated public key. The scheme uses an Oblivious PRF whose security we define using a UC-style ideal functionality for which we show concrete, truly practical realizations in the random oracle model as well as standard-model instantiations. As an important application we use this scheme to build the first single-round password-only Threshold-PAKE protocol in the CRS and ROM models for arbitrary (t, n) parameters with no PKI requirements for any party (clients or servers) and no inter-server communication. Our T-PAKE protocols are built by combining suitable key exchange protocols on top of our PPSS schemes. We prove T-PAKE security via a generic composition theorem showing the security of any such composed protocol.

1 Introduction

Remarkably, passwords have become a fundamental pillar of electronic security. That's quite a high task for these low-entropy easily-memorable easily-guessed short character strings. In spite of repeated evidence of their vulnerability to misuse and attack, passwords are still in widespread use and will probably remain as such for a long while. The portability of passwords makes them ubiquitous keys to access remote services, open computing devices, decrypt encrypted files, protect financial and medical information, etc. Replacing passwords with long keys requires storing these keys in devices that are not always available to the user and are themselves at risk of falling in adversarial hands, hence endangering these keys and the data they protect.

An increasingly common solution to the problem of data security and availability is to store the data itself, or at least the keys protecting its security, at a remote server, which in turn is accessed using a password. This requires full trust in this single server and the one password. In particular, compromising such a server (or just its password file) is sufficient to crack most passwords stored at it through an *off-line* dictionary attack. Indeed, loss of millions of passwords to such attacks are common news nowadays [31]. Unfortunately, off-line attacks are unavoidable in single-server scenarios. A natural approach to solving this problem is to distribute the above trust over a set of servers, for example by sharing information among these servers using a secret sharing scheme. However, how does the user access these servers? Using the same password in each of these servers makes the off-line password recovery attack even worse (as it can be performed against any of these servers) while memorizing a different password for each server is impractical.

PPSS. The above problem and a framework for solution is captured by the notion of Password-Protected Secret Sharing (PPSS), originated by the work of Ford and Kaliski [16] and Jablon [18] and formalized by Bagherzandi et al. [2]. In such a scheme, parametrized by (t, n), user Alice has some secret information sc that she wants to store and protect, and be able to later access on the basis of a single password pw. (Secret sc can represent any form of information, but it is best to think of it as a cryptographic key which protects some cryptographic capability.) The scheme has an *initialization phase* where Alice communicates with each one of a set of n servers $S_1, \ldots, S_n$ after which each server S_i stores some information ω_i associated with user Alice. When Alice needs to retrieve sc, she performs a *reconstruction protocol* by interacting with at least $t + 1$ servers where the only input from Alice is her password pw.

The main requirements from this protocol are, informally: (i) an attacker breaking into t servers cannot gain any information on sc other than by correctly guessing Alice's password and running an on-line attack with it (more on this below). It follows, in particular, that off-line attacks on the password are not possible as long as the attacker has not compromised more than t servers. In this case, the only avenue of attack against the secrecy of sc is for the attacker to select one value pw' from a given dictionary D of passwords (from which the user has selected a password at random) and check its validity by interacting with the user and servers using pw' as the password. If the overall number of interactions between the attacker and the user, and between the attacker and the servers, is q then we allow the attacker to break the semantic security of sc with advantage $q/|D|$. Moreover, we will require that "testing" a guessed password by impersonating the user to the servers will require interacting with $t + 1$ different servers. (ii) Soundness: Similarly, a compromise of up to t servers cannot allow an attacker to make the user reconstruct a wrong secret $\mathsf{sc}' \neq \mathsf{sc}$, except with probability proportional to the number of *on-line* interactions between the attacker and the user. This is a necessary exception as the attacker can isolate the user and simulate a run with the servers with a password pw' and secret sc' chosen by the adversary; what's required is that *only if* pw' happens to be the user's password will the attack succeed. Additionally, a desirable property is (iii) Robustness: Alice can correctly reconstruct sc as long as (a) no more than t servers are corrupted and (b) Alice communicates without disruptions with at least $t+1$ honest servers. Note that robustness can only be achieved if $2t+1 \leq n$ while the other properties do not impose such intrinsic limitation.

T-PAKE. While PPSS schemes have many uses such as for retrieving keys, credentials, data, and so on, the main PPSS application is for bootstrapping a *Threshold Password-Authenticated Key Exchange (T-PAKE)* [27]. In a (t, n) T-PAKE protocol, a user with a single password is to establish authenticated keys with any given subset of the n servers, such that security of the keys established with uncorrupted servers is guaranteed as long as there are no more than t corrupted servers. PPSS schemes make it possible to build T-PAKE protocols by combining the PPSS scheme with a regular key exchange protocol in a modular and generic way. This allows one to focus on the PPSS design which by virtue of being a much simpler primitive, e.g., avoiding the intricacies of the security of (password) authenticated key exchange protocols, is likely to result in simpler and stronger solutions, as is indeed demonstrated by our results below.

Prior/Concurrent Work and Our Contributions

For the general case of (t, n) parameters, Bagherzandi et al. [2] showed a PPSS scheme in the random oracle model (ROM) where the reconstruction protocol involves three messages between the user and a subset of $t+1$ servers (effectively 4 messages in the typical case that the user initiates the interaction). However, if any of these servers deviates from the correct execution of the protocol, the protocol needs to be re-run with a new subset of servers, which potentially increases the number of protocol rounds to $O(n)$. Another significant shortcoming of the PPSS solution from [2] is that it is secure only in the PKI model, namely, where the user can authenticate the public keys of the servers. Indeed, if the attacker can induce the user to run the protocol on incorrect public keys, the protocol of [2] becomes completely insecure. Thus, [2] leaves at least two open questions: Do PPSS protocols with *optimal single-round communication* exist (i.e., requiring a single message from user to server and single message from server to user), and can such protocols work in the *password-only model*, namely when the user does not have a guaranteed authentic public key.

We answer both questions in the affirmative by exhibiting a PPSS protocol for a general (t, n) setting with optimal single-round communication (in ROM) which works in the password-only model. Concurrently to our work, Camenisch et al. [7] have also presented a general (t, n) setting PPSS which works in the password-only model (and ROM). Their protocol sends 10 messages between the user and each server, its total communication complexity is $O(t^2)$, and the computational cost is more than 7 times the cost of our solution. Moreover, its robustness is fragile in the same way as that of [2], i.e. the user runs the reconstruction protocol with a chosen subset of $t+1$ players, and the protocol must be re-started if any server in this chosen group deviates. By contrast, the protocol we present has 2 messages and $O(n \log n)$ (worst-case) communication complexity, which reduces to $O(n)$ if the user caches $O(n)$ data between reconstruction protocol instances. Our protocol also has stronger robustness guarantee, namely, Alice recovers her shared secret sc in the single protocol instance as long as it has unobstructed communication with at least $t+1$ honest servers and if $2t+1 \leq n$. While [7] formalize a UC functionality for PPSS (which they call "TPASS", for Threshold Password-Authenticated Secret Sharing) and prove their protocol to realize this functionality, we model the security of a PPSS scheme in the password-only setting with a game-based notion. We show that our game-based security notion is strong enough to imply the security of a natural T-PAKE construction built on top of a PPSS scheme. However, a PPSS satisfying a UC formalization might simplify the use of PPSS in the design of other cryptographic schemes, which leaves designing a UC secure PPSS with low message complexity and good robustness as an interesting open question.

Our PPSS construction is based on a novel version of so-called Oblivious Pseudorandom Function (OPRF) [17] and our contributions touch on three distinct elements, OPRF's, PPSS, and T-PAKE's, which we discuss next.

OPRF. The basic building block of our PPSS construction is a *Verifiable* Oblivious PRF (V-OPRF). Oblivious PRF (OPRF) was defined [17, 20] as a protocol between two parties, a server and a user, where the first holds the key k for a PRF function f while the latter holds an argument x on which $f_k(\cdot)$ should be evaluated. At the end of the protocol the user learns $f_k(x)$ and is convinced that such value is properly evaluated while the server learns nothing. Formalizing the OPRF primitive in a way that can serve our application is not trivial. Indeed, the intuitive definition of OPRF [17, 20] as the secure computation of a two-party functionality which on input pair (k, x) returns an output pair $(\perp, f_k(x))$, is limiting for at least three reasons: (1) It does not imply security when several OPRF instances are executed concurrently, as is the case in our PPSS construction; (2) It does not apply to our setting where the existence of authenticated channels cannot be assumed; and (3) It is not clear how to instantiate such functionality in the concurrent setting without on-line extractable zero-knowledge proofs of knowledge, which would add a significant overhead to any OPRF instantiation.

We overcome these issues via a novel formalization of the *verifiable* version of the OPRF primitive, V-OPRF, as an ideal functionality in the Universal Composability (UC) framework [10] for which we show several very efficient instantiations. Expressing V-OPRF in the UC framework is a delicate task, especially in the setting of interest to us where there are no authenticated channels. Our formalism enforces that the server who generates the PRF key k also produces a function descriptor π, which fixes a deterministic function f_π. (For honest servers, π is a commitment to k and the fixed function f_π is equal to the PRF

f_k.) Then, in any (non-rejecting) execution of the V-OPRF protocol executed given the function descriptor π, the V-OPRF functionality verifies that the user's output is computed as $f_\pi(x)$. In other words, the V-OPRF functionality ensures *consistency* between V-OPRF instances executed under the same function descriptor π as well as *verifiability* that the output value is computed using the committed function f_π.

