Password-Protected Secret Sharing

Ali Bagherzandi[1], Stanislaw Jarecki[1], Yanbin Lu[1], Nitesh Saxena[2]

[1]University of California, Irvine
{zandi,stasio,yanbinl}@ics.uci.edu
[2]Polytechnic Institute of New York University
nsaxena@poly.edu

Abstract. We revisit the problem of protecting user's private data against adversarial compromise of user's device(s) which would normally store this data. We formalize an attractive solution to this problem as *Password-Protected Secret-Sharing* (PPSS), which is a protocol that allows a user to secret-share her data among n trustees in such a way that (1) the user can retrieve the shared secret upon entering a correct password into a reconstruction protocol which succeeds as long as at least $t + 1$ honest trustees participate, and (2) the shared data remains secret even against the adversary which corrupts at most t servers, with the level of protection expected of password-authentication, i.e. the probability that the adversary learns anything useful about the secret is at most negligibly greater than $q/|D|$ where q is the number of reconstruction protocol instances in which adversary engages and $|D|$ is the size of the dictionary from which the password was randomly chosen.
We propose an efficient PPSS protocol in the public key model, i.e. where the device can remember a trusted public key, provably secure under the DDH assumption, using non-interactive zero-knowledge proofs which are efficiently instantiatable in the Random Oracle Model (ROM). The resulting protocol is robust and practical, with fewer than $4t + 12$ exponentiations per party, and with only three messages exchanged between the user and each server, implying a single round of interaction in the on-line phase. As a side benefit our PPSS protocol yields a new Threshold Password Authenticated Key Exchange (T-PAKE) protocol in the public key model which is significantly faster than existing T-PAKE's provably secure in the public key model in ROM.

Keywords: Secret Sharing; Intrusion Tolerance; Password Authentication; Distributed Protocols

1 Introduction

A user of computing technology, let's call her Alice, needs to store her valuable and private data, including texts, images, passwords, or cryptographic keys, on her personal device. Alice's device, however, can fall prey to viruses, spyware, Trojan horses, and other type of malware, exposing Alice's data and breaching her privacy. Moreover, Alice might want to store her private data on a portable device (e.g. a notebook or a cell-phone) which can easily get temporarily or permanently in control of someone whom Alice does not trust. Finally, Alice's device is susceptible to hardware failures (e.g. a hard disk crash or water damage of a cell phone) in which case her data could be lost.

In this paper, we consider the general problem of protecting Alice's data, let's call it M, in the event of the compromise or failure of the device on which this data is stored. A commonly used approach aimed at addressing this problem is to store an encryption of M using a key derived from a password. This approach, however, is vulnerable to an offline password dictionary attack once the device is corrupted (see, e.g. [FK90,Kle90,MT79,Wu99]), and is also not robust to device failure. Another possible approach is to outsource M to a trusted remote server, and let Alice authenticate to this remote server whenever she needs to retrieve M. Since we want Alice's data to be secure against a compromise of Alice's device, no authentication information can be stored on her device, or otherwise the adversary would learn it after a corruption, and can use it to retrieve M from the trusted server. Thus Alice has to use password authentication, e.g. a password authenticated key exchange (PAKE) protocol, e.g. [BPR00,BMP00]), to retrieve M using a password shared with the server. This approach, however, places all Alice's trust in a single server. Moreover, it exposes Alice's data to an on-line dictionary attack which a network adversary can stage against this server. Both concerns can be assuaged if Alice secret-shares M, keeps one share for herself, and outsources the other share to the server. Instead of doing this with a generic combination of PAKE and secret-sharing, MacKenzie and Reiter [MR03] show that this can be done in a way which gives the server no useful information about either the data or the password, and after corruption of the device the adversary can recover any information about either the data or the password only via an on-line dictionary attack against the server. This solution, however, suffers from several bottlenecks: Alice would be cut off from her data whenever the server is down or overloaded, and she could loose her data completely if the adversary compromises *either* the server or her device.

Password-Protected Secret Sharing: In order to improve both security and robustness of the above solution, i.e. both data privacy and its accessibility in case of server or device failures, we can secret-share M among a *set* of n players so that only a compromise of more than some threshold of players would let disable the system. However, secret sharing by itself is not enough because Alice still needs some way to authenticate to the secret-sharing servers to trigger a secret reconstruction protocol, and the only authentication information she can use must be a human-memorable password. It can be easily seen that the generic combination of PAKE and secret-sharing would either require Alice to keep n independently chosen passwords, or it would expose Alice's single password to the adversary who corrupts just one server via an off-line dictionary attack. Since the corrupt server could use this password to authenticate to the others and recover the secret-shared secret, all benefits of secret-sharing would be lost. This motivates the question of how to distribute Alice's data among n players so that: (1) Alice can retrieve her data by triggering a reconstruction protocol using only her password, and the reconstruction is guaranteed to succeed as long as at least $t + 1$ players are honest and available; and (2) Alice's data remains secret even when t players are corrupted, and the level of this protection is as expected of password-authenticated protocol, i.e. the probability that adversary learns any information about Alice's data is at most negligibly greater than $q_S/|D|$, where q_S is roughly the number of reconstruction protocols which the adversary triggers and $|D|$ is the size of the dictionary from which Alice chose her password.

We refer to a protocol that satisfies the above properties as (t, n) *Password-Protected Secret Sharing (PPSS)*. We say that a PPSS protocol is in a Public Key Model if Alice needs to trust some public key(s) to run the protocol, as is the case for the protocol we propose in this paper. The exact probability threshold we want to achieve in the security property in item (2) above is (no more than negligible amount greater than) $\lfloor \frac{q_S}{t-t'+1} \rfloor * \frac{1}{|D|}$, where $t' \leq t$ is the number of corrupted players and q_S counts the number of individual sessions run by any player where the adversary injects himself as a man in the middle. Note that this implies that an adversary which corrupts $t' \leq t$ players must stage an active attack on at least $(t - t' + 1)k$ other player's sessions in order to make k password guesses. This is indeed the optimum guessing-attack resistance such scheme can provide.

Two Application Settings: Private Storage There are two settings in which such PPSS protocol can be implemented, implying different security properties: In the (default) "no private storage" setting which was implicitly assumed in the above description, Alice's device does not have any private storage, and the n players participating in the PPSS scheme are n separate trustees, implemented by servers or any networked devices. This setting is highly robust, because the adversary can destroy Alice's data only by corrupting the memory of $n - t$ out of n trustees. Moreover, the loss of Alice's device(s) gives no information to the adversary at all since they hold no secret information at all, and Alice's data (and password) are secure even if the adversary corrupts t out of n trustees. However, Alice's data (and password) are vulnerable to network attacker via an online dictionary attack, and the adversary corrupting $t + 1$ trustees recovers both Alice's data and her password (possibly after an off-line dictionary attack).

Alternatively, a PPSS scheme can be used in a "private storage" setting, in case Alice does not fully trust *any* external servers: Alice's device and n trustees can run an instance of (t', n')-PPSS scheme for $t' = n$ and $n' = 2n - t$, keep $d = n - t$ shares on Alice's device and distribute the remaining $n' - d = n$ shares among the n trustees. The PPSS reconstruction protocol proceeds in the same way as in the default setting above, except that Alice's device locally implements the protocol on behalf of d "virtual" players using the d stored shares. Moreover, unlike the default setting, these virtual players do not respond to outside requests to start the protocol. The resulting protocol is less robust because the adversary can now destroy Alice's data by either corrupting the memory of $n - t$ trustees *or* by corrupting the memory of Alice's device. However, it is more secure. As in the default setting, an adversary gains no information about Alice's data or password even after corrupting Alice's device together with t trustees, but now the adversary that does not corrupt Alice's device gains no useful information even after corrupting all n trustees. Moreover, a network attacker can no longer stage an on-line dictionary attack against Alice's data (and password) even after corrupting all the trustees.

Applications: A PPSS scheme can be implemented by a dedicated service provider as service to individual users, but it can also be administered by users themselves, exploiting users' real-world trust relationships, e.g. from online social networks such as *Facebook*. For example using the PPSS scheme in a private storage setting, Alice can improve the availability of her data without any exposure to its security (compared to storing the data locally) if no more than t out of her n chosen friends are dishonest and/or their computers are corrupted. The role of the trustees can also be played by different devices belonging to Alice herself, perhaps connected by any combination of cellular, Bluetooth, or internet networks. For example, Alice could use PPSS so that the adversary can gain possession of the secret data Alice needs to retrieve to her mobile phone only by getting control not only of her phone but also of t out of n of devices, e.g. her Bluetooth-connected watch, her (always connected) home computer, her work computer. Note that these various types of players could be mixed, so a dedicated service provider and the home computers of Alice's Facebook friends can join the pool of the n "devices" in the last example.

Note that the sensitive data which Alice can protect using a PPSS scheme can be her secret key, which can then be used for any cryptographic computation such as decryption, signing, or authentication. Thus in particular a PPSS scheme generalizes the notion of *capture-resilient* cryptographic devices, introduced by MacKenzie and Reiter [MR03], whose key is recoverable only by an online dictionary attack after device corruption, by making such system more robust to trustees' failures. One instance of such application is a *password management* scheme, where Alice's data to be protected are her passwords to various on-line services. Recovering such passwords via a PPSS scheme using a "master password" avoids the need for recalling and typing in an appropriate password every time Alice needs it to authenticate to various services. Moreover, such password management scheme could improve the security of passwords, because now Alice can use an independently random string as password for each of her services. Currently various entities offer remote password management service on the web, e.g. LastPass [Las09] and Mozilla Weave Sync [Lab09], but these are effectively centralized solutions. A PPSS scheme could enable alternatives to such services with better robustness and security properties.

Related Work: The idea of password-authenticated recovery of secret-shared data was to the best of our knowledge proposed by Ford and Kaliski [FJ00]. As pointed out by a subsequent paper by Jablon [Jab01], the Ford-Kaliski protocol implicitly assumes a public key model, while the improvement given by [Jab01] removed the need for trusted public keys, but both protocol handles only the $t = n$ case of secret-sharing, and neither paper clearly specified the security properties of such scheme nor did they formally argue the security of the protocols they proposed.

The notion of PPSS is closely related to Threshold Password Authenticated Key Exchange (T-PAKE) of MacKenzie et al. [MSJ02], which followed up on the work of [FJ00,Jab01] by formalizing the functionality which these works were after in a general-purpose way, namely as password-authenticated key exchange between Alice's device and each of the n trustees. This is a more general functionality than password-authenticated secret recovery. Indeed, a T-PAKE scheme can be used to implement a PPSS scheme at negligible extra costs: The client authenticates to the servers by running a T-PAKE instance, and the servers use the keys established in this instance to encrypt (and authenticate) their secret-shares of Alice's secret. As we show in this paper, it turns out that in the public key model the implication works in the opposite direction as well, i.e. a PPSS implies a T-PAKE with little extra costs: Alice picks a (private,public) key pair in a CCA-secure encryption scheme, and uses a PPSS scheme to share the private decryption key. A T-PAKE protocol consists of a PPSS reconstruction protocol followed by each server encrypting a freshly chosen session key under Alice's public key. Alice can recover these keys if she recovered her private decryption key in this PPSS instance. To authenticate the servers to Alice each Server supplies also its signature on this encrypted key, which Alice verifies using its knowledge of Servers' public keys. (We supply the details of this simple compilation in Section 4 below.) Note that this construction relies on a public key model, and it is not immediately obvious whether a PPSS protocol can be converted into a T-PAKE protocol in a similarly simple way without relying on Alice's knowledge of Servers' public keys.[1]

The work of MacKenzie et al. [MSJ02], in addition to defining the notion of T-PAKE, showed a T-PAKE protocol in the public key model which was provably secure under the Decisional Diffie-Hellman (DDH) assumption in the Random Oracle Model (ROM), but the protocol required three rounds of server-to-server communication. This was followed by Raimondo-Gennaro [RG03] who showed a T-PAKE scheme based on the PAKE protocol of Katz et al. [KOY01] which removed both the reliance on ROM and the need for trusted public keys, but their protocol was even more communication-heavy, requiring several rounds of verifiable secret sharing among the servers, each of which requires a few communication rounds over a reliable broadcast channel. Such trustee-to-trustee communication might be feasible for T-PAKE applications discussed in [MSJ02,RG03], where all n secret-sharing servers belong to the same organization and can foreseeably be connected by a LAN. However, for many PPSS applications we discussed above such communication requirements would be difficult to bear in practice, e.g. when trustees are highly distributed, connected over internet, phone, and/or Bluetooth networks. In such applications one should assume all protocol communication is between the user and a trustee, and there is no direct communication between trustees. In such model the PPSS protocol implied by T-PAKE of [MSJ02] requires 9 messages exchanged between the user and each trustee, while the T-PAKE of [RG03] requires even more.

The general (t, n)-threshold T-PAKE protocol of [MSJ02] was improved in efficiency for the special case of 2 servers, i.e. $(t, n) = (1, 2)$, by Brainard [BJKS03], but the resulting protocol assumes direct (and secure) connection between the two servers, and would imply a 7 messages protocol in our all-communication-via-the-client model. In the setting without public keys the protocol of [RG03] was extended to the same special case by Katz et al. [KMTG05], but while significantly faster than [RG03], the resulting protocol still needs 5 message exchanges in our communication model, assuming the zero-knowledge proofs are implemented non-interactively in ROM. (Otherwise the message complexity of [KMTG05] would be even higher).