Our UC V-OPRF formalization bears interesting similarities to UC blind signatures [25, 15, 1]. In a nutshell, instead of on-line extraction of argument x from the (potentially malicious) client in every V-OPRF instance, a V-OPRF functionality issues a *ticket* for every instance executed under a given descriptor π. The user (or adversary) can then use these tickets to evaluate function f_π on inputs of their choice, but with the constraint that m tickets cannot be used to compute f_π values on more than m distinct inputs. Given this similarity, we observe that an efficient realization of V-OPRF can be achieved (in ROM) by hashing a deterministic blind signature-message pair.

We obtain three highly efficient variants of this design strategy, which provide three *single-round* V-OPRF instantiations in ROM, and we prove them UC-secure under "one-more" type of assumptions [3, 21]. Specifically, we show such V-OPRF instantiations in ROM under a one-more Gap DH assumption on any group of prime order, a similar assumption on the group with a bilinear map, and a one-more RSA assumption. We also provide an efficient *standard model* V-OPRF construction for the Naor-Reingold PRF [30], based on the honest-but-curious OPRF protocol given by [17]. This protocol has four messages and is secure under Strong-RSA and the Decisional Composite Residuosity (DCR) assumptions. A single round standard-model (CRS) protocol is possible too but at a significant higher computational complexity.

We note that the UC formalization of the Verifiable Oblivious PRF functionality that is at the core of our security treatment is likely to have applications beyond this work. Indeed, OPRF's have been shown to be useful in a variety of scenarios, including Searchable Symmetric Encryption (SSE) schemes, e.g. [13, 11], and secure two-party computation of Set Intersection [17, 20, 21].

PPSS. Our PPSS protocol is *password-only* in the Common Reference String (CRS) model, i.e. the user needs no other inputs than her password and a CRS string defining a non-malleable commitment scheme instance which can be part of the user's V-OPRF software. Our PPSS protocol is *single-round* in the hybrid model where parties can access the V-OPRF functionality. Given the V-OPRF instantiations discussed above, this implies three different instantiations of a single-round (i.e. two-message) PPSS schemes in ROM based on different one-more type of assumptions, and a four-message PPSS scheme in the CRS model.

Our PPSS construction follows the strategy of the early protocols of Ford and Kaliski [16] and Jablon [18] who treated the case of $t = n$: Secret-share the secret sc into shares $(s_1, \ldots, s_n)$, let each server S_i pick key k_i for a PRF f, and let $\mathbf{c} = (e_1, \ldots, e_n)$ where e_i is an encryption of s_i under $\rho_i = f_{k_i}(\mathsf{pw})$. Each server S_i stores $(k_i, \mathbf{c})$, and in the reconstruction protocol the user re-computes each ρ_i via an instance of a V-OPRF protocol with each server S_i on its input

pw and S_i's input k_i. If the user also gets string $\mathbf{c}$ from the servers, the user can decrypt shares s_i using the ρ_i's and interpolate these shares to reconstruct sc. The first thing to note is that ciphertexts e_i must *not* be committing to the encryption key ρ_i. Otherwise, an adversary could test a password guess pw^* in an interaction with a single V-OPRF instance (instead of requiring $t + 1$ interactions with $t + 1$ different servers as our security notion imposes on the attacker), by computing $\rho_i^* = f_{k_i}(\mathsf{pw}^*)$ and testing if decryption of e_i under ρ_i^* returns a plausible share value. We prevent such tests by sharing sc over a binary extension field $\mathbb{F} = GF(2^\ell)$, choosing a PRF f which maps onto ℓ-bit strings, and setting e_i to $s_i \oplus f_{k_i}(\mathsf{pw})$. Secondly, the above simple protocol can allow a malicious server S_i to find the user's password pw if S_i is not forced to use the same function f_{π_i} in each V-OPRF instance. Consider the OPRF protocol of [17] for the Naor-Reingold PRF $f_{k_i}(x) = g^v$ where $v = k_{i,0} \cdot \prod_{x_j=1} k_{i,j}$ for $k_i = (k_{i,0}, \ldots, k_{i,\ell})$ [30]. If in some PPSS instance, a misbehaving S_i uses key k_i' which differs from k_i on one index j, i.e. in one component $k_{i,j}$, S_i can conclude that the j-th bit of pw is 0 if the user recovers its secret correctly from such PPSS instance. Note that the adversary can learn whether user's secret is reconstructed correctly by observing any higher-level protocol which uses this secret, e.g. a T-PAKE protocol discussed below. We counter this attack by using the verifiability property of our V-OPRF functionality, which ensures that S_i computes the function committed in π_i, and by extending the user-related information stored by each server to $\omega = (\boldsymbol{\pi}, \mathbf{c}, C)$ where $\boldsymbol{\pi}$ is a vector of function descriptors $\pi_1, \ldots, \pi_n$ of each server, and C is a non-malleable commitment to the values $\boldsymbol{\pi}$, $\mathbf{c}$ and user's password pw. This commitment is the basis for ensuring that the on-line attacker playing the role of the servers can test at most one password guess per one reconstruction protocol instance. Note that the described solution requires $O(n)$ storage and bandwidth per server, but it is straightforward to reduce these to $O(\log n)$ using a Merkle tree commitment.

With an instantiation of V-OPRF in ROM we achieve a remarkably efficient reconstruction protocol without relying on PKI or secure channels. The user runs an optimal 2-message protocol with $t+1$ (or more) servers, and in the case of our V-OPRF construction based on the one-more Gap DH assumption, the protocol involves just 2 exponentiations by the server and a total of $2t+3$ multi-exponentiations for the user, employing the optimized ROM-based NIZK for discrete logarithm equality of [12], plus a few inexpensive operations. The (one-time) initialization stage is also very efficient, involving $2n + 1$ exponentiations for the user and 3 exponentiations per each server. Note that there is no inter-server communication in the protocol and that the user can communicate with each server independently, so it can be done in parallel and/or in any order without the servers being aware of each other. Moreover, the user can initiate the V-OPRF protocol with more than $t+1$ servers, and it will reconstruct secret sc as long as $t + 1$ contacted servers reply with correct triple $\omega = (\boldsymbol{\pi}, \mathbf{c}, C)$ and complete the V-OPRF instances on function descriptors π_i in $\boldsymbol{\pi}$.

Our PPSS protocol in the password-only setting enjoys the following security hedging property: While avoiding the need to rely on the authenticity of servers' public keys held by the user is an important security property, when such public keys are available they can add significant security, because they render on-line attacks against a user ineffective and strengthen the security and the soundness properties of the PPSS scheme. Thus, to get the benefits of both worlds, both with and without the correctly functioning PKI, running a password-only PPSS protocol over PKI-authenticated links achieves the following: If the user has the correct servers' public keys, she gets the additional security benefits stated above, otherwise, if some or all of the public keys are either incorrect or missing, she still enjoys the security of the password-only setting.

T-PAKE. When composed with regular key exchange protocols, our PPSS scheme leads to the most efficient T-PAKE protocols to date even when compared to protocols that assume that the user carries a public key that it can use to authenticate the servers. Figure 1 summarizes the state of the art in T-PAKE protocols and how our protocols compare to this prior work. Interestingly, while there is a large body of work on single-server PAKE protocols (e.g. [4, 23, 24, 5]) that has produced remarkable schemes, including one-round password-only protocols in the standard model, threshold PAKE has seen less progress, with most protocols showing disadvantages over a single-server PAKE. In particular, before our work, no single-round (t, n)-PAKE protocol was known, not even in the ROM and assuming PKI. Most protocols assume a public key carried by the client (making them non password-only) and all assume secure channels (or PKI) between servers. Even in the $n = t = 2$ case no one-round protocol was known, and all previously known protocols for this case require inter-server secure channels. Our work improves on these parameters achieving the best known properties in *all* the aspects reflected in the table.

In particular, *we achieve single-round password-only protocol in the CRS and ROM models for arbitrary (t, n) parameters with no PKI requirements for any party and no inter-server communication* (secure communication is only assumed when a user first registers with the servers). In addition, the protocol is computationally very efficient (and more so than any of the previous protocols, even for the (2,2) case). We also exhibit a password-only standard-model implementation of our scheme requiring two rounds of communication (4 messages in total) between client and servers. Our T-PAKE protocols are built by combining (existing) suitable key exchange protocols on top of our V-OPRF-based PPSS scheme. We prove T-PAKE security via a generic composition theorem showing the security of any such composed protocol.