In another related work, Xu and Sandhu [XS03] consider a different variant of T-PAKE functionality, namely password-authenticated threshold signature generation which they call TPAKE-TSig. However, the

[1] In spite of this near-equivalence between PPSS and T-PAKE, we believe that the primary reason for introducing PPSS as a separate notion is that it is a natural functionality – a protocol that reconstructs a (long) secret if and only if supplied with a correct (short) password. It is also easier to define than T-PAKE, which requires a somewhat elaborate formalism developed to model Password-Authenticated Key Exchange protocols, i.e. [BPR00,BMP00]. The PPSS functionality is simpler because it recovers one secret at one party (the user), instead of creating keys for each party, possibly at multiple sessions, which the adversary can then adaptively reveal, with somewhat complicated rules governing which key should remain secret given that another key was revealed. Such functionality simplification can be productive, and indeed, in this paper we show a PPSS protocol which is faster and simpler than existing provably secure T-PAKE protocols.

security notion they formalize for such TPAKE-TSig's continues the same complicated model of T-PAKE's, and their construction is a straightforward composition of T-PAKE with threshold signatures, along the lines of the simple T-PAKE$\rightarrow$PSSS complier sketched above, thus in particular the resulting protocol can only be as fast as existing T-PAKE's.

The PPSS protocol cast into the private storage setting is a generalization of the aforementioned work on "capture-resilient devices" by MacKenzie and Reiter [MR03], from the case of just one external trustee to the case of any n trustees, with resilience to the corruptions of the device and t trustees. [MR03] consider not only password-authenticated recovery of a shared secret but also password-protected generation of RSA signatures and ElGamal decryption. However, note that any T-PAKE protocol, and a PPSS protocol implies T-PAKE, can be used very simply to implement such password-protected signature or decryption given a non-interactive threshold computation protocol for the respective signature/decryption function: The servers simply encrypt the messages of the non-interactive threshold function-computation protocol under the session keys output by a T-PAKE instance, and the user can reconstruct the function value if she established the same session keys in the T-PAKE instance. Thus the 2-party password-protected signature/decryption computation protocols of [MR03] can be generalized using to any (t, n) case using our PPSS and non-interactive threshold RSA signature of [Sho00] and non-interactive threshold ElGamal decryption of [DF90].

PPSS is more remotely related to the so-called "key-insulated" and "intrusion-tolerant" cryptosystems, e.g. [DKXY02,DKXY03,DFK$^+$04,DFK$^+$03], which are similarly concerned with protecting user's sensitive data against device intrusions or corruptions by secret-sharing user's private key among more trusted components, e.g. smartcards or co-processors. However, unlike PPSS, these cryptosystems assume semi-trusted components which are embedded in user's computational device so the user does not need to remotely authenticate herself to them, e.g. via password authentication.

Our Contributions: We develop an efficient PPSS scheme in the public key model provably secure under the DDH assumption, using non-malleable non-interactive zero-knowledge proofs which can be efficiently instantiated in ROM. Our protocol is robust against active failures and, unlike PPSS based on any existing provably secure T-PAKE, it is much more efficient: In particular, in the absence of active attacks each party makes fewer than $4t + 15$ exponentiations and the protocol involves only three messages exchanged between the user and each server. (In particular, this implies a single round of interaction during the on-line phase.) We also show that a PPSS scheme implies a T-PAKE protocol at a very little computation overhead and no additional communication cost. Thus, as a side benefit, our PPSS implies a T-PAKE protocol in the public key model in ROM, with three message exchanges and less than $4t + 18$ exponentiations per party, which is significantly faster than the most efficient existing T-PAKE protocol of [MSJ02]. For the special case of $n = 2$, the resulting T-PAKE reduces message complexity but not the computation of the 2-party T-PAKE of [BJKS03]. (However, cast in the private storage setting with one external trustee, our protocol does not improve on the one of [MR03], which requires only a few exponentiations and two messages.)

Organization: The rest of this paper is organized as follows. In Section 2 we present our definition and security model for PPSS. In Section 3 we present our PPSS protocol, first the intuition behind it, then the full protocol, and finally the proof of its security. In Section 4 we sketch the compiler from PPSS to T-PAKE. In Appendix A we provide the background on simulation-sound zero-knowledge labeled proof systems used in our PPSS protocol, and Appendix B contains the proofs of technical claims used in Section 3.

2 Password-Protected Secret Sharing: Definitions

A Password-Protected Secret Sharing (PPSS) scheme operates in environment involving a user U and n servers $\mathsf{P}_1, ..., \mathsf{P}_n$. A PPSS scheme for secret space S and dictionary D is a tuple $(\mathsf{Init}, \mathsf{User}, \mathsf{Server})$, where $\mathsf{Init}(p, s)$ is an initialization algorithm which on inputs a secret $s \in S$ and password $p \in D$ generates

$\mathsf{st} = (\mathsf{st}_0, \mathsf{st}_1, ..., \mathsf{st}_n)$ where st_0 are public parameters and st_i is the private state of server P_i; $\mathsf{User}(p^*, \mathsf{st}_0)$ is an interactive algorithm followed by U on its password p^* (presumably equal to p) and parameters st_0; and $\mathsf{Server}(\mathsf{st}_i)$ is an interactive algorithm followed by P_i on input st_i. The user must reconstruct the secret s when she inputs the correct password p, while inputting an incorrect password $p^* \neq p$ should lead to rejection. Formally, if $\mathsf{PPSS}(p, \mathsf{st})$ is a random variable defined as the local output of algorithm $\mathsf{User}(p, \mathsf{st}_0)$ after an interaction with oracles $\mathsf{Server}(\mathsf{st}_1)$, ..., $\mathsf{Server}(\mathsf{st}_n)$ then we must have that $\mathsf{PPSS}(p, \mathsf{Init}(p, s)) = s$ for any $(s, p) \in S \times D$, and we call a PPSS scheme ϵ-sound if for any $s \in S$ and any $p, p^* \in D$ s.t. $p \neq p^*$, we have $\Pr[\mathsf{PPSS}(p^*, \mathsf{Init}(p, s)) \neq \perp] \leq \epsilon$.

To model concurrent execution of several PPSS protocol instances we denote by $\mathsf{User}^\diamond(p, \mathsf{st}_0)$ an oracle which allows the caller to interact with any number of $\mathsf{User}(p, \mathsf{st}_0)$ instances. Importantly, the caller sees only protocol messages output by each User instance it interacts with, and not the local output of any of these instances. Similarly, for any set B we denote by $\mathsf{Server}^\diamond(\mathsf{st}_{\overline{\mathsf{B}}})$ an oracle which allows the caller to interact with any number of $\mathsf{Server}(\mathsf{st}_i)$ instances for any i in $\overline{\mathsf{B}} \triangleq \{1, ..., n\} \setminus \mathsf{B}$. We say that probabilistic algorithm $\mathcal{A}$ interacts with at most q_U user sessions if in any of its executions $\mathcal{A}$ initializes at most q_U instances of $\mathsf{User}(p, \mathsf{st}_0)$ algorithm when interacting with oracle $\mathsf{User}^\diamond(p, \mathsf{st}_0)$, and we say that $\mathcal{A}$ interacts with at most q_S server sessions if in any execution of $\mathcal{A}$ we have $\sum_{i \in \overline{\mathsf{B}}} q_i \leq q_S$ where q_i is the number of $\mathsf{Server}(st_i)$ instances $\mathcal{A}$ initializes when interacting with oracle $\mathsf{Server}^\diamond(\mathsf{st}_{\overline{\mathsf{B}}})$.

We define security of a PPSS scheme in terms of adversary's advantage in distinguishing between two PPSS instances initialized with two different secrets, where the adversary sees the public parameters st_0, the private states $\mathsf{st}_\mathsf{B} \triangleq \{\mathsf{st}_i\}_{i \in \mathsf{B}}$ of the set of servers $\{P_i\}_{i \in \mathsf{B}}$ it corrupts, and has concurrent oracle access to instances of the user and server algorithms executing on inputs defined by the initialization procedure. We call an (t, n)-threshold PPSS scheme secure if this advantage is bounded by $\frac{1}{|D|}$, the probability of guessing the password, times $\lfloor \frac{q_S}{t-t'+1} \rfloor$ where $t' \leq t$ is the number of servers an adversary corrupts, plus at most a negligible amount. The last factor is a threshold equivalent of the probability of success of an on-line dictionary attack: An adversary who learns the shares of t' servers can test any password $\tilde{p}$ in D by executing $\mathsf{User}(\tilde{p}, st_0)$ and interacting with any subset of $t - t' + 1$ uncorrupted servers.

Definition 1. *(security) A PPSS scheme on dictionary D and space S is $(n, t, T, q_U, q_S, \epsilon)$-*secure *if for any $s_0, s_1 \in S$, any set B s.t. $t' \triangleq |\mathsf{B}|$ satisfies $t' \leq t$, and any probabilistic algorithm $\mathcal{A}$ with running time T which interacts with at most q_U user and q_S server sessions, we have*

$$|p_0 - p_1| \leq \lfloor \frac{q_S}{t - t' + 1} \rfloor * \frac{1}{|D|} + \epsilon$$

where

$$p_b \triangleq \Pr[\ 1 \leftarrow \mathcal{A}^{\mathsf{User}^\diamond(p, \mathsf{st}_0), \mathsf{Server}^\diamond(\mathsf{st}_{\overline{\mathsf{B}}})}(s_0, s_1, \mathsf{st}_0, \mathsf{st}_\mathsf{B})\ ]$$

where $\mathsf{st} \leftarrow \mathsf{Init}(p, s_b)$ for $p \leftarrow D$, and the probability goes over the randomness of all algorithms.

To make PPSS scheme easy to use as a building block in further applications, e.g. in a construction of a T-PAKE protocol shown in Section 4, we need to strengthen the above security definition so that the shared secret remains hidden even if the adversary learns whether each user session output some reconstructed secret or it rejected in that protocol instance:

Definition 2. *(strong security) A PPSS scheme on dictionary D and space S is $(n, t, T, q_U, q_S, \epsilon)$-*strongly secure *if it satisfies definition 1 even when the user oracle $\mathsf{User}^\diamond(p, \mathsf{st}_0)$ is modified so that for every instance of $\mathsf{User}(p, \mathsf{st}_0)$ the adversary learns a bit which indicates whether this instance locally outputted some secret s or it locally outputted a rejection sign $\perp$.*

We define two more useful notions for a PSSS scheme, of *soundness* and *robustness*. Soundness says that even corrupt servers cannot make the user recover a secret which is different from the one which was initially shared. Robustness says that recovery of the shared secret is assured as long the user communicates with at least $t+1$ honest servers.

Definition 3. *(soundness) A PPSS scheme on dictionary D and space S is* (T,ϵ)-sound *if for any $s \in S$, any $p \in D$, and any probabilistic algorithm $\mathcal{A}$ with running time T, we have*

$$\Pr[\ s' \notin \{s, \bot\} \mid s' \leftarrow \mathsf{User}^{\mathcal{A}(s,p,\mathsf{st})}(p, \mathsf{st}_0)\ ] \leq \epsilon$$

where $\mathsf{st} \leftarrow \mathsf{Init}(p,s)$*, and the probability goes over the randomness of all algorithms.*

Definition 4. *(robustness) A PPSS scheme on dictionary D and space S is* (T,ϵ)-robust *if for any $s \in S$, any $p \in D$, any set B s.t. $n - |\mathsf{B}| \geq t+1$, and any probabilistic algorithm $\mathcal{A}$ with running time T, we have*

$$\Pr[\ s' \neq s \mid s' \leftarrow \mathsf{User}^{\mathcal{A}(s,p,\mathsf{st_B}),\mathsf{Server}^{\diamond}(\mathsf{st}_{\overline{\mathsf{B}}})}(p, \mathsf{st}_0)\ ] \leq \epsilon$$

where $\mathsf{st} \leftarrow \mathsf{Init}(p,s)$*, and the probability goes over the randomness of all algorithms.*

3 Password-Protected Secret Sharing Secure under DDH

Notation: Before we describe our protocol we first introduce some notation: Let g be a generator of group G of prime order q. Let $y = g^x$ for[some $x \in \mathbb{Z}_q$. We denote the "shifted" ElGamal encryption under public key y as E_y, i.e. if $m \in \mathbb{Z}_q$ then $\mathsf{E}_y(m) = (g^r, y^r g^m)$ for $r \leftarrow \mathbb{Z}_q$, and we denote the textbook ElGamal encryption as $\mathsf{E'}_y$, i.e. if $m \in G$ then $\mathsf{E'}_y(m)$ outputs $c = (g^r, y^r m)$ for $r \leftarrow \mathbb{Z}_q$. We write $\mathsf{E}_y(m;r)$ or $\mathsf{E'}_y(m;r)$ when we want r to refer to the randomness in these encryptions. By convention we use c_m to denote a ciphertext that encrypts variable m, and we use r_m to denote the randomness in this ciphertext, e.g. $c_p = \mathsf{E}_y(p; r_p)$, $c_{\tilde{p}} = \mathsf{E}_y(\tilde{p}; r_{\tilde{p}})$, etc. Since ElGamal ciphertexts are (ordered) pairs of elements in G, we use $c_{m,L}$ and $c_{m,R}$ to denote respectively the first and the second element of c_m.

Protocol PPSS is a modification of a simple protocol based on threshold ElGamal encryption presented below. We note that while this basic protocol can be adopted to other homomorphic encryption schemes with threshold decryption, e.g. Paillier, its modification PPSS relies on special properties of ElGamal to make this protocol one round on-line.