Organization. In Section 2 we present the formalization of the V-OPRF functionality in the UC setting. In Section 3 we show an efficient realization of this functionality in the random oracle model (ROM) (further ROM and standard (CRS) model constructions are presented in the full version [19]). In Section 4 we define and formalize PPSS in the password-only model. In Section 5 we present an efficient PPSS realization which relies on the V-OPRF functionality. Finally, in Section 6 we consider T-PAKE schemes obtained by composing a PPSS scheme with a regular key-exchange protocol, and present a full specification of our most efficient instantiation of the PPSS and T-PAKE protocols.

scheme	$(t+1,n)$	ROM/std	client	inter-server	msgs	total comm.	comp. C | S
BJKS [6]	$(2,2)$	ROM	PKI	PKI	7	O(1)	O(1)
KMTG [22]	$(2,2)$	Std/ROM	CRS	sec.chan.	≥ 5	O(1)	O(1)
CLN [9]	$(2,2)$	Std/ROM	CRS	PKI	8	O(1)	O(1)
DRG [14]	$t<n/3$	Std	CRS	sec.chan.	≥ 12	$O(n^3)$	$O(1) \mid O(n^2)$
MSJ [27]	any	ROM	PKI	PKI	7	$O(n^2)$	$O(1) \mid O(n)$
BJSL [2]	any	ROM	PKI	PKI	3	$O(t)$	$8t+17 \mid 16$
CLLN [7]	any	ROM	CRS	PKI	10	$O(t^2)$	$14t+24 \mid 7t+28$
Our PPSS1	any	ROM	CRS	none	2	$O(t \log n)$	$2t+3 \mid 2$
Our PPSS2	any	Std	CRS	none	4	$O(\ell t \log n)$	$O(t\ell) \mid O(l)$

Fig. 1. Comparison between PPSS/T-PAKE schemes. "PPSS1" and "PPSS2" refer to our PPSS scheme of Section 5 with V-OPRF instantiated, respectively, with protocol 2HashDH-NIZK of Section 3 (this instantiation is shown in Figure 5) and with protocol NR-V-OPRF (deferred to the full version [19]). The "total comm." column counts the number of transmitted group elements and other objects of length polynomial in the security parameter, like public-key signatures. Variable ℓ denotes the length of the password string (or its hash). The last column counts (multi)exponentiations in a prime-order group (except for our PPSS2 where exponentiations are modulo a Paillier modulus) performed by the client and each server in the reconstruction protocol. All costs in the last four rows refer to an optimistic scenario with no adversarial interference. With worst-case adversarial interference, for BJSL and CLLN all costs grow by the factor of $n-t$, while for our schemes costs grow by the factor of $\lfloor n/(t+1) \rfloor$.

This version of the paper omits many details, constructions and proofs; please refer to the full version [19] for a complete presentation.

2 Functionality $\mathcal{F}_{\mathbf{VOPRF}}$

The $\mathcal{F}_{\text{VOPRF}}$ functionality can be thought of as a collection of tables that are indexed by "labels" denoted by the function parameters π. Users may obtain values from these tables on inputs x of their choice without leaking any information about these inputs (and corresponding outputs) to the adversary. $\mathcal{F}_{\text{VOPRF}}$ generates these tables dynamically and fills them with random values on demand. Each table is associated by the functionality with a specific sender. In addition to the tables registered to honest senders, the adversary is allowed to register with $\mathcal{F}_{\text{VOPRF}}$ its own tables. Interacting with an adversary-registered table does not jeopardize the privacy of the user's input but naturally $\mathcal{F}_{\text{VOPRF}}$ will provide no pseudorandomness guarantee for the output derived from such tables. However, $\mathcal{F}_{\text{VOPRF}}$ will ensure that all adversarial tables are completely determined according to a deterministic function that is committed by the adversary at the time of the table's initialization in the form of a circuit M.

A major consideration in our definition of $\mathcal{F}_{\text{VOPRF}}$ is to avoid the need for input extractability (from dishonest users) in the real-world realizations of the functionality. Such need is common in UC-defined functionalities but in our

Functionality $\mathcal{F}_{\text{VOPRF}}$

Key generation:

On message $(\textsc{KeyGen}, sid)$ from S, forward $(\textsc{KeyGen}, sid, S)$ to adversary $\mathcal{A}^*$.

On message $(\textsc{Parameter}, sid, S, \pi, M)$ from $\mathcal{A}^*$, ignore this call if $\mathsf{param}(S)$ is already defined. Otherwise, set $\mathsf{param}(S) = \langle\pi\rangle$ and initialize $\mathsf{tickets}(\pi) = 0$, and $\mathsf{hist}(\pi)$ to the empty string. If S is honest send $(\textsc{Parameter}, sid, \pi)$ to party S, else parse M as a circuit with ℓ-bit output and insert (π, M) in $\mathsf{CorParams}$.

V-OPRF evaluation:

On message $(\textsc{Eval}, sid, S, x)$ from party U for sender S, record (U, x) and forward $(\textsc{Eval}, sid, U, S)$ to $\mathcal{A}^*$.

On message $(\textsc{SenderComplete}, sid, S)$ from $\mathcal{A}^*$ for some honest S output $(\textsc{SenderComplete}, sid)$ to party S and set $\mathsf{tickets}(\pi) = \mathsf{tickets}(\pi) + 1$ for π s.t. $\langle\pi\rangle = \mathsf{param}(S)$.

On message $(\textsc{UserComplete}, sid, U, \pi, \mathsf{flag})$ from $\mathcal{A}^*$, recover (U, x) and:

- If $\mathsf{flag} = \top$ and $\langle\pi\rangle = \mathsf{param}(S)$ for an honest S then: If $\mathsf{tickets}(\pi) \leq 0$ ignore the $\textsc{UserComplete}$ request of $\mathcal{A}^*$. Otherwise: (1) if $\mathsf{hist}(\pi)$ includes a pair $\langle x, \rho'\rangle$, set $\rho = \rho'$, else sample ρ at random from $\{0,1\}^\ell$ and enter $\langle x, \rho\rangle$ into $\mathsf{hist}(\pi)$; (2) Set $\mathsf{tickets}(\pi) = \mathsf{tickets}(\pi) - 1$ and output $(\textsc{Eval}, \pi, \rho)$ to party U.
- Else, if $\mathsf{flag} = \bot$ then return $(\textsc{Eval}, \pi, \bot)$ to U.
- Else, if $\mathsf{flag} = \top$ and π is such that $(\pi, M) \in \mathsf{CorParams}$ for some circuit M, compute $\rho = M(x)$, enter $\langle x, \rho\rangle$ in $\mathsf{hist}(\pi)$, output $(\textsc{Eval}, \pi, \rho)$ to party U.

Fig. 2. Verifiable Oblivious PRF functionality $\mathcal{F}_{\text{VOPRF}}$.

case it would disqualify the more efficient instantiations of $\mathcal{F}_{\text{VOPRF}}$ presented here. Thus, instead of resorting to input extraction requirements, we define a "ticket mechanism" that increases a ticket upon function evaluation at a sender and decreases it when this value is computed at the user (or the adversary). The functionality guarantees that tickets remain non-negative, namely, for any function parameter π registered with a honest sender S, the number of inputs on which users compute the function π is no more than the number of evaluations of the function at S.

Another important aspect of our $\mathcal{F}_{\text{VOPRF}}$ formalism is the way we handle the 1-1 relationship between a sender S and its function parameter π, where S is used to identify a sender and π describes this sender's committed function. The unique sender-function binding that is known to the functionality cannot be enforced in a real-world setting where users cannot validate such a binding as is the case when no authenticated channels (or other forms of authenticated information) are available to the user. Since these settings are common in our applications, we define $\mathcal{F}_{\text{VOPRF}}$ so that the user can provide a name of a sender whose function it intends to compute but the result returned to the user applies a function π determined by the attacker, and is possibly different than the function associated to the requested S.

Parameters:

Generator g of cyclic group of order m, hash functions $H_1(\cdot), H_2(\cdot), H_3(\cdot)$.

Key Generation:

On $(\textsc{KeyGen}, sid)$, pick $k \in_R \mathbb{Z}_m$, set $y = g^k$, return $(\textsc{Parameter}, sid, y)$.

V-OPRF Evaluation:

On message $(\textsc{Eval}, sid, S, x)$, pick $r \in_R \mathbb{Z}_m$ and send $a = H_1(x)^r$ to S.

On message a from network entity U, check if $a \in \langle g \rangle$, compute $b = a^k$ and $\zeta = \mathtt{NIZK}_{\mathtt{EQ}}^{H_3}[g, y, a, b]$, and send $\langle y, b, \zeta \rangle$ to U.

On message $\langle y, b, \zeta \rangle$ from party S, verify the $\mathtt{NIZK}$ ζ and $b \in \langle g \rangle$. If the tests pass return $(\textsc{Eval}, y, H_2(y, x, b^{1/r}))$, else return $(\textsc{Eval}, y, \perp)$.

Fig. 3. Protocol 2HashDH-NIZK.

In spite of the above, note that if $\mathcal{F}_{\text{VOPRF}}$ is used in a context where the user knows a-priori a correspondence between S and π, the user can reject responses that are not consistent with it. We make essential use of this capability in our applications. Finally, note that $\mathcal{F}_{\text{VOPRF}}$ guarantees that value ρ obtained by the user is in the table π even though such table may not have been the user's original target. This provides $\mathcal{F}_{\text{VOPRF}}$ with a verifiability property which is verifier-dependent and may not be transferable to others; in particular, it is a weaker guarantee than the verifiability propery of verifable random functions [29].

3 Efficient Realization of $\mathcal{F}_{\text{VOPRF}}$

We present a class of constructions for realizing $\mathcal{F}_{\text{VOPRF}}$ in the random oracle model. Our constructions share the following general structure: the receiver hashes and blinds her input and requests the sender's secret-key application on this blinded value. The receiver verifies the sender's response and then obtains the V-OPRF output by applying a second hash function. Due to the double hashing (which is essential in the security proof) we term the constructions with the "2Hash" prefix. For lack of space we present here only a single instance of this design methodology, protocol 2HashDH-NIZK in Figure 3, while our further ROM constructions, based on RSA or a group with a bilinear map, are presented in the full version [19]. Protocol 2HashDH-NIZK uses a non-interactive zero-knowledge proof $\mathtt{NIZK}_{\mathtt{EQ}}^{H_3}[g, y, a, b]$ of discrete logarithm equality $\mathsf{DL}(g, y) = \mathsf{DL}(a, b)$. This NIZK has to be straight-line simulatable and simulation sound, and it can be implemented with one multi-exponentiation for both the prover and the verifier using hash function H_3 modeled as a random oracle [12].

We will argue the security of the construction employing the following assumption: the (N, Q) One-more Gap DH assumption, states that for any PPT $\mathcal{A}$ it holds that the following probability is negligible:

$$\mathsf{Prob}[\mathcal{A}^{(\cdot)^k, \mathsf{DDH}(\cdot,\cdot,\cdot,\cdot)}(g, g^k, g_1, \ldots, g_N) = \{(g_{j_s}, g_{j_s}^k) \mid s = 1, \ldots, Q+1\}]$$

where Q is the number of queries that $\mathcal{A}$ poses to the $(\cdot)^k$ oracle. The probability is taken over all choices of $g^k, g_1, \ldots, g_N$ which are assumed to be random elements of $\langle g \rangle$. We denote by $\epsilon_{\mathsf{omdh},G}(N, Q)$ the maximum advantage of any PPT adversary against the assumption.

Theorem 1. *The 2HashDH-NIZK protocol over group G of order m UC-realizes $\mathcal{F}_{\text{VOPRF}}$ per Fig. 2 in ROM assuming (i) the existence of PRF functions, (ii) the (N, Q) One-More Gap DH assumption on G where Q is the number of V-OPRF executions and $N = Q + q_1$ where q_1 is the number of $H_1(\cdot)$ queries.*

More precisely, for any adversary against 2HashDH-NIZK there is an ideal-world adversary (simulator) that produces a view that no environment can distinguish with advantage better than $q_S \cdot \epsilon_{\mathsf{omdh},G}(N, Q) + q_3/m^2 + 2 \cdot q_U/m + N^2/m + \epsilon_{\mathsf{PRF}}(q_2)$ where q_S is the number of senders, q_U the number of users, q_2, q_3 are the number of queries to oracles H_2, H_3, and $\epsilon_{\mathsf{PRF}}(q_2)$ is the security of the PRF function against adversaries executing in comparable time and posing q_2 queries.

Proof. See full version [19].

Remark. We note that in the construction of Figure 3, the outgoing message that is constructed given an $(\textsc{Eval}, sid, S, x)$ command is independent of S. It is easy to see that security is preserved even if the user employs the same outgoing message for any sequence of consecutive $(\textsc{Eval}, sid, S_1, x), \ldots, (\textsc{Eval}, sid, S_n, x)$ commands for any $n \geq 1$. We make essential use of this feature in the optimized protocol of Figure 5 where the user uses the same blinded input with all servers.

V-OPRF Constructions in the Standard Model. A 4-message realization of V-OPRF in the standard model based on the Strong RSA and DCR assumptions is presented in the full version [19].

4 Password-Protected Secret Sharing: Definitions

Our definition of PPSS adapts the PPSS notion of [2] to the CRS model, but also re-defines PPSS in terms of a *key derivation mechanism* rather than an encryption-style notion used in [2]. In other words, rather than used directly to semantically protect any message, a PPSS will generate and protect a random key. This change allows for better modularity, because the resulting key can be used not only for message encryption (and authentication), but also e.g. for an Authenticated Key Exchange. A Password-Protected Secret Sharing (PPSS) scheme in the CRS model is a protocol involving $n+1$ parties, a user U, and n servers $\mathsf{S}_1, \ldots, \mathsf{S}_n$. A PPSS scheme is a tuple $(\mathsf{ParGen}, \mathsf{SKeyGen}, \mathsf{Init}, \mathsf{Rec})$, where ParGen and $\mathsf{SKeyGen}$ are randomized algorithms and Init and Rec are multi-paty protocols with the following syntax:
1) Algorithm ParGen generates string CRS for a given security parameter τ.
2) Each S_i runs $\mathsf{SKeyGen}(\mathsf{CRS})$ to generate private state σ_i and public param. π_i.
3) Protocol Init is executed by U and servers $\mathsf{S}_1, \ldots, \mathsf{S}_n$, where U runs algorithm $\mathsf{U}_{\mathsf{Init}}$ on inputs a password $\mathsf{pw} \in \{0,1\}^\tau$, global parameters CRS, and a vector of server's public parameters $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$, while each S_i runs algorithm $\mathsf{S}_{\mathsf{Init}}$ on

input $(\mathsf{CRS}, \sigma_i, \pi_i)$. The outputs of Init is a τ-bit key K for U and a user-specific information ω_i for each server S_i.
4) Protocol Rec is executed by U and servers $\mathsf{S}_1, \ldots, \mathsf{S}_n$, where U runs algorithm $\mathsf{U}_{\mathsf{Rec}}$ on $(\mathsf{CRS}, \mathsf{pw})$, and each S_i runs algorithm $\mathsf{S}_{\mathsf{Rec}}$ on $(\mathsf{CRS}, \sigma_i, \pi_i, \omega_i)$. Protocol Rec generates no output for the servers, while U outputs K' which is either a τ-bit string or a rejection symbol $\perp$.

The *correctness* requirement is that Rec returns the same key K which was generated in Init, i.e. that for any τ, any CRS output by $\mathsf{ParGen}(1^\tau)$, any (σ_i, π_i) output by n instances of $\mathsf{SKeyGen}(\mathsf{CRS})$, and any $\mathsf{pw} \in \{0,1\}^\tau$, if $(K, \omega_1, \ldots, \omega_n)$ is the vector of outputs of Init executed on inputs $(\mathsf{pw}, \mathsf{CRS}, \boldsymbol{\pi})$ for U and $(\mathsf{CRS}, \sigma_i, \pi_i)$ for each S_i, then U's local output in an instance of Rec executed on inputs $(\mathsf{CRS}, \mathsf{pw})$ for U and $(\mathsf{CRS}, \sigma_i, \pi_i, \omega_i)$ for each S_i, is equal to K.

Server's User-Related State. We stress that the state $(\sigma_i, \pi_i, \omega_i)$ of each server S_i is stored *for each user separately*, and the PPSS security notion we define below assumes that each S_i stores a separate $(\sigma_i, \pi_i, \omega_i)$ tuple for each user account. Indeed, the security of the PPSS protocol we present in Section 5 would be decreased if S_i re-uses the same OPRF key, stored in σ_i in this PPSS protocol, across multiple user accounts. (Technically, the adversary would get additional oracle access to the same $\mathsf{S}^\diamond_{\mathsf{Rec}}$ oracle, see below, for each user account on which the server re-uses the same (σ_i, π_i) pair.) Consequently, if S_i wants to provide PPSS service to multiple users, it has to generate a separate (σ_i, π_i) pair for each user (these per-user keys can be derived internally by S_i using a PRF and a global PRF key applied to a user's identifier).

Security. We define security of a PPSS scheme in terms of adversary's advantage in distinguishing the key K output by U from a random string. We assume that the adversary sees CRS and the vector of server's public parameters $\boldsymbol{\pi}$ used in the initialization instance, as well as the private states $\boldsymbol{\sigma}_{\mathsf{B}} \triangleq \{\sigma_i\}_{i \in \mathsf{B}}$ and $\boldsymbol{\omega}_{\mathsf{B}} \triangleq \{\omega_i\}_{i \in \mathsf{B}}$ for some set B of corrupted servers, and that it has concurrent oracle access to instances of $\mathsf{U}_{\mathsf{Rec}}(\mathsf{CRS}, \mathsf{pw})$ and $\mathsf{S}_{\mathsf{Rec}}(\mathsf{CRS}, \sigma_i, \pi_i, \omega_i)$, for i in $\overline{\mathsf{B}} \triangleq \{1, ..., n\} \setminus \mathsf{B}$. We denote as $\mathsf{U}^\diamond_{\mathsf{Rec}}(\mathsf{CRS}, \mathsf{pw}, b, K^{(0)})$ an oracle which executes the interactive algorithm $\mathsf{U}_{\mathsf{Rec}}(\mathsf{CRS}, \mathsf{pw})$, and when this algorithm terminates with a local output K, the $\mathsf{U}^\diamond_{\mathsf{Rec}}$ oracle (re-)sets K to $K^{(0)}$ if $b = 0$ and $K \neq \perp$, and then returns K to the caller. However, if $b = 1$ or $K = \perp$ then the caller receives the unmodified value K (to which we will refer as $K^{(1)}$) as it was output by the $\mathsf{U}_{\mathsf{Rec}}$ instance. We denote as $\mathsf{S}^\diamond_{\mathsf{Rec}}(\mathsf{CRS}, \boldsymbol{\sigma}_{\overline{\mathsf{B}}}, \boldsymbol{\pi}, \boldsymbol{\omega}_{\overline{\mathsf{B}}})$ an oracle which on input $i \in \overline{\mathsf{B}}$ executes the interactive algorithm $\mathsf{S}_{\mathsf{Rec}}(\mathsf{CRS}, \sigma_i, \pi_i, \omega_i)$.