Basic Protocol: Take dictionary $D = \mathbb{Z}_q$ and message space $S = G$. (Any other dictionary can be hashed into $\mathbb{Z}_q$ using a collision-resistant hash, and message space G can also be easily extended to standard message spaces consisting of bitstrings.) For $(p,s) \in \mathbb{Z}_q \times G$, procedure $\mathsf{Init}(p,s)$ picks an ElGamal private-public key-pair (x,y), secret-shares x among servers using a (t,n) Shamir secret sharing [Sha79] (i.e. $\mathsf{st}_j = x_j = f(j)$ where f is a random t-degree polynomial over $\mathbb{Z}_q$ s.t. $f(0) = x$), and as the public parameter st_0 it outputs the public key y, a shifted encryption $c_p = \mathsf{E}_y(p) = (g^{r_p}, y^{r_p} g^p)$ of password p, and a textbook encryption $c_s = \mathsf{E'}_y(s) = (g^{r_s}, y^{r_s} s)$ of secret s. The PPSS protocol proceeds as follows:

1. User sends $c_{\tilde{p}} = \mathsf{E}_y(\tilde{p})$ to each Server_j, where $\tilde{p} = p$ for a legitimate user.
2. Each Server_j randomizes $c_\delta = c_p / c_{\tilde{p}}$ as $c_{\beta_j} = (c_\delta)^{t_j}$ where $t_j \xleftarrow{r} \mathbb{Z}_q$, and sends c_{β_j} to User.
3. User computes $c_\beta = \prod_i c_{\beta_i}$ and sends c_β to each Server_j. Note that c_β is a shifted ElGamal encryption of $\beta = \sum_i \beta_i = \delta \cdot \sum_i t_i$ where $\delta = p - \tilde{p}$.
4. Each Server_j using its share x_j of ElGamal decryption key x computes a partial decryption of $c_\alpha = c_s \cdot c_\beta$ (where c_α is treated as a *textbook* ElGamal ciphertext), and sends it back to User. Note that in threshold ElGamal decryption [DF89] each Server_j simply returns $z_j = (c_{\beta,L})^{x_j}$, which User interpolates to compute $(c_{\beta,L})^x$ and hence to decrypt c_β.

Note that $\alpha = s \cdot g^{\beta} = s \cdot g^{\sum_i \beta_i} = s \cdot g^{(p-\tilde{p})\sum_i t_i}$, and therefore $\alpha = s$ if $\tilde{p} = p$ but α is random in G if $\tilde{p} \neq p$ and $\sum t_i$ is random in $\mathbb{Z}_q$.

This protocol is secure in the honest-but-curious setting under the DDH assumption on G, however in the presence of malicious parties a whole range of issues needs to be addressed: For example, since the protocol relies on the homomorphic property of encryption E, a malicious user can recover the secret by setting $c_{\tilde{p}}$ as a randomization of c_p, thus ensuring that $\tilde{p} = p$ without the knowledge of p. Also, a malicious server can cancel out honest servers' contributions c_{β_j} to c_β so that the sum $t = \sum_i t_i$ hits some known constant t, and thus there is effectively no randomization of the $\delta = \tilde{p} - p$ value, and the information the adversary receives as $\alpha = s \cdot g^{t(\tilde{p}-p)}$ allows the adversary to recover s in an off-line dictionary attack against p.

$\underline{\mathsf{Init}(p,s)}$ (on public parameters g, q, n, t and $ID_1, ..., ID_n$)

$x \xleftarrow{r} \mathbb{Z}_q$, $y \leftarrow g^x$,$\{x_i\}_{i=1}^{n} \xleftarrow{(n,t)} \mathsf{SS}(x)$, $h \xleftarrow{r} G$, $(r_p, r_s) \xleftarrow{r} (\mathbb{Z}_q)^2$,

$(c_{p,L}, c_{p,R}) \leftarrow (g^{r_p}, y^{r_p}h^p)$, $(c_{s,L}, c_{s,R}) \leftarrow (g^{r_s}, y^{r_s}s)$,

$\mathsf{st}_0 \leftarrow (g, y, h, c_p, c_s)$, $\{\mathsf{st}_i \leftarrow x_i\}_{i=1}^{n}$.

$\underline{\mathsf{User}(\mathsf{st}_0, \tilde{p}) \rightleftharpoons (\mathsf{Server}(\mathsf{st}_0, \mathsf{st}_1), \cdots, \mathsf{Server}(\mathsf{st}_0, \mathsf{st}_n))}$

S1 (Server_j) : Pick $t_j \xleftarrow{r} \mathbb{Z}_q$, compute $a_j \leftarrow g^{t_j}, b_j \leftarrow (c_{p,L})^{t_j}$. Send (ID_j, a_j, b_j) to User.

U1 (User) : Pick a set V of $t+1$ servers. Pick $r_{\tilde{p}} \xleftarrow{r} \mathbb{Z}_q$.
Compute $\{e_j \leftarrow (a_j)^{r_{\tilde{p}}}\}_{j \in V}$ and $c_{\tilde{p}} \leftarrow (g^{r_{\tilde{p}}}, y^{r_{\tilde{p}}}h^{\tilde{p}})$
Send $(\{ID_j, a_j, b_j, e_j\}_{j \in V}, c_{\tilde{p}})$ to Server_j, for each $j \in V$.

S2 (Server_j) : Compute: $\lambda_j \leftarrow \prod_{j' \in V/\{j\}} (-ID_{j'})/(ID_j - ID_{j'}) \pmod{q}$
$d_{\alpha,L,j} \leftarrow (c_{s,L} \cdot b_j/e_j \cdot \prod_{j' \in V/\{j\}} (b_{j'}/e_{j'}))^{\lambda_j \cdot x_{ID_j}}$.
$c_{\beta,R,j} \leftarrow (c_{p,R}/c_{\tilde{p},R})^{t_j}$ and $z_j \leftarrow c_{\beta,R,j} \cdot (d_{\alpha,L,j})^{-1}$
Send z_j to User.

U2 (User) : Output $s \leftarrow \prod_{j \in V} z_j \cdot c_{s,R}$.

Fig. 1. PPSS Simplified: Protocol for Honest-but-Curious Adversaries

Reducing Communication Rounds: On-Line Non-Interactive PPSS Protocol. We sketch how to take care of all active threats in the above basic protocol, and at the same time to reduce the round complexity of this protocol, by combining the distributed c_δ-randomization step, $c_\beta \leftarrow (c_\delta)^{\sum t_i}$, and the threshold decryption step, $\alpha \leftarrow \mathsf{TDec}(c_s \cdot c_\beta)$. The basic outline of the resulting protocol PPSS is described in Figure 1, followed by the full protocol presented in Figure 2.

To see how ciphertext randomization and threshold decryption can be combined in one step, note that in the threshold decryption step each P_j outputs value $z_j = (c_{s,L} \cdot c_{\beta,L})^{x_j} = (g^{r_s + \delta_r \sum_i t_i})^{x_j}$ where $\delta_r \triangleq r_p - r_{\tilde{p}}$. The user interpolates any $t+1$ of z_j's to compute $z = (g^{r_s + \delta_r \sum_i t_i})^{x} = y^{r_s} y^{\delta_r \sum_i t_i}$. The user also computes the $c_{\beta,R} = \prod_i c_{\beta_i,R} = y^{\delta_r \sum_i t_i} g^{\delta_p \sum_i t_i}$, and the decryption works because if $\delta_p = 0$ then $c_{\beta,R} = y^{\delta_r \sum_i t_i}$, and thus $s = c_{s,R} \cdot c_{\beta,R} \cdot (z)^{-1}$. Note that if the protocol was executed not by all n servers but by a subset V of $t+1$ servers, i.e. in particular if $c_\beta = \prod_{i \in V} c_{\beta_i}$, and if each P_j in V computed $z'_j = c_{\beta_j,R} \cdot (c_{s,L} \cdot c_{\beta,L})^{-\lambda_j x_j}$ where λ_j is a coefficient s.t. $x = \sum_{j \in V} \lambda_j x_j$, then U can still recover $s = c_{s,R} \cdot \prod_{j \in V} z'_j$. However, in order to let P_j compute z'_j in this way, each P_j needs to know two values: $c_{\tilde{p},R}$, to compute $c_{\beta_j,R} = (c_{p,R}/c_{\tilde{p},R})^{t_j}$, and $c_{\beta,L}$. U sends the $c_{\tilde{p},R}$ value to each Server_j as part of the basic protocol, but value $c_{\beta,L} = g^{\delta_r \sum_{i \in V} t_i}$ involves t_i's randomization values for all $i \in V$. What we can do, however, is for each P_j to first send to U a pair of "randomization commitment" values, $(a_j, b_j) = (g^{t_j}, (c_{p,L})^{t_j}) = (g^{t_j}, g^{r_p t_j})$, because then U can compute $c_{\beta,L} = \prod_{j\in V}(b_j \cdot (a_j)^{-r_{\tilde{p}}})$.

These observations allow us to reduce the total number of communication flows to three, with the *on-line* phase, i.e. between User contributing its password $\tilde{p}$ and recovering the secret s, taking only two communication flows (hence we say that our protocol is *non-interactive on-line*). The resulting protocol, secure for honest-but-curious adversaries, is shown in Figure 1. In Server's precomputation step, denoted as **S1**, each P_j picks $t_j \leftarrow \mathbb{Z}_q$ and sends $(a_j, b_j) = (g^{t_j}, (c_{p,L})^{t_j})$ to U. In step **U1**, on input $\tilde{p}$, user U chooses a set V of $t+1$ servers, computes $e_j = (a_j)^{r_{\tilde{p}}}$ for each $j \in V$, and forwards the set $\{(a_j, b_j, e_j)\}_{j\in V}$ to each P_j in V together with its password encryption $c_{\tilde{p}}$. In the final step **U2**, the user decrypts $s = c_{s,R} \cdot \prod_{j\in V} z_j$ if each P_j returns, as part of step **S2**,

$$z_j = c_{\beta_j,R} \cdot (d_{\alpha_j,L})^{-1} \text{ where } c_{\beta_j,R} = (c_{p,R}/c_{\tilde{p},R})^{t_j} \text{ and } d_{\alpha_j,L} = (c_{s,L} \cdot \prod_{j'\in V}(b_{j'}/e_{j'}))^{\lambda_j \cdot x_j}$$

The protocol in Figure 1 contains one more change compared to the "Basic Protocol" we explained first. Namely, the base used in the shifted ElGamal encryption is changed from g to an additional random base h in G. In other words, $E_y(m) = r_m, y^{r_m}h^m)$ instead of $(g^{r_m}, y^{r_m}h^m)$. The reason for this change is that otherwise for any $\delta_p = (p - \tilde{p})$ the user would recover $\alpha = g^{\delta_p \sum t_i} \cdot s$. However, unlike the basic protocol, the modified protocol reveals $a = \prod a_i = g^{\sum t_i}$ to the user. This would allow for an off-line dictionary attack where each p' can be tested by checking if $s' = (a)^{\tilde{p}-p'} \cdot \alpha$ looks like a valid plaintext s. This very attack is prevented by modifying the shifted ElGamal encryption so that $\mathsf{E}_y(m) = (g^r, y^r h^m)$, because under DDH the value $\alpha = h^{\delta_p \sum_i t_i}$ acts like a one-time pad even given $g^{\sum_i t_i}$.

Assuring Security against Malicious Adversary. To achieve security against malicious adversaries the full PPSS protocol in Figure 2 contains additional modifications to the above sketch. First, to prevent man-in-the-middle attacks we encrypt P_j's response z_j under the user's public key (step **S2**), and we bind this key to the user's message using labeled simulation-sound zero-knowledge proofs (step **U1**).

Second, in step **U1**, we incorporate an additional encryption of the shifted $\tilde{p}$ value into the user's message, namely $o_1 = (\bar{y}_1)r_{\tilde{p}}h_1^{\tilde{p}}$, using an additional base h_1 and an "encryption key" $\bar{y}_1$, both random elements in G. (The only party which holds the decryption key corresponding to $\bar{y}_1$ is the simulator.) We also include in the user's message value $o_2 = h_2^{r_{\tilde{p}}}$, for yet another random base element $h_2 \in G$. (This is not hiding user-created value $r_{\tilde{p}}$ in a semantically secure way, but it is good under DDH because h_2 is random in G and $r_{\tilde{p}}$ is random in $\mathbb{Z}_q$.) These two values allow the simulator to compute elements h_i^w where w is a witness in the user's message, i.e. $w = \tilde{p}$ or $w = r_{\tilde{p}}$, where h_i is a random base element. This allows the simulator to compute z^w for $w \in \{\tilde{p}, r_{\tilde{p}}\}$ and any challenge element $z \in G$, by setting the corresponding h_i base as z^α. It turns out that this ability to exponentiate any challenge element $z \in G$ to witnesses $\tilde{p}$ or $r_{\tilde{p}}$ contained in user's messages, is sufficient for the simulator to efficiently simulate the honest server's responses to user's messages, and it avoids having to extract the witnesses in user's statements. (Recall that the cheapest known concurrently extractable zero-knowledge proofs of knowledge in ROM, due to Fischlin [Fis05], multiply the protocol cost by a factor close to 10).

Third, to enable simulation of the user's algorithm without extraction of witnesses contained in server's messages, the user-side simulator utilizes an additional value $w_j = h_0^{t_j}$ which we add to the server's "precomputation" message (a_j, b_j) in step **S1**. Finally, both sides employ non-interactive simulation-sound zero-knowledge proofs (see a paragraph above) which ensure well-formedness of their messages, and in particular ensure that the $(h_i)^w$ values in each party's messages involve the true witness(es) w contained in this party's statement.

Non-Interactive Simulation-Sound Zero-Knowledge Proofs. As mentioned above, in order to assure the well-formed-ness of certain messages exchanged between the user and the servers, and hence achieve security against malicious adversaries, we use simulation-sound zero-knowledge non-interactive labeled proof

$\underline{\mathsf{Init}(p, s)}$ (on public parameters g, q, n, t and $ID_1, ..., ID_n$)

$x \xleftarrow{r} \mathbb{Z}_q$, $y \leftarrow g^x$, $\{x_i\}_{i=1}^n \xleftarrow{(n,t)} \mathsf{SS}(x)$, $(h, h_0, h_1, h_2, \bar{y}_1) \xleftarrow{r} (G)^5$, $(r_p, r_s) \xleftarrow{r} (\mathbb{Z}_q)^2$,

$c_p \leftarrow (g^{r_p}, y^{r_p} h^p)$, $c_s \leftarrow (g^{r_s}, y^{r_s} s)$, $\mathsf{st}_0 \leftarrow (g, y, h, h_0, h_1, h_2, \bar{y}_1, c_p, c_s)$, $\{\mathsf{st}_i \leftarrow x_i\}_{i=1}^n$.