Intuitively, we should call an (t, n)-threshold PPSS scheme secure if for any password dictionary D, if pw is randomly chosen in D then the adversary's advantage in distinguishing the PPSS-protected key K from a random string (i.e., guessing b) is at most negligibly above $1/|D|$, the probability of guessing the password, times $q_u + \lfloor q_s/(t - t' + 1) \rfloor$, where q_u and q_s are the numbers, respectively, of the $\mathsf{U}_{\mathsf{Rec}}$ and $\mathsf{S}_{\mathsf{Rec}}$ protocol instances the adversary can interact with, and $t' \leq t$ is the number of corrupted servers. Factor $1/|D| \cdot \lfloor q_s/(t - t' + 1) \rfloor$ corresponds to an inherent vulnerability due to on-line dictionary attacks: An adversary who learns the shares of $t' \le t$ servers can test any password $\tilde{\mathsf{pw}}$ in D by running the user's protocol on $\tilde{\mathsf{pw}}$ interacting with $t - t' + 1$ uncorrupted servers. Factor $1/|D| \cdot q_u$ corresponds to an inherent vulnerability of password-authenticated protocols in the CRS model, because the adversary can run the initialization protocol Init on a password guess $\tilde{\mathsf{pw}}$ and then run the servers' protocol interacting with the user: If the user does not reject (by outputting $\perp$), the adversary can conclude that $\tilde{\mathsf{pw}} = \mathsf{pw}$.

To make the PPSS notion easier to use in applications it is important that the adversary sees the key-pseudorandomness challenge, either a real key or a random key, already after the initialization protocol Init, rather than only when this key is reconstructed in protocol Rec. (E.g. our T-PAKE constructions rely on this property.) To make sure that the PPSS-protected key remains pseudorandom in each key usage, whether after the initialization or after each reconstruction instance, we let the adversary see the key generated by Init as well as the key(s) output by every Rec instance. That is, at the end of Init and after each Rec instance the attacker is given $K^{(0)}$ if $b = 0$, but if $b = 1$ then the attacker is given the actual value of the key output by, respectively, $\mathsf{U}_{\mathsf{Init}}$ or $\mathsf{U}_{\mathsf{Rec}}$. Note that $K^{(0)}$ does not change across different reconstructions because it is fixed at the start of the experiment, while $K^{(1)}$ is determined by the actual outputs of $\mathsf{U}_{\mathsf{Init}}$ and $\mathsf{U}_{\mathsf{Rec}}$ instances. Importantly, note that this definition implies that in the real execution the reconstruction instances must output the same key that was created in the initialization or the attacker can trivially guess b. We further discuss this *soundness* property below.

Definition 1. *A PPSS scheme is* (T, q_u, q_s, ϵ)-secure *(for fixed threshold parameters* (t, n) *if for any* $D \subseteq \{0,1\}^\tau$*, any set* $\mathsf{B} \subseteq \{1, ..., n\}$ *of size* $t' \le t$*, and any algorithm* $\mathcal{A}$ *with running time* T*, we have*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{ppss}} \le \left(q_u + \left\lfloor \frac{q_s}{t - t' + 1} \right\rfloor \right) \cdot \frac{1}{|D|} + \epsilon \qquad (1)$$

where $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{ppss}} = |p_{\mathcal{A}}^{(1)} - p_{\mathcal{A}}^{(0)}|$ *and* $p_{\mathcal{A}}^{(b)} = \Pr[b' = 1]$ *in a game below, for* $b \in \{0,1\}$*:*
(1) Choose pw *at random in* D*, generate* $\mathsf{CRS} \leftarrow \mathsf{ParGen}(1^\tau)$ *and* $(\sigma_i, \pi_i) \leftarrow \mathsf{SKeyGen}(\mathsf{CRS})$ *for* $i \in \overline{\mathsf{B}}$*. Give* CRS *and* $\{\pi_i\}_{i \in \overline{\mathsf{B}}}$ *to* $\mathcal{A}$ *and let* $\mathcal{A}$ *generate* $\{\pi_i\}_{i \in \mathsf{B}}$*.*
(2) Run an instance of Init *between* U*, which executes protocol* $\mathsf{U}_{\mathsf{Init}}(\mathsf{CRS}, \mathsf{pw}, \pi_1, ..., \pi_n)$*, and the servers, where each* S_i *for* $i \in \overline{\mathsf{B}}$ *executes protocol* $\mathsf{S}_{\mathsf{Init}}(\mathsf{CRS}, \sigma_i, \pi_i)$*, while servers* S_i *for* $i \in \mathsf{B}$ *are controlled by adversary* $\mathcal{A}$*. The protocol proceeds on public channels, with* $\mathcal{A}$ *playing a man-in-the-middle on all communications. Denote* U*'s output in this* Init *instance as* $K^{(1)}$*, and denote* S_i*'s output, for* $i \in \overline{\mathsf{B}}$*, as* ω_i*. Choose* $K^{(0)}$ *at random in* $\{0,1\}^\tau$*. Give key* $K^{(b)}$ *to* $\mathcal{A}$*.*
(3) Let $\mathcal{A}$ *interact with* q_u *instances of* $\mathsf{U}^{\diamond}_{\mathsf{Rec}}(\mathsf{CRS}, \mathsf{pw}, b, K^{(0)})$ *and* q_s *instances of* $\mathsf{S}^{\diamond}_{\mathsf{Rec}}(\mathsf{CRS}, \boldsymbol{\sigma}_{\overline{\mathsf{B}}}, \boldsymbol{\pi}, \boldsymbol{\omega}_{\overline{\mathsf{B}}})$*. Let* b' *be the final output of* $\mathcal{A}$*.*

Secure Initialization. Note that in the above definition we assume that in the initalization protocol U runs the $\mathsf{U}_{\mathsf{Init}}$ procedure on input a vector of public parameters $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ where π_i for each $i \in \overline{\mathsf{B}}$ is the true output of

$\mathsf{SKeyGen}$ executed by server S_i. In other words, we assume that the user runs the initialization procedure on correct (i.e. authentic) values π_i for the honest servers. This is equivalent to assuming that the user can authenticate, e.g. via the PKI, the servers with whom it wants to initialize the PPSS scheme. The requirement of authenticated channels between the user and the honest servers *during the initialiation protocol* is indeed necessesary, or otherwise the adversary would be able to pose as $t+1$ servers among $\mathsf{S}_1, \ldots, \mathsf{S}_n$ and recover U's secret sc from the initialization protocol. (A similar assumption on authenticity of servers' public keys in the initialization is also made in [8].)

Soundness. The above definition captures also a *soundness* property of a PPSS scheme, because it implies an upper-bound on the probability that an adversary causes any $\mathsf{U}_{\mathsf{Rec}}$ instance to output $K' \notin \{K, \perp\}$ where K was an output of $\mathsf{U}_{\mathsf{Init}}$. Assume algorithm $\mathcal{A}$ which outputs 0 if every key returned by $\mathsf{U}^{\diamond}_{\mathsf{Rec}}$ oracle is either equal to $K^{(b)}$ which was output by $\mathsf{U}_{\mathsf{Init}}$, or to $\perp$. Note that in the security experiment with $b = 0$, oracle $\mathsf{U}^{\diamond}_{\mathsf{Rec}}$ always returns $K^{(0)}$ or $\perp$, so $p^{(0)}_{\mathcal{A}} = 0$. The security definition implies that $p^{(1)}_{\mathcal{A}} \leq \left(q_u + \left\lfloor \frac{q_s}{t-t'+1} \right\rfloor\right) \cdot \frac{1}{|D|} + \epsilon$, hence this is also an upper-bound on the probability that any $\mathsf{U}_{\mathsf{Rec}}$ instance outputs K' which is neither $\perp$ nor K output in $\mathsf{U}_{\mathsf{Init}}$.