$\underline{\mathsf{User}(\mathsf{st}_0, \tilde{p}) \rightleftharpoons (\mathsf{Server}(\mathsf{st}_0, \mathsf{st}_1), \cdots, \mathsf{Server}(\mathsf{st}_0, \mathsf{st}_n))}$

S1 (Server_j) : Pick $t_j \xleftarrow{r} \mathbb{Z}_q$, compute $a_j \leftarrow g^{t_j}, b_j \leftarrow (c_{p,L})^{t_j}, w_j \leftarrow h_0^{t_j}$, and
$\pi_{1j} \leftarrow \mathcal{P}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}]((a_j, b_j, w_j), t_j)$. Send $(ID_j, a_j, b_j, w_j, \pi_{1j})$ to User.

U1 (User) : Pick a set V of $t+1$ servers s.t. $\mathcal{V}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}]((a_j, b_j, w_j), \pi_{1j}) = 1$ for all $j \in V$.
Pick $r_{\tilde{p}} \xleftarrow{r} \mathbb{Z}_q$ and $(sk, pk) \xleftarrow{r} \mathcal{K}$.

Compute: $\{e_j \leftarrow (a_j)^{r_{\tilde{p}}}\}_{j \in V}$, $c_{\tilde{p}} \leftarrow (g^{r_{\tilde{p}}}, y^{r_{\tilde{p}}} h^{\tilde{p}})$, $(o_1, o_2) \leftarrow (\bar{y}_1^{r_{\tilde{p}}} h_1^{\tilde{p}}, h_2^{r_{\tilde{p}}})$
$\pi_2 \leftarrow \mathcal{P}^{\mathcal{H}}[\mathcal{L}_U^{\mathsf{st}_0}]((\{a_j, e_j\}_{j \in V}, c_{\tilde{p}}, o_1, o_2), (r_{\tilde{p}}, \tilde{p}), pk)$

$\forall j \in V$, send $(\{ID_{j'}, a_{j'}, b_{j'}, w_{j'}, e_{j'}, \pi_{1j'}\}_{j' \in V/\{j\}}, e_j, pk, c_{\tilde{p}}, o_1, o_2, \pi_2)$ to Server_j.

S2 (Server_j) : Stop if for any $j' \in V/\{j\}$, $\mathcal{V}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}]((a_{j'}, b_{j'}, w_{j'}), \pi_{1j'}) = 0$ or
$\mathcal{V}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}]((\{a_{j'}, e_{j'}\}_{j' \in V}, c_{\tilde{p}}, o_1, o_2), \pi_2, pk) = 0$.

Compute: $\lambda_j \leftarrow \prod_{j' \in V/\{j\}} (-ID_{j'})/(ID_j - ID_{j'}) \pmod q$
$d_{\alpha,L,j} \leftarrow (c_{s,L} \cdot b_j/e_j \cdot \prod_{j' \in V/\{j\}} (b_{j'}/e_{j'}))^{\lambda_j \cdot x_{ID_j}}$.
$c_{\beta,R,j} \leftarrow (c_{p,R}/c_{\tilde{p},R})^{t_j}$ and $z_j \leftarrow c_{\beta,R,j} \cdot (d_{\alpha,L,j})^{-1}$

Send $\zeta_j \leftarrow \mathcal{E}_{pk}(z_j)$ to User.

U2 (User) : For each $j \in V$, compute $z_j \leftarrow \mathcal{D}_{sk}(\zeta_j)$. Output $s \leftarrow \prod_{j \in V} z_j \cdot c_{s,R}$.

Fig. 2. PPSS: The Full Protocol

(NILP) system for two languages $\mathcal{L}_U$ and $\mathcal{L}_S$, both parameterized by the public parameters st_0 which are output by Init algorithm. Let G be a group of order q. Let $\mathsf{st}_0 = (g, y, h, h_0, h_1, h_2, \bar{y}_1, c_p, c_s)$ where $g, y, h, h_0, h_1, h_2, \bar{y}_1 \in G$ and $c_p, c_s \in (G)^2$. We define languages $\mathcal{L}_U^{\mathsf{st}_0}$ and $\mathcal{L}_S^{\mathsf{st}_0}$ as follows:

$$\begin{aligned}\mathcal{L}_U^{\mathsf{st}_0} &= \{(a_1, e_1, ..., a_{t+1}, e_{t+1}, c_{\tilde{p}}, o_1, o_2) \in (G)^{2t+6} \mid \exists\, (r_{\tilde{p}}, \tilde{p}) \in (\mathbb{Z}_q)^2 \textit{ s.t.} \\ &\qquad \{e_j = (a_j)^{r_{\tilde{p}}}\}_{j=1}^{t+1}, c_{\tilde{p}} = (y^{r_{\tilde{p}}} h^{\tilde{p}}, g^{r_{\tilde{p}}}), \text{and } (o_1, o_2) = (\bar{y}_1^{r_{\tilde{p}}} h_1^{\tilde{p}}, h_2^{r_{\tilde{p}}})\} \\ \mathcal{L}_S^{\mathsf{st}_0} &= \{(a, b, w) \in (G)^3 \mid \exists\, \alpha \in \mathbb{Z}_q \textit{ s.t. } a = g^\alpha, b = (c_{\tilde{p},L})^\alpha, \text{and } w = (h_0)^\alpha\}\end{aligned}$$

The idea of having *labeled* non-interactive proof (NILP) is to attach a label to the proven statement, similarly to how labeled encryption adds a label to a ciphertext. The simulation-soundness of a NILP system demands that if the simulator produces a NILP proof for any (wrong) statement x under label l then the adversary cannot produce a valid proof for x under any different label $l' \neq l$. We define the simulation-soundness and zero-knowledge properties of labeled NILP's in Appendix A. The constructions we provide for SS-ZK NILP's for both languages above are simple extensions of the well-known ROM-based non-interactive proof of knowledge of discrete-logarithm, and these constructions are included in Appendix A for completeness.

Standard Message Spaces, Soundness, Robustness, Efficiency. Protocol in Figure 2 works for a non-standard message space G, but this can be easily assuaged as follows: The public parameters string st_0 is amended by a random key k to a universal hash function H_k which maps elements of G onto 160-bit bitstrings in such a way that $H_k(x)$ is statistically indistinguishable from random 160-bit string if x is random in G. (Note also that such hashing is trivial in ROM, which our protocol assumes anyway for efficient NIZK's.) The PPSS initialization on password p and a secret bitstring s of any length would run the PPSS protocol of Figure 2 on random s' in G, and attach to st_0 an encryption of bitstring s under key $k' = H_k(s')$, using any symmetric semantically secure encryption scheme. The PPSS protocol would then proceed as in Figure 2 except that the user uses the recovered s' to compute key $k' = H_k(s')$ and decrypt secret s from the above ciphertext.

The PPSS protocol in Figure 2 also does not have the soundness or robustness properties. Recall that soundness requires that malicious Servers cannot make the User recover any other secret except of the one which was initially secret-shared, while robustness requires that recovery of the correct secret is assured if there are at least $t + 1$ honest Servers participating in the protocol. Both properties can be easily added to the protocol. Soundness can be added by including a public key of an existentially unforgeable signature scheme in st_0, using the corresponding signature key to sign secret s and running the above modified PPSS algorithm on s concatenated with this signature. The protocol then proceeds as above except that at the end the user parses the recovered secret as $s|\sigma$ and outputs s only if σ is a valid signature on it under the key in st_0. It's easy to see that this makes the scheme (T, ϵ) robust given a (one-time) signature scheme which is unforgeable by T-bounded adversary except for probability ϵ. Note that these modifications have a minimal impact on protocol efficiency. Robustness can be achieved using Feldman verification values to the initial secret-sharing, i.e. extending st_0 with a set of values $\{y_i = g^{x_i}\}_{i=1}^{n}$), and adding a proof of well-formedness to the final server's message **S2**. This proof is a standard Schnorr-like zero-knowledge proof of equality of representations which shows that $z_j = \alpha^{t_j}\beta^{x_{ID_j}}$ where α, β are elements in G computed (by both parties) as in step **S2**, and exponents t_j and x_{ID_j} are the same as in equations $a_j = g^{t_j}$ and $y_j = g^{x_{ID_j}}$. (The NIZK proof for this statement works along the same lines as the proofs in Appendix A, and it costs only 3 additional exponentiations for both parties.) This makes the protocol robust if User who detects some misbehaving parties among its chosen set V just restarts the PPSS protocol using a set of players that excludes these misbehaving parties, although this may take $O(t)$ rounds in the worst case. The resulting protocol is (T, ϵ)-robust where ϵ is the total soundness error of the Server-side NIZK proofs when faced with a T-time limited adversary.

Assuming no active faults (which trigger protocol re-start as explained above), the efficiency of the PPSS protocol including these modifications is $4t + 15$ (multi)exponentiations per Server and $8t + 16$ (multi)exponentiations per User, assuming that the proof systems are instantiated in ROM as shown in Appendix A. However, $4t + 5$ of Servers' exponentiations are in verification of the NIZK's in step **S2**, whose cost is reduced by a factor of at least 4 using batch verification techniques, e.g. [BGR98], resulting in about $t + 11$ (multi)exponentiations. Similarly $6(t + 1)$ exponentiations for the User come from two NIZK verification steps, $3(t+1)$ exponentiations each, whose cost can be reduced by a factor of 4 again, resulting in User's costs of about $3.5t + 12$ (multi)exponentiations.

Theorem 1. *Let $t < n$, let G be a group of prime order q where the DDH problem is $(T_{ddh}, \epsilon_{ddh})$-hard, let $t_{\exp}$ be the time of full exponentiation in G, let $\Sigma = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public key encryption scheme which is $(T_{cpa}, \epsilon_{cpa})$-indistinguishable, and let $\Pi^{\mathcal{H}}[\mathcal{L}_U^{\mathsf{st}_0}]$ and $\Pi^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}]$ be labeled proof systems which are respectively $(T_U, q_P^U, q_H^U, \epsilon_{ZK}^U, \epsilon_{SS}^U)$ and $(T_S, q_P^S, q_H^S, \epsilon_{ZK}^S, \epsilon_{SS}^S)$ simulation-sound zero-knowledge in ROM. The* PPSS *protocol in Figure 2 is $(n, t, q_U, q_S, T, \epsilon)$-secure where $q_U \le q_P^U$, $q_S \le q_P^S/n$, and:*

$$T \le \min\{T_{cpa}, T_{ddh}\} - (T_U + T_S) - [11 + (20n + 22)q_S + (3n + 22)q_U]t_{exp}$$
$$\epsilon \le (2q_U + 8)\epsilon_{ddh} + 2nq_S\epsilon_{cpa} + 6\epsilon_{ZK}^S + 2\epsilon_{ZK}^U + 6q_S\epsilon_{SS}^U$$

Note that Appendix A provides very efficient instantiations of the two proof systems used in PPSS, secure in the Random Oracle Model. The simulation-soundness and zero-knowledge bounds for these proofs, given an adversary making q_H hash function queries, are $\epsilon_{ZK} = q_P \cdot q_H/q$ and $\epsilon_{SS} = 1/q$, for both U and S, while simulation times are $T_U = ((t + 6)q_P^U) \cdot t_{exp}$ and $T_S = (3q_P^S) \cdot t_{exp}$.

Proof. Let $\mathcal{A}$ be an interactive algorithm followed by an adversary attacking the PPSS scheme while interacting with at most q_U user and q_S server sessions and corrupting servers $\{\mathsf{P}_i\}_{i\in\mathsf{B}}$ for some set B s.t. $|\mathsf{B}| = t' \leq t$. Figure 3 describes a series of games, $\mathsf{G}_0, ..., \mathsf{G}_8$, all initialized on secret s, where G_0 models the interaction of $\mathcal{A}$ with the PPSS scheme i.e. $\mathcal{A}^{\mathsf{User}^\diamond(p,\mathsf{st}_0),\mathsf{Server}^\diamond(\mathsf{st}_{\overline{\mathsf{B}}})}(\mathsf{st}_0, \mathsf{st}_\mathsf{B})$ where $p \leftarrow D$ and $\mathsf{st} \leftarrow \mathsf{Init}(p, s)$, and games $\mathsf{G}_1, ..., \mathsf{G}_8$ are modifications of G_0 used in the security argument. We also define a crucial event F as shown in figure 3, which roughly corresponds to an adversary $\mathcal{A}$ sending an encryption of the correct password to a group of at least $t - t' + 1$ distinct servers. The goal of the security argument is to show that for any two secrets s_0, s_1,

$$|p_0(s_1) - p_0(s_0)| \leq \lfloor \frac{q_S}{t - t' + 1} \rfloor * \frac{1}{|D|} + \epsilon$$

where ϵ is some negligible quantity. This would follow if we showed that $|p_0(s_1) - p_0(s_0)| \leq \Pr[\mathsf{F}] + \epsilon_1$ and $\Pr[\mathsf{F}] \leq \lfloor \frac{q_S}{t-t'+1} \rfloor * \frac{1}{|D|} + \epsilon_2$ where ϵ_1, ϵ_2 are negligible. Technically the proof we show below contains a few extra steps but this is its most intuitive outline.