Robustness. Another desirable property of a PPSS scheme is robustness, which we define as the requirement that the user reconstructs the key created in Init as long as it communicates without obstructions with at least $t+1$ non-corrupt servers and with at most t corrupt ones. This property is distinct from soundness in that it assumes that the adversary lets the user communicate with $t+1$ non-corrupt servers without interference. Note that this implies that the number t number of corrupt servers satisfies $t < n/2$, a restriction which is not imposed by either the security or the soundness properties.

5 A PPSS protocol in the $\mathcal{F}_{\mathrm{VOPRF}}$-hybrid world

We show a PPSS protocol based on any realization of the $\mathcal{F}_{\mathrm{VOPRF}}$ functionality. The protocol is shown in Figure 4 in the $\mathcal{F}_{\mathrm{VOPRF}}$-hybrid model (a specific instantiation based on the 2HashDH-NIZK V-OPRF of Fig. 3 is shown in Fig. 5). The protocol is secure in the CRS model and it assumes a pseudorandom generator and a computationally hiding, computationally binding, and non-malleable (with respect to decommitment) commitment scheme, which can be realized e.g. by a CCA-secure public key encryption, or by hashing the message together with a random nonce in ROM. To provide rationale for our design we first consider a subset of the protocol in Figure 4 and then show the necessity of some additional elements. In $\mathsf{SKeyGen}$, each server S_i picks its public parameter π_i as the V-OPRF function descriptor, which in all our V-OPRF instantiations is a commitment to the private key of the underlying PRF (see Section 2). In protocol Init, on U's inputs a password pw and a vector of function descriptors $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$, which are authentically delivered to the user, user U picks a random key K, secret-shares it into shares $s_1, \ldots, s_n$, and then encrypts each s_i using one-time pad encryption under key $\rho_i = F_{\pi_i}(\mathsf{pw})$, computed in a V-OPRF instance with server S_i. The vector of function descriptors $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ and the ciphertexts $\mathbf{c} = (c_1, \ldots, c_n)$, where $c_i = s_i \oplus F_{\pi_i}(\mathsf{pw})$, is public, and given to each server. At reconstruction the servers send these two vectors to U, who can recover $t+1$ shares s_i, and interpolate them to recover K, after $t+1$ V-OPRF instances in which U recomputes the values $\rho_i = F_{\pi_i}(\mathsf{pw})$ for $t+1$ different i's.

This simplified protocol, however, is not secure. As we explain in the introduction, if the attacker learns whether the receiver recovers the shared key K correctly, the above protocol would enable a malicious server S_i to get information about user's password pw (including recovering it completely using binary search in an OPRF based on the Naor-Reingold PRF), by manipulating the function descriptor π_i in each OPRF instance executed by S_i in this reconstruction protocol. In fact, in the PPSS security model defined in section 4, the above simplified protocol allows a malicious server to recover π through an off-line dictionary search after a single instance of PPSS reconstruction. Note that our PPSS security model reveals the whole key K output in a PPSS reconstruction to the adversary, which models putting this key to an arbitrary usage by the higher-level protocol, e.g. by the T-PAKE scheme built from PPSS in Section 6. Now, if $\mathcal{A}$ sends to $\mathsf{U}_{\mathsf{Rec}}$ a vector of function descriptors π_i^* which correspond to PRF keys k_i^* which $\mathcal{A}$ creates, and if $\mathcal{A}$ learns the key K output by this $\mathsf{U}_{\mathsf{Rec}}$ instance, then $\mathcal{A}$ can stage an off-line dictionary attack running the user's reconstruction algorithm for every guess $\tilde{\mathsf{pw}}$ in the password dictionary D, and locally computing values $F_{\pi_i^*}(\tilde{\mathsf{pw}})$ using the PRF keys k_i^*. This is yet another reason why we need to extend the above protocol by adding a non-malleable commitment C that binds user's password pw to the reconstructed secret K. We accomplish this binding as follows: The CRS string will include an instance of a non-malleable commitment scheme COM. In the initialization procedure, the user secret-shares not the key K directly, but a random value s, and then it uses s as a PRG seed to derive the key K together with the commitment randomness r, and sets each state ω_i given to S_i to $(\boldsymbol{\pi}, \mathbf{c}, C)$ where $C = \mathsf{COM}((\mathsf{pw}, \boldsymbol{\pi}, \mathbf{c}); r)$. By the binding property of commitment COM, the adversary playing the role of the servers must commit to a password guess $\tilde{\mathsf{pw}}$ in value C it sends to the user, and the reconstruction procedure rejects unless the guess was right, i.e. unless $\tilde{\mathsf{pw}} = \mathsf{pw}$, disabling the off-line dictionary attack above. We need the non-malleability of the commitment scheme to forestall the possibility that the adversary modifies either the vector of function descriptors $\boldsymbol{\pi}$ or the ciphertexts $\mathbf{c}$, and hence in particular modifies the reconstructed key K, without guessing the password.

Communication Complexity, Robustness. In Figure 4 we show a PPSS scheme whose communication complexity is $O(n^2 \cdot \mathrm{poly}(\tau))$ where τ is a security parameter, because the protocol starts with each server S_i sending to U a tuple ω which contains n function descriptors π_i and n field elements c_i. The reason we do this is simplicity, plus we suspect that in most applications the number of servers n will be small enough that the $O(n^2)$ cost of this communication will not

Parameters: Security parameters τ and ℓ, binary extension field $\mathbb{F} = GF(2^\ell)$, session ID $sid = (\mathsf{S}_1, \ldots, \mathsf{S}_n)$, threshold parameters $t, n \in \mathbb{N}$.

$\underline{\mathsf{ParGen}(\tau)}$: Sets CRS as an instance of a non-malleable commitment COM.

$\underline{\mathsf{SKeyGen}(\mathsf{CRS})}$: S_i sends (KeyGen, sid) to $\mathcal{F}_{\text{VOPRF}}$ and sets π_i to π it receives in the response $(\text{Parameter}, sid, \pi)$ from $\mathcal{F}_{\text{VOPRF}}$. The private state σ_i of S_i is the unique handle "S_i" has to the V-OPRF function F_{π_i} implemented by the ideal $\mathcal{F}_{\text{VOPRF}}$ functionality. (In all our V-OPRF instantiations σ_i is a PRF key and the function descriptor π_i is a commitment to it.)

$\underline{\mathsf{U}_{\mathsf{Init}}(\mathsf{CRS}, \mathsf{pw}, \pi_1, \ldots, \pi_n) \rightleftharpoons \{\mathsf{S}_{\mathsf{Init}}(\mathsf{CRS}, \sigma_i, \pi_i)\}_{i=1}^n}$:
Step 1. User U picks $s \leftarrow \mathbb{F}$ and generates $(s_1, \ldots, s_n)$ as a (t, n) Shamir's secret-sharing of s over field $\mathbb{F}$. (Indices $0, 1, \ldots, n$ used in Shamir's secret-sharing are encoded as some distinct field elements $\langle 0\rangle_{\mathbb{F}}, \langle 1\rangle_{\mathbb{F}}, \ldots, \langle n\rangle_{\mathbb{F}}$.) For $i = 1$ to n, U sends $(\text{Eval}, sid, \mathsf{S}_i, \mathsf{pw})$ to $\mathcal{F}_{\text{VOPRF}}$.
Step 2. User U collects $\mathcal{F}_{\text{VOPRF}}$ responses $(\pi'_1, \rho_1), \ldots, (\pi'_n, \rho_n)$, and aborts if $\pi'_i \neq \pi_i$ for any i. If all parameters π'_i match those in the inputs, U computes $c_i \leftarrow s_i \oplus \rho_i$ for $i = 1$ to n, $\mathbf{c} \leftarrow (c_1, \ldots, c_n)$, $\boldsymbol{\pi} \leftarrow (\pi_1, \ldots, \pi_n)$, $[r||K] \leftarrow G(s)$, $C \leftarrow \mathsf{COM}((\mathsf{pw}, \boldsymbol{\pi}, \mathbf{c}); r)$, sends $\omega = (\boldsymbol{\pi}, \mathbf{c}, C)$ to each S_i, and outputs K as a local output.

$\underline{\mathsf{U}_{\mathsf{Rec}}(\mathsf{CRS}, \mathsf{pw}) \rightleftharpoons \{\mathsf{S}_{\mathsf{Rec}}(\mathsf{CRS}, \sigma_i, \pi_i, \omega_i)\}_{i\in S}}$:
For each $i = 1, \ldots, n$, user U sends $(\text{Eval}, sid, \mathsf{S}_i, \mathsf{pw})$ to $\mathcal{F}_{\text{VOPRF}}$ and initiates a run of the protocol Rec with S_i.
Each S_i responds by sending ω_i to U and $(\text{SenderComplete}, sid, \mathsf{S}_i)$ to $\mathcal{F}_{\text{VOPRF}}$, and U collects $\mathcal{F}_{\text{VOPRF}}$ responses (π'_i, ρ_i) and ω_i for each $i \in S$.
Let S be a subset of servers such that: (i) $|S| = t+1$; (ii) there exists $\omega = (\boldsymbol{\pi}, \mathbf{c}, C)$ with $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ and $\mathbf{c} = (c_1, \ldots, c_n)$ such that $\omega_i = \omega$ for all $S_i \in S$; (iii) for all $S_i \in S$, $\pi'_i = \pi_i$ and $\rho_i \neq \perp$. If no such subset exists output $\perp$ and halt.
Reconstruction: Set $u_i \leftarrow c_i \oplus \rho_i$ for all $i \in S$. Interpolate points $\{(\langle i\rangle_{\mathbb{F}}, u_i)\}_{i\in S}$ with a polynomial $U \in \mathbb{F}[x]$, and set $s \leftarrow U(\langle 0\rangle_{\mathbb{F}})$. Compute $[r||K] \leftarrow G(s)$. If $\mathsf{COM}((\mathsf{pw}, \boldsymbol{\pi}, \mathbf{c}); r) = C$ then output K, else output $\perp$.