Modeling the PPSS Attack: Let us first look closer at how the game G_0 models the interaction of $\mathcal{A}$ with the $\mathsf{User}^\diamond(p, \mathsf{st}_0)$ and $\mathsf{Server}^\diamond(\mathsf{st}_{\overline{\mathsf{B}}})$ oracles in an attack on a PPSS scheme. The $\mathsf{Init}^\diamond$ procedure assumes the discrete-log setting of our PPSS scheme, i.e. generator g of group G of prime order q, and it treats threshold (t, n), the set of corrupted players B, and the dictionary $D \subseteq \mathbb{Z}_q$ as parameters. Moreover, it assumes certain assignment between all possible server sessions and the identities of the servers executing these sessions. Note that adversary $\mathcal{A}$ can engage in at most q_S sessions but it's up to $\mathcal{A}$ to decide which servers will be involved in these sessions. Therefore in the security game we prepare q_S distinct sessions for every n servers, thus $n \cdot q_S$ total sessions, even though a q_S-limited adversary will utilize only q_S of them. For notational ease we fix the identity of a server executing the j-th session, where $j = 1, .., nq_S$, as P_{ID_j} where $ID_j = (j \bmod n) + 1$. Since our PPSS protocol starts from Servers' messages to the Client, without loss of generality we let the $\mathsf{Init}^\diamond$ procedure prepare the first message from all these nq_S sessions, which it then passes to adversary $\mathcal{A}$.

Procedure $\mathsf{Init}^\diamond$ on input s picks $p \overset{r}{\leftarrow} D$ and generates the vector of initial states st, including all the public values in st_0 and the shares $st_i = x_i$ for each server P_i. Note that in game G_0, which models the real execution, procedure $\mathsf{Init}^\diamond$ generates these values with exactly the same distribution as the real initialization procedure $\mathsf{Init}(p, s)$. As a result the adversary $\mathcal{A}$ receives st_0, initial states of the corrupted servers $\mathsf{st}_B = \{x_i\}_{i\in B}$, and a Server's first message on all nq_S sessions. Afterwards $\mathcal{A}$ can make q_U queries to the $\mathsf{User}^\diamond$ oracle, which models execution of procedure $\mathsf{User}(\mathsf{st}_0, p)$ on a set of messages, specified by the adversary, which look like the initial messages from some $t + 1$ servers. $\mathcal{A}$ can also make q_S queries to the $\mathsf{Server}^\diamond_j$ oracle, for q_S *distinct* values j, which models execution of the Server procedure on j-th session. The variables used in the description of $\mathsf{Server}^\diamond_j$ algorithm consist of inputs received from the adversary, the public values $\mathsf{st}_0 = (t, g, y, h, h_0, h_1, h_2, \bar{y}_1, c_p, c_s)$, the share x_{ID_j} of server ID_j in whose name this session is executed, and the local variables (t_j, a_j, b_j) created for the j-th session by the $\mathsf{Init}^\diamond$ procedure. All the inputs that $\mathsf{User}^\diamond$ and $\mathsf{Server}^\diamond_j$ oracles receive from the adversary are freely chosen by the adversarial algorithm $\mathcal{A}$, but in Figure 3 we denote some of these values as "primed", for example values $ID'_{j'}, a'_{j'}, b'_{j'}, \Pi'_{1j'}$ in the inputs to the $\mathsf{Server}^\diamond_j$ oracle, so that to stress that they are potentially different from the global constants ID_j or values a_j, b_j, Π_{1j} created by the $\mathsf{Init}^\diamond$ procedure. Similarly we denote all inputs to the $\mathsf{User}^\diamond$ oracle with primes to stress that these can be different values from the corresponding values output by $\mathsf{Init}^\diamond$.

Proof Roadmap: In the rest of the proof we will use the following notation: By $\mathsf{F}_{i,s}$ we denote event F happening in the interaction between $\mathcal{A}$ and game G_i initialized on secret s; by $p_i(s)$ we denote $\Pr[1 \leftarrow (\mathcal{A} \rightleftharpoons \mathsf{G}_i(s))]$; and by $p_i^{\neg\mathsf{F}}(s)$ we denote the joint probability $\Pr[1 \leftarrow (\mathcal{A} \rightleftharpoons \mathsf{G}_i(s)) \wedge, \neg\mathsf{F}_{i,s}]$. Let t_{exp} be a time of a single (multi)-exponentiation in G, let the DDH problem be $(T_{ddh}, \epsilon_{ddh})$-hard in group G, let the encryption scheme $\mathcal{E} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be $(T_{cpa}, \epsilon_{cpa})$-indistinguishable, and let labeled proof systems $\Pi^{\mathcal{H}}[\mathcal{L}_U^{\mathsf{st}_0}] =$

$(\mathcal{P}^{\mathcal{H}}[\mathcal{L}_U^{\mathsf{st}_0}], \mathcal{V}^{\mathcal{H}}[\mathcal{L}_U^{\mathsf{st}_0}])$ and $\Pi^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}] = (\mathcal{P}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}], \mathcal{V}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}])$ be respectively $(T_U, q_P^U, q_H^U, \epsilon_{ZK}^U, \epsilon_{SS}^U)$ and $(T_S, q_P^S, q_H^S, \epsilon_{ZK}^S, \epsilon_{SS}^S)$ simulation-sound zero-knowledge. Assume that $q_U \leq q_p^U$, $q_S \leq q_p^S/n$, and $T_{\mathcal{A}} \leq \min\{T_{cpa}, T_{ddh}\} - T_U - T_S - [11 + (20n+22)q_S + (3n+22)q_U]t_{exp}$. We will show the following facts, where $\epsilon_{0,3} = (q_U+1)\epsilon_{ddh} + nq_S\epsilon_{cpa} + 2\epsilon_{ZK}^S + \epsilon_{ZK}^U + q_S\epsilon_{SS}^U$, $\epsilon_{4,5} = \epsilon_{ddh}$, and $\epsilon_{4,8} = 4\epsilon_{ddh} + 2\epsilon_{ZK}^S + q_S\epsilon_{SS}^U$.

Claim 1 $|p_0(s) - p_3(s)| \leq \epsilon_{0,3}$, *for all* s

Claim 2 $|p_3(s) - p_3(s')| \leq |p_4^{\neg\mathsf{F}}(s) - p_4^{\neg\mathsf{F}}(s')| + \max(\Pr[\mathsf{F}_{4,s}], \Pr[\mathsf{F}_{4,s'}])$, *for all* s, s'

Claim 3 $\Pr[\mathsf{F}_{4,s}] \leq \lfloor \frac{q_S}{t-t'+1} \rfloor * \frac{1}{|D|} + \epsilon_{4,8}$, *for all* s

Claim 4 $|p_4^{\neg\mathsf{F}}(s) - p_4^{\neg\mathsf{F}}(s')| \leq 2\epsilon_{4,5}$, *for all* s, s'

Summing these inequalities, with the first instantiated first for s and then for s', we obtain an inequality which completes the proof:

$$|p_0(s) - p_0(s')| \leq 2(\epsilon_{0,3} + \epsilon_{4,5}) + \epsilon_{4,8}$$

Claim 1 follows from claims 5-7 below, because $\epsilon_{0,3}$ is the sum of the upper bounds on the distances $|p_{i-1}(s) - p_i(s)|$ shown for $i = 1, 2, 3$ in claims 5-7. (The proofs of all the technical claims relating adversary's view between subsequent games are removed to Appendix B.) Similarly claim 3 follows from claims 9-12 below (the proof to these claims are in section Appendix B), because $\epsilon_{4,8}$ is the sum of the upper bounds on the distances $|\Pr[\mathsf{F}_{i-1,s}] - \Pr[\mathsf{F}_{i,s}]|$ which are shown for $i = 5, 6, 7, 8$ in claims 9-12. Claim 4 follows from claim 9 below because game G_5 is independent of secret s, and therefore for every s, s' we have that $p_5^{\neg\mathsf{F}}(s) = p_5^{\neg\mathsf{F}}(s')$. Finally claim 2 follows from claim 8 below: Note first that $|p_3(s) - p_3(s')| = |(p_3^{\neg\mathsf{F}}(s) + p_3^{\mathsf{F}}(s)) - (p_3^{\neg\mathsf{F}}(s') + p_3^{\mathsf{F}}(s'))|$ can be upper bounded by $|p_3^{\neg\mathsf{F}}(s) - p_3^{\neg\mathsf{F}}(s')| + \max(\Pr[\mathsf{F}_{3,s}], \Pr[\mathsf{F}_{3,s'}])$. Secondly, since claim 8 shows that $p_3^{\neg\mathsf{F}}(s) = p_4^{\neg\mathsf{F}}(s)$ and $\Pr[\mathsf{F}_{3,s}] = \Pr[\mathsf{F}_{4,s}]$ for all s, this upper bound is equal to $|p_4^{\neg\mathsf{F}}(s) - p_4^{\neg\mathsf{F}}(s')| + \max(\Pr[\mathsf{F}_{4,s}], \Pr[\mathsf{F}_{4,s'}])$.

Claim 5 *Games* G_0 *and* G_1 *are indistinguishable under DDH assumption. Concretely, for any secret* s*, if* $q_U \leq q_P^U$*, and* $T_{\mathcal{A}} \leq T_{ddh} - T_U - [7 + (11n+9)q_S + (5n+12)q_U]t_{exp}$ *then* $|p_0(s) - p_1(s)| \leq q_U\epsilon_{ddh} + \epsilon_{ZK}^U$.

Claim 6 *Games* G_1 *and* G_2 *are indistinguishable assuming the* $\Sigma = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *encryption scheme is IND-CPA. Concretely, for any secret* s*, if* $T_{\mathcal{A}} \leq T_{cpa} - T_U - [6 + (11n+9)q_S + (5n+12)q_U]t_{exp}$ *then* $|p_1(s) - p_2(s)| \leq nq_S\epsilon_{cpa}$.

Claim 7 *Games* G_2 *and* G_3 *are indistinguishable under DDH assumption. Concretely, for any secret* s*, if* $q_S \leq q_P^S/n$*,* $q_U \leq q_P^U$*, and* $T_{\mathcal{A}} \leq T_{ddh} - T_U - T_S - [6 + (14n+12)q_S + (5n+12)q_U]t_{exp}$ *then* $|p_2(s) - p_3(s)| \leq \epsilon_{ddh} + 2\epsilon_{ZK}^S + q_S\epsilon_{SS}^U$.

Claim 8 *For any secret* s*,* $p_3^{\neg\mathsf{F}}(s) = p_4^{\neg\mathsf{F}}(s)$.

Claim 9 *Games* G_4 *and* G_5 *are indistinguishable under DDH assumption. Concretely, for any secret* s*, if* $T_{\mathcal{A}} \leq T_{ddh} - T_U - [3 + (14n+12)q_S + (5n+12)q_U]t_{exp}$ *then* $|p_4^{\neg\mathsf{F}}(s) - p_5^{\neg\mathsf{F}}(s)| \leq \epsilon_{ddh}$ *and* $|\Pr[\mathsf{F}_{4,s}] - \Pr[\mathsf{F}_{5,s}]| \leq \epsilon_{ddh}$.

Claim 10 *Games* G_5 *and* G_6 *are indistinguishable under DDH assumption. Concretely for any secret* s*, if* $q_S \leq q_P^S/n$*,* $q_U \leq q_P^U$*, and* $T_{\mathcal{A}} \leq T_{ddh} - T_U - T_S - [3 + (14n+17)q_S + (5n+12)q_U]t_{exp}$*, then* $|p_5(s) - p_6(s)| \leq nq_S\epsilon_{ddh} + 2\epsilon_{ZK}^S + q_S\epsilon_{SS}^U$ *and* $|\Pr[\mathsf{F}_{5,s}] - \Pr[\mathsf{F}_{6,s}]| \leq nq_S\epsilon_{ddh} + 2\epsilon_{ZK}^S + q_S\epsilon_{SS}^U$.