Fig. 4. A PPSS scheme in the $\mathcal{F}_{\text{VOPRF}}$-hybrid-model.

be significant in practice. However, for large n we can reduce the communication to $O(n \log n)$ using a Merkle Tree hash [28]. Each server S_i would then send only its own π_i, c_i values together with the co-path in the hash tree which allows U to agree on the set of $t+1$ servers whose tree co-paths hash to the same root value. In practice U could also cash the ω vector as it does not change between Rec protocol instances, in which case the communication cost becomes $O(n)$. The communication cost can be decreased even further, to $O(t)$ group elements, at the cost of reducing robustness. The user could instigate V-OPRF instances with only $t+c$ servers instead of with all n, for any c between 1 and $n-t$. This would reduce bandwidth at the price of increasing the protocol costs in the case of an active attack: If just c among the $t+c$ servers U contacts are either corrupted or connected to U over corrupted links, the reconstruction attempt fails, and the user needs to instigate V-OPRF instances with the remaining servers.

Theorem 2. (PPSS Security) *Assuming commitment scheme* COM *is computationally hiding, computationally binding, and non-malleable (with respect to decommitment), and that* G *is a pseudorandom generator, the PPSS scheme in Figure 4 is* (T, q_u, q_s, ϵ)*-secure for* $\epsilon = \epsilon_H + \epsilon_B + q_u \cdot \epsilon_{NM} + 4\epsilon_G$*, where* ϵ_H*,* ϵ_B*,* ϵ_{NM} *and* ϵ_G *are the bounds implied by, respectively, computational hiding of* COM*, copmutational binding of* COM*, non-malleability of* COM *with respect to decommitment, and the pseudorandomness of* G*, on input sizes implied by the usage of* COM *and* G *in the PPSS scheme, for adversaries whose time is bounded by* T *plus the time taken by a single instance of* Init*,* q_u *instances of* $\mathsf{U_{Rec}}$*, and* q_s *instances of* $\mathsf{S_{Rec}}$*.*

Proof. See full version [19].

6 From PPSS to Single-Round T-PAKE

Composition of a PPSS scheme with a (regular) key-exchange protocol allows us to obtain very efficient one-round T-PAKE protocols with arbitrary threshold parameters and in the password-only CRS model, i.e. no PKI or secure channels are assumed. For lack of space we refer to the full version [19] for a general composition theorem proving the security of T-PAKE protocols built by this methodology. Here we only present examples of T-PAKE schemes obtained through this approach, and illustrate them with the most efficient T-PAKE instantiation, presented in Figure 5, resulting from our single-round PPSS of Section 5 implemented with the 2Hash-DH OPRF shown of Section 3.

T-PAKE via PPSS and symmetric-key KE. Let $\boldsymbol{P}$ be a (t, n)-PPSS protocol in the CRS model. To bootstrap a (t, n)-TPAKE protocol using $\boldsymbol{P}$, each server S_i, $i = 1, \ldots, n$, generates its state pair (σ_i, π_i) and runs with client C the Init procedure of protocol $\boldsymbol{P}$. As a result a user's secret, which we call K_C, is (t, n)-secret-shared among these servers under the protection of the PPSS scheme and the client's password pw. Next, client C uses key K_C to compute n keys $K_i = f_{K_C}(i), i = 1, \ldots, n$, where f is a pseudorandom function, and transmits each K_i (protected under the secure communication assumed at initialization) to the corresponding S_i who stores K_i in its client-specific $\zeta_i(C)$ state. Later, when a T-PAKE session at C is invoked, C runs the Rec procedure of protocol $\boldsymbol{P}$ with a sufficient number of servers to obtain K_C. C uses K_C to compute $K_1, \ldots, K_n$ and uses these keys as shared keys with the corresponding servers to exchange a session key. Any KE protocol that assumes pre-shared keys between pairs of parties can be used for this purpose. For example, C and S_i can compute their session key as $f_{K_i}(n_C, n_{\mathsf{S}_i}, id_C, id_{\mathsf{S}_i})$ where id_C, id_{S_i} stand for the identities of C and S_i respectively, and n_C, n_{S_i} are nonces exchanged between these parties that also serve as session identifiers. Note that when using a one-round PPSS scheme, the exchange of nonces can be piggybacked on top of the PPSS messages hence *preserving the single round complexity of the protocol* (with one additional message from C to S_i if key confirmation is desired). A full specification of this protocol based on the 2HashDH-NIZK V-OPRF is presented in Figure 5. One can also add forward secrecy to the protocol by using the shared key to authenticate a Diffie-Hellman exchange (also piggybacked on top of the two PPSS messages to preserve the single-round complexity).

T-PAKE via PPSS and public-key KE. The above scheme provides a full T-PAKE protocol with very little extra cost over the PPSS scheme. Its relative drawback is (as in any pre-shared key scheme) that the server needs to keep a per-client secret and also that it requires secrecy for the transmission of key K_i to S_i (otherwise, our PPSS scheme only needs authenticated channels during initialization). To avoid these secrecy requirements, key exchange protocols based on public keys of the parties can be accommodated on top of a PPSS as follows. At initialization, the client generates a pair of private and public keys, and obtains public keys for all its servers. C then generates a file (we call it a $\mathsf{keystore}$ in our formal treatment [19]) that includes its own key pair (with the private key encrypted under a key derived from K_C) and the servers' public keys. The $\mathsf{keystore}$ is stored at each server authenticated with a MAC computed by C using a key derived from K_C. In addition, each server stores C's public key. When a T-PAKE session is invoked at C, the client retrieves $\mathsf{keystore}$ from the servers and, after reconstructing K_C, uses this key to check the integrity of $\mathsf{keystore}$ and to decrypt its private key. With this information and the (authenticated) public keys of the servers contained in $\mathsf{keystore}$, C is ready to perform the key exchange protocol. Similarly, the servers can use C's public key that they stored to bootstrap the public-key based key exchange. In particular, using a two-message KE protocol whose messages are independent of the parties private- and public-keys. (such as HMQV [26]), one obtains a single-round T-PAKE by piggy-backing the two KE messages on top of the two PPSS ones.

DH-based Instantiation of PPSS and T-PAKE. For illustration and for the reader's convenience we describe in Figure 5 the specific instantiation of the PPSS and T-PAKE protocols based on the 2HashDH-NIZK V-OPRF, with the NIZK for DL equality implemented as in [12], and a symmetric-key KE scheme. We comment on some of our choices for this illustration. The initialization is presented for the case in which the client generates the servers' V-OPRF keys and computes all the values in the ω vector by itself. Another option, more in line with the formal description of the PPSS protocol from Figure 4, is for the servers to choose their own V-OPRF keys and engage in an V-OPRF computation with the client for generating the pads used to encrypt the shares s_i. One advantage of the latter option is that servers can save in the amount of secret memory and derive the V-OPRF keys for each user U using a single key MK and a PRF F, i.e., as $k_{\mathsf{U}} = F_{MK}(\mathsf{U})$ (we are abusing the symbols U and S_i to denote the identities of these parties). This option is more useful with a PK-based KE, where servers do not need to store user-specific secrets (in contrast, the protocol from Figure 5 requires the server storing the session key with each user). User performance during reconstruction is improved by choosing a common value ρ for blinding the $H_1(\mathsf{pw})$ value sent to all servers. We stress that while we specifiy the actions of honest servers, corrupted ones can deviate from the protocol in any way they choose to. Finally, note that the protocol as presented does not include

Parties: User U, Servers $\mathsf{S}_1, \ldots, \mathsf{S}_n$.

Public parameters and components: Security parameters τ and ℓ, threshold parameters $t, n \in \mathbb{N}, t \leq n$, field $\mathbb{F} = GF(2^\ell)$, cyclic group of prime order m with generator g; hash functions H_1, H_2, H_3, H_4, H_5 with ranges $\langle g \rangle, \{0,1\}^\ell, \{0,1\}^\tau, Z_m, Z_m$, respectively; pseudorandom generator G and $\boxed{\text{pseudorandom function family } f}$.