Claim 11 *Games* G_6 *and* G_7 *are indistinguishable under DDH assumption. Concretely, for any secret* s*, if* $T_{\mathcal{A}} \leq T_{ddh} - T_U - [5 + (14n+12)q_S + (5n+20)q_U)t_{exp}$ *then* $|\Pr[\mathsf{F}_{6,s}] - \Pr[\mathsf{F}_{7,s}]| \leq \epsilon_{ddh} + \epsilon_{UU}^S$.

```
$\underline{\mathsf{Init}^{\diamond}(\text{on input } s)}$

$x \xleftarrow{r} \mathbb{Z}_q, y \leftarrow g^x, \{x_i\}_{i=1}^n \xleftarrow{(n,t)} \mathsf{SS}(x)$    $\boxed{\mathsf{G}_0 - \mathsf{G}_3}$
$y \xleftarrow{r} G, \{x_i\}_{i=1}^n \xleftarrow{(n,t)} \mathsf{SS}(0)$    $\boxed{\mathsf{G}_4 - \mathsf{G}_8}$
$(h, h_0, h_1, h_2) \xleftarrow{r} (G)^4$    $\boxed{\mathsf{G}_0 - \mathsf{G}_8}$
$\bar{y}_1 \xleftarrow{r} G$    $\boxed{\mathsf{G}_0 - \mathsf{G}_2}$
$\bar{x}_1 \xleftarrow{r} \mathbb{Z}_q, \bar{y}_1 \leftarrow g^{\bar{x}_1}$    $\boxed{\mathsf{G}_3 - \mathsf{G}_8}$
$r_s \xleftarrow{r} \mathbb{Z}_q, c_s \leftarrow (g^{r_s}, y^{r_s} s)$    $\boxed{\mathsf{G}_0 - \mathsf{G}_4}$
$c_s \xleftarrow{r} (G)^2$    $\boxed{\mathsf{G}_5 - \mathsf{G}_8}$
$p \leftarrow D, r_p \xleftarrow{r} \mathbb{Z}_q, c_p \leftarrow (g^{r_p}, y^{r_p} h^p)$    $\boxed{\mathsf{G}_0 - \mathsf{G}_7}$
$c_p \xleftarrow{r} (G)^2$    $\boxed{\mathsf{G}_8}$
$\mathsf{idSet} \leftarrow \{\}, \mathsf{pkSet} \leftarrow \{\}, \mathsf{F} \leftarrow \mathsf{false}$    $\boxed{\mathsf{G}_0 - \mathsf{G}_8}$
$\{t_j \xleftarrow{r} \mathbb{Z}_q, (a_j, b_j, w_j) \leftarrow (g^{t_j}, (c_{p,L})^{t_j}, h_0^{t_j}), \pi_{1j} \leftarrow \mathcal{P}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}](t_j, (a_j, b_j, w_j))\}_{j=1}^{nq_S}$
$\mathsf{st}_0 \leftarrow (g, y, h, \{h_i\}_{i=0}^2, \bar{y}_1, c_p, c_s), \{\mathsf{st}_i \leftarrow x_i\}_{i=1}^n, \mathsf{Ret}(\mathsf{st}_0, \{\mathsf{st}_i\}_{i \in \mathsf{B}}, \{a_j, b_j, w_j, \pi_{1j}\}_{j=1}^{nq_S})$

$\underline{\mathsf{Server}_j^{\diamond}(pk', \{(ID'_{j'}, a'_{j'}, b'_{j'}, e_{j'}, \pi'_{1j'})\}_{j'=1}^t, e_j, c_{\tilde{p}}, \{o_i\}_{i=1}^2, \pi_2)}$

If $(\exists_{1 \le j' \le t}$ st $\mathcal{V}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}]((a'_{j'}, b'_{j'}, w'_{j'}), \pi'_{1j'}) = 0) \vee$    $\boxed{\mathsf{G}_0 - \mathsf{G}_8}$
  $(\mathcal{V}^{\mathcal{H}}[\mathcal{L}_U^{\mathsf{st}_0}]((\{(a'_{j'}, e_{j'})\}_{j'=1}^t \cup \{(a_j, e_j)\}, c_{\tilde{p}}, \{o_i\}_{i=1}^2), \pi_2, pk') = 0)$ then ABORT this session.
If $(o_1/(c_{\tilde{p},L})^{\bar{x}_1} = h_1^p)$ then $\mathsf{idSet} \leftarrow \mathsf{idSet} \cup \{ID_j\}$
If $(o_1/(c_{\tilde{p},L})^{\bar{x}_1} = h_1^p \wedge |\mathsf{idSet}| > t - t')$ then set event $\mathsf{F} \leftarrow \mathsf{true}$
$\lambda_j \leftarrow \prod_{j'=1}^t ((-ID'_{j'})/(ID_j - ID'_{j'})), d_{\alpha,L,j} \leftarrow (c_{s,L} \cdot (b_j/e_j) \cdot \prod_{j'=1}^t (b'_{j'}/e'_{j'}))^{\lambda_j \cdot x_{ID_j}}$
$c_{\beta,R,j} \leftarrow (c_{p,R}/c_{\tilde{p},R})^{t_j}$    $\boxed{\mathsf{G}_0 - \mathsf{G}_2}$
if $(o_1/(c_{\tilde{p},L})^{\bar{x}_1} = h_1^p)$ then $c_{\beta,R,j} \leftarrow (c_{p,R}/c_{\tilde{p},R})^{t_j}$    $\boxed{\mathsf{G}_3 - \mathsf{G}_5}$
else $c_{\beta,R,j} \xleftarrow{r} G$
$c_{\beta,R,j} \xleftarrow{r} G$    $\boxed{\mathsf{G}_6 - \mathsf{G}_8}$
$z_j \leftarrow c_{\beta,R,j} \cdot (d_{\alpha,L,j})^{-1}$    $\boxed{\mathsf{G}_0 - \mathsf{G}_1}$
$\mathsf{Ret}(\mathcal{E}_{pk'}(z_j))$
If $(pk' \in \mathsf{pkSet})$ then $\mathsf{Ret}(\mathsf{E}_{pk'}(1))$    $\boxed{\mathsf{G}_2 - \mathsf{G}_8}$
else $z_j \leftarrow c_{\beta,R,j} \cdot (d_{\alpha,L,j})^{-1}$, $\mathsf{Ret}(\mathcal{E}_{pk'}(z_j))$

$\underline{\mathsf{User}^{\diamond}(\{ID'_j, a'_j, b'_j, w'_j, \pi'_{1j}\}_{j=1}^t)}$

If $(\exists_{1 \le j \le t}$ st $\mathcal{V}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}]((a'_j, b'_j, w'_j), \pi'_{1j}) = 0)$ then ABORT this session.    $\boxed{\mathsf{G}_0 - \mathsf{G}_8}$
$r_{\tilde{p}}, (sk, pk) \xleftarrow{r} \mathcal{K}$, $\mathsf{pkSet} \leftarrow \mathsf{pkSet} \cup \{pk\}$, $\{e_j \leftarrow {a'_j}^{r_{\tilde{p}}}\}_{j=1}^t$
$o_1 \leftarrow \bar{y}_1^{r_{\tilde{p}}} h_1^{\tilde{p}}, o_2 \leftarrow h_2^{r_{\tilde{p}}}$    $\boxed{\mathsf{G}_0}$
$o_1 \xleftarrow{r} G, o_2 \leftarrow h_2^{r_{\tilde{p}}}$    $\boxed{\mathsf{G}_1 - \mathsf{G}_8}$
$c_{\tilde{p}} \leftarrow (g^{r_{\tilde{p}}}, y^{r_{\tilde{p}}} h^p)$    $\boxed{\mathsf{G}_0 - \mathsf{G}_6}$
$c_{\tilde{p},R} \xleftarrow{r} G, c_{\tilde{p}} \leftarrow (g^{r_{\tilde{p}}}, c_{\tilde{p},R})$    $\boxed{\mathsf{G}_7 - \mathsf{G}_8}$
$\pi_2 \leftarrow \mathcal{P}^{\mathcal{H}}[\mathcal{L}_U^{\mathsf{st}_0}]((\{a'_j, e_j\}_{j \in V}, c_{\tilde{p}}, \{o_i\}_{i=1}^2), (r_{\tilde{p}}, p, r_o), pk)$, $\mathsf{Ret}(\{e_j\}_{j=1}^t, pk, c_{\tilde{p}}, \{o_i\}_{i=1}^2, \pi_2)$    $\boxed{\mathsf{G}_0}$
$\pi_2 \leftarrow \mathcal{S}[\mathcal{L}_U^{\mathsf{st}_0}]((\{a'_j, e_j\}_{j \in V}, c_{\tilde{p}}, \{o_i\}_{i=1}^2), pk)$, $\mathsf{Ret}(\{e_j\}_{j=1}^t, pk, c_{\tilde{p}}, \{o_i\}_{i=1}^2, \pi_2)$    $\boxed{\mathsf{G}_1 - \mathsf{G}_8}$
```

Fig. 3. Games G_0,G_1,...,G_8 used in the security proof of the PPSS protocol

Claim 12 *Games* G_7 *and* G_8 *are indistinguishable under DDH assumption. Concretely, for any secret* s*, if* $T_{\mathcal{A}} \leq T_{ddh} - T_U - [1 + (14n + 12)q_S + (5n + 12)q_U]t_{exp}$ *then* $|\Pr[\mathsf{F}_{7,s}] - \Pr[\mathsf{F}_{8,s}]| \leq \epsilon_{ddh}$.

Adding Strong Security. Theorem 1 shows that the PPSS protocol in figure 2 satisfies the basic security property defined in definition 1, but it is easy to modify the protocol to achieve the *strong security* notion defined in definition 2 at little additional cost. Consider the protocol of figure 2 modified as explained in the paragraph on soundness and robustness on page 10. Note that in this modified protocol all server's actions are accompanied by proofs of well-formedness, and therefore is it easy for the simulator to verify whether or not a particular User instance of the PPSS protocol will reconstruct some secret or reject, because that's dependent only on verification of the NIZK's. Furthermore, it the soundness of these NIZK proof systems implies that once the User instance accepts the NIZK's then, except for NIZK soundness error(s), it successfully outputs the originally shared secret. This is the case because encryption c_s, the key y, and the private-key encryption of the secret concatenated with its signature, are all part of the (trusted) public string st_0, and they are all committing to s and the real shared secret encrypted under the key derived from s via H_k key (see page 10). Therefore it is easy for the simulator in the proof of theorem 1 above to decide whether or not a particular User instance accepts (and, by correctness, outputs the correct shared secret), or rejects. Technically, the proof adopt the proof of theorem 1 to a proof of strong security, the sequence of games should be amended by a game in which the challenger picks the (private,public) key pair of an encryption scheme on behalf of each User oracle, and it determines the "accept/reject" bit which User oracle should send back to $\mathcal{A}$ based on verifying the NIZK proofs in all messages $\mathcal{A}$ sent to this User instance, including the messages in round **S2**, decrypted using the above-mentioned private key.

4 Efficient T-PAKE from PPSS

A password protected secret sharing (PPSS) scheme can be used as a black box to achieve a threshold password authenticated key exchange (T-PAKE) protocol (in the public key model) at very little additional cost. In particular, the round complexity of the resulting T-PAKE is the same as the PPSS because all the T-PAKE messages can be piggybacked onto the PPSS protocol flows. Figure 4 shows a secure T-PAKE protocol assuming that $\mathcal{E} = (\mathsf{EKg}, \mathsf{Enc}, \mathsf{Dec})$ is a chosen-ciphertext-attack secure public-key encryption, $\mathcal{S} = (\mathsf{SKg}, \mathsf{Sign}, \mathsf{Vrfy})$ is a signature scheme which is existentially unforgeable under chosen message attack, and $\mathsf{PPSS} = (\mathsf{PPSS.Init}, \mathsf{PPSS.User}, \mathsf{PPSS.Server})$ is a strongly secure password protected secret sharing protocol. We assume that PPSS consists of m rounds, and we denote the output of PPSS.User procedure in round k as M^k_{User} and the output of procedure PPSS.Server run by Server_j in round k as $M^k_{\mathsf{Server}_j}$. Note that the Server and Client procedures of T-PAKE protocol just follows the Server and User procedures of underlying PPSS scheme for rounds $2 \leq k \leq (m - 1)$. We relegate a formal proof that this construction implies a T-PAKE to the final version of the paper, but, very briefly, the signatures and CCA encryption scheme ensure that the network adversary cannot re-route messages from a session in which honest players are involved, or modify them in any way, and hence in particular all User sessions are independent of each other. Then by the security of the PPSS scheme, except for $\lfloor q_s/(t - t' + 1)\rfloor \cdot (1/|D|)$ probability, the view of the PPSS protocol initialized with the real decryption key sk is indistinguishable from a view where sk is replaced by an independent key, in which case CCA security of encryption ensures that $\mathcal{A}$ gets no information about any unrevealed session keys even given a capability to reveal any other session keys (handled by decryption queries in a reduction to CCA encryption security).

References

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT*, pages 236–255, 1998.

$\underline{\mathsf{Init}(p)}$ (on public parameters κ, n, t)

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{EKg}$, $\{(\mathsf{ssk}_i, \mathsf{vk}_i) \leftarrow \mathsf{SKg}\}_{i=1}^{n}$, $(\tilde{\mathsf{st}}_0, \tilde{\mathsf{st}}_0, ..., \tilde{\mathsf{st}}_n) \leftarrow \mathsf{PPSS.Init}(p, \mathsf{sk})$

$\mathsf{st}_0 \leftarrow (\tilde{\mathsf{st}}_0, \mathsf{pk}, \{\mathsf{vk}_i\}_{i=0}^{n})$, $\{\mathsf{st}_i \leftarrow (\tilde{\mathsf{st}}_i, \mathsf{ssk}_i)\}_{i=1}^{n}$

$\underline{\mathsf{Client}(\mathsf{st}_0, \tilde{p}, uId) \rightleftharpoons (\mathsf{Server}(\mathsf{st}_0, \mathsf{st}_1), \cdots, \mathsf{Server}(\mathsf{st}_0, \mathsf{st}_n))}$

C1 (Client) : Pick $sId \overset{r}{\leftarrow} \{0,1\}^{\kappa}$. Send (uId, sId) to each Server.
Run the interactive algorithm $\mathsf{PPSS.User}(s\tilde{t}_0, \tilde{p})$ with the Servers.

S1 (Server$_j$) : Run the interactive algorithm $\mathsf{PPSS.Server}_j(\tilde{\mathsf{st}}_0, \tilde{\mathsf{st}}_j)$ with the Client.
If this PPSS instance aborts then assign $k_j \leftarrow \perp$ on session sId and stop.
Otherwise pick $k_j \overset{r}{\leftarrow} \{0,1\}^{\kappa}$. Compute $\sigma_j \leftarrow \mathsf{Sign}(\mathsf{ssk}_{Id_j}, uId, sId, k_j)$ and $e_j \leftarrow \mathsf{Enc}(\mathsf{pk}, (k_j, \sigma_j))$.
Send e_j to Client.

C2 (Client) : Let $\hat{sk}$ be the Client's local output of the PPSS instance. If $\hat{sk} = \perp$ then set $k_j \leftarrow \perp$ for all j and stop.
Let $(\hat{k_j}, \hat{\sigma_j}) \leftarrow \mathsf{Dec}(\hat{\mathsf{sk}}, e_j)$ for all servers P_j which sent their e_j's to the Client.
For each j, if $\mathsf{Vrfy}(\mathsf{vk}_{Id_j}, \hat{\sigma_j}, (uId, sId, \hat{k_j})) = 1$ then output $\hat{k_j}$ on session sId with server j,
o/w output $\perp$.

[BJKS03] John Brainard, Ari Juels, Burt Kaliski, and Michael Szydlo. Nightingale: A new two-server approach for authentication with short secrets. In *12th USENIX Security Symp*, pages 201–213. IEEE Computer Society, 2003.

[BMP00] Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using diffie-hellman. In *EUROCRYPT*, pages 156–171, 2000.

[BPR00] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT*, pages 139–155, 2000.

[BR93] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, 1989.

[DF90] Yvo Desmedt and Yair Frankel. Threshold Cryptosystems. In *CRYPTO '89*, volume 435 of *LNCS*, pages 307–315, 1990.

[DFK$^+$03] Yevgeniy Dodis, Matthew K. Franklin, Jonathan Katz, Atsuko Miyaji, and Moti Yung. Intrusion-resilient public-key encryption. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 19–32. Springer, 2003.

[DFK$^+$04] Yevgeniy Dodis, Matthew K. Franklin, Jonathan Katz, Atsuko Miyaji, and Moti Yung. A generic construction for intrusion-resilient public-key encryption. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 81–98. Springer, 2004.

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 65–82. Springer, 2002.

[DKXY03] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Strong key-insulated signature schemes. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 130–144. Springer, 2003.

[Fis05] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Crypto'05*, 2005.

[FJ00] Warwick Ford and Burton S. Kaliski Jr. Server-assisted generation of a strong secret from a password. In *WETICE*, pages 176–180, 2000.

[FK90] David C. Feldmeier and Philip R. Karn. Unix password security - ten years later. In *CRYPTO '89: Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, pages 44–63. Springer-Verlag, 1990.

[Jab01] D. Jablon. Password authentication using multiple servers. In *CT-RSA'01: RSA Cryptographers' Track*, pages 344–360. Springer-Verlag, 2001.

[Kle90] D. Klein. Foiling the cracker: A survey of, and improvements to, password security. In *The 2nd USENIX Security Workshop*, pages 5–14, 1990.

[KMTG05] Jonathan Katz, Philip Mackenzie, Gelareh Taban, and Virgil Gligor. Two-server password-only authenticated key exchange. In *Proc. Applied Cryptography and Network Security ACNS05*, 2005.

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti 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.

[Lab09] Mozilla Labs. Weave sync, 2009. Available at http://labs.mozilla.com/projects/weave.

[Las09] LastPass. Lastpass password manager, 2009. Available at https://lastpass.com.

[MR03] Philip D. MacKenzie and Michael K. Reiter. Networked cryptographic devices resilient to capture. *Int. J. Inf. Sec.*, 2(1):1–20, 2003.

[MSJ02] Philip D. MacKenzie, Thomas Shrimpton, and Markus Jakobsson. Threshold password-authenticated key exchange. In *Advances in Cryptology - CRYPTO 2002, International Cryptology Conference*, 2002.

[MT79] Robert Morris and Ken Thompson. Password security: a case history. *Commun. ACM*, 22(11):594–597, 1979.

[RG03] Mario Di Raimondo and Rosario Gennaro. Provably secure threshold password-authenticated key exchange. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Application of Cryptographic Techniques*, 2003.

[Sch90] C.P. Schnorr. Efficient identification and signatures for smart cards. In *Crypto '89*, pages 239–252, 1990.

[Sha79] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[Sho00] Victor Shoup. Practical Threshold Signatures. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 207–220, 2000.

[Wu99] Thomas D. Wu. A real-world analysis of kerberos password security. In *Proceedings of the Network and Distributed System Security Symposium, NDSS*, 1999.

[XS03] Shouhuai Xu and Ravi S. Sandhu. Two efficient and provably secure schemes for server-assisted threshold signatures. In *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference*, 2003.

Appendix A Simulation-Sound Zero-Knowledge Labeled Proof Systems

```
Experiment $\mathbf{Exp}^{\mathsf{ZK}}_{\mathsf{NILP}}(\mathcal{A}, \Pi^{\mathcal{H}}[\mathcal{L}])$
  $b \xleftarrow{r} \{0,1\}$, $\mathsf{HTable} \leftarrow \emptyset$.
  Run $\mathcal{A}$, and handle $\mathcal{A}$'s queries as follows:
  On query $(x, w, \tau)$ to the prover:
    If $b = 0$ then $\pi \leftarrow \mathcal{P}^{\mathcal{H}}[\mathcal{L}](x, w, \tau)$,
    o/w pass $(x, \tau)$ to $\mathcal{S}[\mathcal{L}]$ to obtain $(\pi, (\sigma, \rho))$,
        add $(\sigma, \rho)$ to $\mathsf{HTable}$, return $\pi$ to $\mathcal{A}$.
  On query $\sigma$ to hash function $\mathcal{H}$:
    If $b = 1$ and $(\sigma, \rho) \in \mathsf{HTable}$ for some $\rho$
        then return $\rho$, o/w return $\mathcal{H}(\sigma)$
  When $\mathcal{A}$ halts, parse its output as $b'$.
  If $b = b'$ then output 1, o/w output 0.
```

```
Experiment $\mathbf{Exp}^{\mathsf{SS}}_{\mathsf{NILP}}(\mathcal{A}, \Pi^{\mathcal{H}}[\mathcal{L}])$
  $\mathsf{SList} \leftarrow \emptyset$, $\mathsf{HTable} \leftarrow \emptyset$.
  Run $\mathcal{A}$, and handle $\mathcal{A}$'s queries as follows:
  On query $(x, \tau)$ to the prover:
    Add $(x, \tau)$ to $\mathsf{SList}$, pass $(x, \tau)$ to $\mathcal{S}[\mathcal{L}]$ to
    obtain $(\pi, (\sigma, \rho))$, add $(\sigma, \rho)$ to $\mathsf{HTable}$, return $\pi$ to $\mathcal{A}$.
  On query $\sigma$ to hash function $\mathcal{H}$:
    If $(\sigma, \rho) \in \mathsf{HTable}$ for some $\rho$
        then return $\rho$, o/w return $\mathcal{H}(\sigma)$
  When $\mathcal{A}$ halts, parse its output as $(x, \pi, \tau)$,
  If $(\mathcal{V}[\mathcal{L}](x, \pi, \tau) = 1) \wedge (x \notin \mathcal{L}) \wedge ((x, \tau) \notin \mathsf{SList})$
        then output 1, o/w output 0.
```

Fig. 4. Experiments for Simulation-Sound Zero-Knowledge for Non-Interactive Labeled Proof Systems

Non-Interactive Labeled Proofs: A non-interactive *labeled* proof (NILP) system for NP language $\mathcal{L}$ is a standard non-interactive proof system where all algorithms take as an additional input a bitstring called 'label'. The point of introducing such label is to efficiently facilitate non-malleable binding of a proof to some protocol-specific value. The non-malleability property of such labeled non-interactive proof is that an adversary cannot substitute a label used by an honest prover with his own label. This is technically expressed by the notion of simulation-soundness defined below as inability of an adversary, on oracle access to a simulator which creates a simulated "proof" for (statement,label) pair queried by the adversary, to create a valid proof for any wrong statement, i.e. a statement outside of $\mathcal{L}$, with any label, except for one of the (statement,label) pair for which the adversary received a simulated "proof" from the simulator. In our PPSS protocol we use labeled proofs to bind user's proof of correct formation of its message to its public key, so that a man in the middle attacker cannot replace the honest user's public key while re-using its proof of correctness.

Formally, a NILP $\Pi[\mathcal{L}]$ is a tuple of probabilistic polynomial time algorithms $(\mathcal{P}[\mathcal{L}], \mathcal{V}[\mathcal{L}])$ where the prover algorithm, $\mathcal{P}[\mathcal{L}]$, on input a statement x, a witness for that statement w, and a label τ outputs a proof π; and the verifier algorithm, $\mathcal{V}[\mathcal{L}]$, on input a statement x, a proof π, and a label τ decides whether or not π is a correct proof of membership of x in $\mathcal{L}$ given label τ. These two algorithms must satisfy the usual

completeness and *soundness* properties of a proof system:
completeness: For any label $\tau \in \{0,1\}^*$ and any statement $x \in \mathcal{L}$, there exists a witness w st if π is output by $\mathcal{P}[\mathcal{L}](x, w, \tau)$ then the verification passes $i.e.$ $\mathcal{V}[\mathcal{L}](x, \pi, \tau)$ outputs 1.
(T, ϵ)-*soundness:* For any label $\tau \in \{0,1\}^*$, any statement $x \notin \mathcal{L}$, and any adversarial algorithms $\mathcal{A}$ limited by time T, $\Pr[1 \leftarrow \mathcal{V}[\mathcal{L}](x, \pi, \tau) \,|\, \pi \leftarrow \mathcal{A}(\mathcal{L}, \tau, x)] \leq \epsilon$.

Simulation-Sound Zero-Knowledge: The property that makes a labeled proof system most useful in a multi-party protocol like PPSS is *simulation-sound zero-knowledge*. Since our NILP constructions work in the Random Oracle Model (ROM) [BR93], we define both properties in ROM, i.e. the prover, the verifier, and the adversarial algorithms have oracle access to a hash function $\mathcal{H}$ modeled as a Random Oracle. Simulation-sound zero-knowledge of NILP in ROM is defined by two experiments in Figure 4 above. In the first experiment the adversary is faced with access to either the real prover and a real hash function or to the simulator, and his goal is to distinguish between the two. In the second experiment the adversary has access to the simulator and succeeds if he makes a proof for any invalid statement different s.t. either the statement or its label are different from the (statement,label) pairs for which the simulator supplied the adversary a (simulated) proof. One can note that Figure 4 restricts the way the simulator manipulates the hash function $\mathcal{H}$: Namely, the simulator is only allowed to rewrite one (input,output) pair in $\mathcal{H}$ per each query to the prover. We choose to express the SSZK notion in this way because it makes it simpler to state and because the simulators for the non-interactive zero-knowledge proofs using the Fiat-Shamir heuristic adhere to this restriction.

Formally, we define a non-interactive labeled proof (NILP) system $\Pi^{\mathcal{H}}[\mathcal{L}] = (\mathcal{P}^{\mathcal{H}}[\mathcal{L}], \mathcal{V}^{\mathcal{H}}[\mathcal{L}])$ to be $(T_S, q_P, \epsilon_{ZK}, \epsilon_{SS})$-simulation-sound zero-knowledge if there exists a simulator $\mathcal{S}[\mathcal{L}]$ with running time bounded by T_S $s.t.$ for any adversarial algorithm $\mathcal{A}$ that makes at most q_P queries to prover $\mathcal{P}[\mathcal{L}]$,

$$\Pr[1 \leftarrow \mathbf{Exp}^{\mathsf{ZK}}_{\mathsf{NILP}}(\mathcal{A}, \Pi^{\mathcal{H}}[\mathcal{L}])] \leq 1/2 + \epsilon_{ZK}$$
$$\Pr[1 \leftarrow \mathbf{Exp}^{\mathsf{SS}}_{\mathsf{NILP}}(\mathcal{A}, \Pi^{\mathcal{H}}[\mathcal{L}])] \leq \epsilon_{SS}$$

where $\mathbf{Exp}^{\mathsf{ZK}}_{\mathsf{NILP}}$ and $\mathbf{Exp}^{\mathsf{SS}}_{\mathsf{NILP}}$ are as in Figure 4 and the probabilities go over randomness of $\mathcal{A}$, of the experiments $\mathbf{Exp}^{\mathsf{ZK}}_{\mathsf{NILP}}$ and $\mathbf{Exp}^{\mathsf{SS}}_{\mathsf{NILP}}$, and of the Random Oracle $\mathcal{H}$.

```
$\mathcal{P}^{\mathcal{H}}[\mathcal{L}_U^{\mathsf{st}_0}](x_U, w_U, \tau)$
    Parse $\mathsf{st}_0$ as $(g, y, h, h_0, h_1, h_2, \bar{y}_1, c_p, c_s)$
    Parse $x_U$ as $(\{a_j, e_j\}_{j=1}^{t+1}, c_{\tilde{p}}, o_1, o_2)$
    Parse $w_U$ as $(r_{\tilde{p}}, \tilde{p})$
    Pick $\sigma_1, \sigma_2 \xleftarrow{r} \mathbb{Z}_q$
    $\forall j \in [t+1] : A_j \leftarrow (a_j)^{\sigma_1}$
    $\{B_i\}_{i=1}^4 \leftarrow (y^{\sigma_1} h^{\sigma_2}, g^{\sigma_1}, \bar{y}_1^{\sigma_1} h_1^{\sigma_2}, h_2^{\sigma_1})$
    $c \leftarrow \mathcal{H}(x, \{A_j\}_{j=1}^{t+1}, \{B_i\}_{i=1}^4, \tau)$
    $(\zeta_1, \zeta_2) \leftarrow (\sigma_1 + c \cdot r_{\tilde{p}}\ , \sigma_2 + c \cdot \tilde{p})$
    output $\pi = (c, \zeta_1, \zeta_2)$

$\mathcal{V}^{\mathcal{H}}[\mathcal{L}_U^{\mathsf{st}_0}](x_U, \pi, \tau)$
    Parse $\mathsf{st}_0, x_U, \pi$ as in the prover algorithm above
    $\forall j \in [t+1] : A_j \leftarrow (a_j)^{\zeta_1} (e_j)^{-c}$
    $B_1 \leftarrow y^{\zeta_1} h^{\zeta_2} (c_{\tilde{p},L})^{-c}$ , $B_2 \leftarrow g^{\zeta_1} (c_{\tilde{p},R})^{-c}$
    $B_3 \leftarrow (\bar{y}_1)^{\zeta_1} h_1^{\zeta_2} o_1^{-c}$ , $B_4 \leftarrow h_2^{\zeta_1} o_2^{-c}$
    If $(c = \mathcal{H}(x, \{A_j\}_{j=1}^{t+1}, \{B_i\}_{i=1}^4, \tau))$ output 1
    o/w output 0

$\mathcal{P}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}](x_S, w_S, \tau)$
    Parse $\mathsf{st}_0$ as $(g, y, h, h_0, h_1, h_2, \bar{y}_1, c_p, c_s)$
    Parse $x_S$ as $(a, b, w)$
    Parse $w_S$ as $\alpha$
    $\sigma \xleftarrow{r} \mathbb{Z}_q$
    $A \leftarrow g^{\sigma}$ , $B \leftarrow (c_{\tilde{p},L})^{\sigma}$ , $C \leftarrow h_0^{\sigma}$
    $c \leftarrow \mathcal{H}(x, A, B, C, \tau)$
    $\zeta \leftarrow \sigma + c \cdot \alpha$
    output $\pi = (c, \zeta)$

$\mathcal{V}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}](x_S, \pi, \tau)$
    Parse $\mathsf{st}_0, x, \pi$ as in the prover algorithm above
    $A \leftarrow g^{\zeta} a^{-c}$ , $B \leftarrow (c_{\tilde{p},L})^{\zeta} b^{-c}$ , $C \leftarrow h_0^{\zeta} w^{-c}$
    If $(c = \mathcal{H}(x, A, B, C, \tau))$ output 1
    o/w output 0
```

Fig. 5. The Prover and Verifier Algorithms for SSZK NILP Systems for $\mathcal{L}_U^{\mathsf{st}_0}$ and $\mathcal{L}_S^{\mathsf{st}_0}$

ROM-based Instantiations of SSZK Proofs. The details of proof systems $\Pi^{\mathcal{H}}[\mathcal{L}_U^{\mathsf{sto}}]$ and $\Pi^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{sto}}]$ are shown in Figure 5. Since both proofs are straightforward generalizations of Schnorr's proof of knowledge of discrete logarithm [Sch90], the well-known simulation for such proof in ROM shows that for any q_P both of the above proof systems achieve $(T_S, q_P, q_H, \epsilon_{ZK}, \epsilon_{SS})$ simulation-sound zero-knowledge given an adversary limited to at most q_H queries to $\mathcal{H}$, as long as $\epsilon_{ZK} = q_P q_H / q$, $\epsilon_{SS} = q_H/q$, and T_S is bounded by $(t+5)q_P t_{exp}$ in the case of $\mathcal{L}_U^{\mathsf{sto}}$ and $3q_P t_{exp}$ in the case of $\mathcal{L}_S^{\mathsf{sto}}$.