Initialization (secure channels between U and each server S_i are assumed only through initialization): User U performs the following steps:

1. Chooses $s \in_{\mathrm{R}} \mathbb{F}$ and generates shares $(s_1, \ldots, s_n)$ as a (t, n) Shamir's secret-sharing of s over field $\mathbb{F}$.
2. For $i = 1, \ldots, n$, U chooses value $k_i \in_{\mathrm{R}} Z_m$ and sets $\pi_i = g^{k_i}$ and $c_i = s_i \oplus H_2(\pi_i, \mathsf{pw}, (H_1(\mathsf{pw}))^{k_i})$.
3. Sets $\mathbf{c} = (c_1, \ldots, c_n)$, $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$, $[r||K] = G(s)$, $C = H_3(r, \mathsf{pw}, \boldsymbol{\pi}, \mathbf{c})$; $\boxed{\text{For } i = 1, \ldots, n, \text{ sets } K_i = f_K(\mathsf{S}_i)}$.
4. For $i = 1, \ldots, n$, sends to server S_i the values $\omega_i = (\boldsymbol{\pi}, \mathbf{c}, C), k_i, \boxed{K_i}$.
5. U memorizes pw and erases all other information.

Each server S_i, $i = 1, \ldots, n$, stores $\omega_i, k_i, y_i = g^{k_i}, \boxed{K_i}$ in its U-specific storage ζ_i.

Reconstruction/Key Exchange

– User U initiates a key exchange session with servers $\mathsf{S}_1, \ldots, \mathsf{S}_n$ by sending to each S_i the value $a = (H_1(\mathsf{pw}))^\rho$ with $\rho \in_R \mathbb{Z}_m$ $\boxed{\text{and a nonce } \mu_i \in_{\mathrm{R}} \{0,1\}^\tau}$.

– Upon receiving $(a, \boxed{\mu_i})$, server S_i checks that $a \in \langle g \rangle$ and if so, S_i retrieves k_i and $y_i = g^{k_i}$ from its U-specific storage $\zeta_i(\mathsf{U})$, picks $z \in_{\mathrm{R}} \mathbb{Z}_m$, and computes $b_i = a^{k_i}$, $\gamma = H_4(g, y_i, a, b_i)$, $v_i = H_5(g, y_i, a, b_i, (g \cdot a^\gamma)^z)$, and $u_i = z + v_i \cdot k_i \bmod m$. S_i sends to U the values y_i, b_i, u_i, v_i as well as $\boxed{\text{a nonce } \mu_i' \in_{\mathrm{R}} \{0,1\}^\tau \text{ and}}$ the value ω_i stored in $\zeta_i(\mathsf{U})$. $\boxed{\mathsf{S}_i \text{ computes the session key with } \mathsf{U} \text{ as } SK_i = f_{K_i}(\mu_i, \mu_i', \mathsf{U}, \mathsf{S}_i).}$

– Upon receiving values $b_i, u_i, v_i, \omega_i, \boxed{\mu_i'}$ from S_i, U proceeds as follows:

- U chooses a subset of servers S for which the following conditions hold: (i) there is a value $\omega = (\boldsymbol{\pi}, \mathbf{c}, C)$ with $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ and $\mathbf{c} = (c_1, \ldots, c_n)$ such that $\omega_i = \omega$ for all $\mathsf{S}_i \in S$; (ii) $y_i = \pi_i$ for all $\mathsf{S}_i \in S$; (iii) $b_i \in \langle g \rangle$ and the equality $v_i = H_5(g, y_i, a, b_i, (g \cdot a^\gamma)^{u_i} \cdot (y_i \cdot b_i{}^\gamma)^{-v_i})$ for $\gamma = H_4(g, y_i, a, b_i)$ holds for all $\mathsf{S}_i \in S$; (iv) $|S| = t + 1$.
- If no such subset exists U aborts. Else, set $s_i = c_i \oplus H_2(y_i, \mathsf{pw}, b_i^{1/\rho})$, for each $\mathsf{S}_i \in S$, and reconstruct s from these s_i shares using polynomial interpolation.
- Compute $[r||K] = G(s)$. If $C \neq H_3(r, \mathsf{pw}, \boldsymbol{\pi}, \mathbf{c})$ then U aborts.
- $\boxed{\text{For each } \mathsf{S}_i \in S, \text{ set } K_i = f_K(\mathsf{S}_i) \text{ and compute } SK_i = f_{K_i}(\mu_i, \mu_i', \mathsf{U}, \mathsf{S}_i).}$

Fig. 5. DH-based PPSS and T-PAKE Protocols (boxed text indicates key-exchange specific operations on top of PPSS).

an explicit authentication mechanism. This can be easily added, for example, by server S_i adding the value $f_{K_i}(0, \mu_i, \mu_i'))$ to its message and by U adding a third message with value $f_{K_i}(1, \mu_i', \mu_i)$ (in this case, the session key could be derived as $SK_i = f_{K_i}(2, \mu_i, \mu_i', \mathsf{U}, \mathsf{S}_i))$.

Acknowledgements. The second author was partly supported by ERC grant CODAMODA.

References

1. M. Abe and M. Ohkubo. A framework for universally composable non-committing blind signatures. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 435–450. Springer, 2009.
2. A. Bagherzandi, S. Jarecki, N. Saxena, and Y. Lu. Password-protected secret sharing. In *ACM Conference on Computer and Communications Security*, 2011.
3. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *J. Cryptology*, 16(3):185–215, 2003.
4. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, May 2000.
5. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New techniques for sphfs and efficient one-round pake protocols. In *Crypto'2013*.
6. J. Brainard, A. Juels, B. Kaliski, and M. Szydlo. A new two-server approach for authentication with short secrets. In *12th USENIX Security Symp*, 2003.
7. J. Camenisch, A. Lehmann, A. Lysyanskaya, and G. Neven. Memento: How to reconstruct your secrets from a single password in a hostile environment. In *CRYPTO'2014*, pages 256–275, 2014.
8. J. Camenisch, A. Lehmann, A. Lysyanskaya, and G. Neven. Memento: How to reconstruct your secrets from a single password in a hostile environment. In J. A. Garay and R. Gennaro, editors, *CRYPTO (2)*, volume 8617 of *Lecture Notes in Computer Science*, pages 256–275. Springer, 2014.
9. J. Camenisch, A. Lysyanskaya, and G. Neven. Practical yet universally composable two-server password-authenticated secret sharing. In *ACM Conference on Computer and Communications Security*, pages 525–536, 2012.
10. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, Oct. 2001.
11. D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for Boolean queries. Crypto'2013. Cryptology ePrint Archive, Report 2013/169, Mar. 2013.
12. S. Chow, C. Ma, and J. Weng. Zero-knowledge argument for simultaneous discrete logarithms. In *Computing and Combinatorics*, volume 6196 of *Lecture Notes in Computer Science*, pages 520–529. Springer Berlin Heidelberg, 2010.
13. R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 79–88. ACM Press, Oct. / Nov. 2006.

14. M. Di Raimondo and R. Gennaro. Provably secure threshold password-authenticated key exchange. *J. Comput. Syst. Sci.*, 72(6):978–1001, 2006.
15. M. Fischlin. Round-optimal composable blind signatures in the common reference string model. In C. Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 60–77. Springer, 2006.
16. W. Ford and B. S. K. Jr. Server-assisted generation of a strong secret from a password. In *WETICE*, pages 176–180, 2000.
17. M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In J. Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324. Springer, Feb. 2005.
18. D. Jablon. Password authentication using multiple servers. In *CT-RSA'01: RSA Cryptographers' Track*, pages 344–360. Springer-Verlag, 2001.
19. S. Jarecki, A. Kiayias, and H. Krawczyk. Round-optimal password-protected secret sharing and t-pake in the password-only model. Cryptology ePrint Archive, Report 2014/650, 2014. `http://eprint.iacr.org/`.
20. S. Jarecki and X. Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In O. Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 577–594. Springer, Mar. 2009.
21. S. Jarecki and X. Liu. Fast secure computation of set intersection. In *SCN 10: 7th International Conference on Security in Communication Networks*, Lecture Notes in Computer Science, pages 418–435. Springer, 2010.
22. J. Katz, P. Mackenzie, G. Taban, and V. Gligor. Two-server password-only authenticated key exchange. In *Proc. Applied Cryptography and Network Security ACNS05*, 2005.
23. J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques*, 2001.
24. J. Katz and V. Vaikuntanathan. Round-optimal password-based authenticated key exchange. *J. Cryptology*, 26(4):714–743, 2013.
25. A. Kiayias and H.-S. Zhou. Equivocal blind signatures and adaptive uc-security. In R. Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 340–355. Springer, 2008.
26. H. Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In *CRYPTO*, pages 546–566, 2005.
27. P. D. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold password-authenticated key exchange. *J. Cryptology*, 19(1):27–66, 2006.
28. R. Merkle. A digital signature based on a conventional encryption function. In C. Pomerance, editor, *Advances in Cryptology CRYPTO 87*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer Berlin Heidelberg, 1988.
29. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 120–130. IEEE Computer Society Press, Oct. 1999.
30. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS*, pages 458–467. IEEE Computer Society, 1997.
31. New York Times. Russian Hackers Amass Over a Billion Internet Passwords. `http://www.nytimes.com/2014/08/06/technology/russian-gang-said-to-amass-more-than-a-billion-stolen-internet-credentials.html?_r=0`, Aug. 5, 2015.