Appendix B Details of the Security Proof for the PPSS Protocol

Claim 5 *Games G_0 and G_1 are indistinguishable under DDH assumption. Concretely, for any secret s, if $q_U \le q_P^U$, and $T_{\mathcal{A}} \le T_{ddh} - T_U - [7+(11n+9)q_S + (5n+12)q_U]t_{exp}$ then $|p_0(s) - p_1(s)| \le q_U \epsilon_{ddh} + \epsilon_{ZK}^U$.*

Proof. Consider game $\bar{\mathsf{G}}_0$ in which the proofs output by $\mathcal{P}^{\mathcal{H}}[\mathcal{L}_U^{\mathsf{sto}}]$ in the $\mathsf{User}^{\diamond}$ oracle is replaced by a simulated proof output by $\mathcal{S}[\mathcal{L}_U^{\mathsf{sto}}]$. Since $\Pi^{\mathcal{H}}[\mathcal{L}_U^{\mathsf{sto}}]$ is $(T_U, q_P^U, q_H^U, \epsilon_{ZK}^U, \epsilon_{SS}^U)$-SSZK, as long as the number of $\mathsf{User}^{\diamond}$ oracle queries are less than q_P^U, any adversarial algorithm has at most ϵ_{ZK}^U chance of distinguishing between G_0 and $\bar{\mathsf{G}}_0$. To show that $\bar{\mathsf{G}}_0$ and G_1 are indistinguishable, we make a hybrid argument over q_U user sessions. We define a series of intermediary games G_0^i between $\bar{\mathsf{G}}_0$ and G_1 where G_0^i in the $\mathsf{User}^{\diamond}$ oracle calls, follows G_1 on the first i user sessions, *i.e.* it picks $o_1 \xleftarrow{r} G$, and then follows $\bar{\mathsf{G}}_0$ on the remaining sessions. Clearly, $\mathsf{G}_0^0 \equiv \bar{\mathsf{G}}_0$ and $\mathsf{G}_0^{q_U} \equiv \mathsf{G}_1$. Let $p_0^i = \Pr[1 \leftarrow (\mathcal{A} \rightleftharpoons \mathsf{G}_0^i)]$. For each $i > 0$ we construct reduction $\mathcal{R}_{0,1}^i$ which on input a DDH challenge (A, B, C) follows $\bar{G}_0$ during $\mathsf{Init}^{\diamond}$ except that it chooses $\bar{y}_1 \leftarrow B$, $h_0 \leftarrow A^{r_0}$, $h_2 \leftarrow g^{r_2}$ for random $(r_0, r_2) \xleftarrow{r} (\mathbb{Z}_q)^2$, then $\mathcal{R}_{0,1}^i$ follows $\bar{G}_0$ in all $\mathsf{Server}^{\diamond}$ calls, but for User sessions it follows G_1 on all sessions prior to the i-th session, on the i-th session it sets $\{e_j \leftarrow w_j^{1/r_0}\}_{j=1}^t o_1 \leftarrow C h_1^{\tilde{p}}, h_2^{r_{\tilde{p}}} \leftarrow A^{r_2}, c_{\tilde{p},L} \leftarrow A$ and simulates π_2, and on remaining sessions it follows $\bar{\mathsf{G}}_0$. Let $A = g^{\alpha}$, $B = g^{\beta}$ and $C = g^{\gamma}$. Note that if (A, B, C) is a DDH tuple then $\mathcal{R}_{0,1}^i(A, B, C) \equiv \mathsf{G}_0^{i-1}$ because if $\gamma = \alpha\beta$ then for the i-th session we have $r_{\tilde{p}} = \alpha, o_1 = \bar{y}_1^{\alpha} h_1^{\tilde{p}}$. However if (A, B, C) is a random tuple then o_1 is independently random from $e_j, \bar{y}_1, o_2, c_{\tilde{p},L}$, and so $\mathcal{R}_{0,1}^i(A, B, C) \equiv \mathsf{G}_0^i$. Therefore for every i it holds that if $T_{\mathcal{R}_{0,1}^i} + T_{\mathcal{A}} < T_{ddh}$ then $|p_0^i - p_0^{i+1}| \le \epsilon_{ddh}$. Since the time $T_{\mathcal{R}_{0,1}^i}$ of algorithm $\mathcal{R}_{0,1}^i$ is bounded by $T_U + [7 + (11n+9)q_S + (5n+12)q_U]t_{exp}$, the claim follows.

Claim 6 *Games G_1 and G_2 are indistinguishable assuming the $\Sigma = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ encryption scheme is IND-CPA. Concretely, for any secret s, if $T_{\mathcal{A}} \le T_{cpa} - T_U - [6 + (11n+9)q_S + (5n+12)q_U]t_{exp}$ then $|p_1(s) - p_2(s)| \le n q_S \epsilon_{cpa}$.*

Proof. This is a hybrid argument over nq_S server sessions. For each $i \in [0, nq_S]$ we define an intermediate game G_1^i which follows G_2 on calls to $\mathsf{Server}_j^{\diamond}$ for $j \le i$ and follows G_1 on calls to $\mathsf{Server}_j^{\diamond}$ for $j > i$. Note $\mathsf{G}_1^0 \equiv \mathsf{G}_1$ and $\mathsf{G}_1^{nq_S} \equiv \mathsf{G}_2$. Let $p_1^i = \Pr[1 \leftarrow (\mathcal{A} \rightleftharpoons \mathsf{G}_1^i)]$. For each $i \ge 0$ we show a reduction $R_{1,2}^i$ which reduces breaking CPA encryption to distinguishing between G_1^i and G_1^{i+1}. Given random pk of encryption scheme Σ, $R_{1,2}^i(pk)$ picks $i' \xleftarrow{r} [1, q_U]$, follows $\mathsf{Init}^{\diamond}$ as G_1, and follows all $\mathsf{User}^{\diamond}$ queries as G_1 except that it embeds its challenge key pk as the key chosen by i'-th user session. $\mathcal{R}_{1,2}^i$ also follows $\mathsf{G}_{1,2}^i$ with regards to all $\mathsf{Server}_j^{\diamond}$ queries except for $\mathsf{Server}_{i+1}^{\diamond}$: If $\mathcal{A}$ passes pk' to $\mathsf{Server}_{i+1}^{\diamond}$ s.t. $pk' \ne pk$ then $\mathcal{R}_{1,2}^i$ outputs a random bit and stops, but if $pk' = pk$ then $\mathcal{R}_{1,2}^i$ sends $(m_0, m_1) = (1, z_j)$ to the encryption challenger, and passes $\mathcal{E}_{pk}(m_b)$ computed by the challenger for a random b back to $\mathcal{A}$ as ζ_j. When $\mathcal{A}$ outputs its final output v, $\mathcal{R}_{1,2}^i$ outputs the bit output by $\mathsf{D}(v)$. It is easy to see that $\mathcal{R}_{1,2}^i$'s advantage in CPA security game is at least $1/q_U$, the probability that it guesses the correct user's key, times $|p_1^i - p_1^{i+1}|$. Therefore for each i it holds that if $T_{\mathcal{R}_{1,2}^i} + T_{\mathcal{A}} + T_{\mathsf{D}} < T_{cpa}$ then $|p_1^i - p_1^{i+1}| \le q_U \epsilon_{cpa}$. Since the time $T_{\mathcal{R}_{1,2}^i}$ of $\mathcal{R}_{1,2}^i$ is bounded by $T_U + [6 + (11n+9)q_S + (5n+12)q_U]t_{exp}$ exponentiations, the claim follows.

Claim 7 *Games G_2 and G_3 are indistinguishable under DDH assumption. Concretely, for any secret s, if $q_S \leq q_P^S/n$, $q_U \leq q_P^U$, and $T_{\mathcal{A}} \leq T_{ddh} - T_U - T_S - [6 + (14n+12)q_S + (5n+12)q_U]t_{exp}$ then $|p_2(s) - p_3(s)| \leq \epsilon_{ddh} + 2\epsilon_{ZK}^S + q_S\epsilon_{SS}^U$.*

Proof. Consider games $\bar{\mathsf{G}}_2$ and $\bar{\mathsf{G}}_3$ in which the proofs output by $\mathcal{P}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}]$ in the $\mathsf{Init}^\diamond$ oracle is replaced by a simulated proof output by $\mathcal{S}[\mathcal{L}_S^{\mathsf{st}_0}]$. Since $\Pi^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}]$ is $(T_S, q_P^S, q_H^S, \epsilon_{ZK}^S, \epsilon_{SS}^S)$-SSZK, as long as the number of $\mathsf{Server}^\diamond$ oracle queries are less than q_P^S/n, any adversarial algorithm has at most ϵ_{ZK}^S chance of distinguishing between G_2 and $\bar{\mathsf{G}}_2$ and similarly between G_3 and $\bar{\mathsf{G}}_3$.

We use a hybrid argument over nq_S server sessions to argue that $\bar{\mathsf{G}}_2$ and $\bar{\mathsf{G}}_3$ are indistinguishable. For each $i \in [0, nq_S]$, we define an intermediate game $\bar{\mathsf{G}}_2^i$ which follows $\bar{\mathsf{G}}_3$ on calls to $\mathsf{Server}_j^\diamond$ for $j \leq i$ and follows $\bar{\mathsf{G}}_2$ on calls to $\mathsf{Server}_j^\diamond$ for $j > i$. Note $\bar{\mathsf{G}}_2^0 \equiv \bar{\mathsf{G}}_2$ and $\bar{\mathsf{G}}_2^{nq_S} \equiv \bar{\mathsf{G}}_3$. Let $p_2^i = \Pr[1 \leftarrow (\mathcal{A} \rightleftharpoons \bar{\mathsf{G}}_2^i)]$. For each $i \geq 0$ we show a reduction $R_{2,3}^i$ which reduces breaking DDH assumption to distinguishing between $\bar{\mathsf{G}}_2^{i-1}$ and $\bar{\mathsf{G}}_2^i$. Let DDH challenge (A, B, C) where $A = g^\alpha$, $B = g^\beta$ and $C = g^\gamma$ be an input to $\mathcal{R}_{2,3}^i$. The reduction algorithm sets $h = B$, $a_i = A$, $b_i = A^{r_p}$. In other words, $\mathcal{R}_{2,3}^i$ assumes that $t_i = \alpha$. If (A, B, C) is a DDH tuple ($i.e.$ $\gamma = \alpha\beta$), then $c_{\beta,R,i} = (b_i/e_i)^x C^p (o_1/c_{\tilde{p},R}^{\bar{x}_1})^{-1/r_1} = y^{(r_p - r_{\tilde{p}})t_i} h^{(p-\tilde{p})t_i} = (c_{p,R}/c_{\tilde{p},R})^{t_i}$. And hence $\mathcal{R}_{2,3}^i$ is equivalent to $\bar{\mathsf{G}}_2^{i-1}$ except when the adversary manages to make correct proof on wrong statement (o_1, o_2, e_i) which can happen with probability at most equal to ϵ_{SS}^U as long as the number of $\mathsf{User}^\diamond$ oracle queries are less than q_P^U. If (A, B, C) is a random tuple, $i.e.$ (α, β, γ) are distributed uniformly and independently at random, then $c_{\beta,R,i}$ can be replaced with a random variable which is independent from a_i, b_i when $p \neq \tilde{p}$. Therefore $\mathcal{R}_{2,3}^i$ is equivalent to $\bar{\mathsf{G}}_2^i$. So for each i, it holds that if $T_{\mathcal{R}_{2,3}^i} + T_{\mathcal{A}} < T_{ddh}$, then $|p_2^i - p_2^{i+1}| \leq \epsilon_{ddh}$. Since $T_{\mathcal{R}_{2,3}^i}$ is bounded by $T_U + T_S + [6 + (14n+12)q_S + (5n+12)q_U]t_{exp}$, the claim follows.

Claim 8 *For any secret s, $p_3^{\neg\mathsf{F}}(s) = p_4^{\neg\mathsf{F}}(s)$.*

Proof. We argue that under condition that event F does not happen the adversarial views in games G_3 and G_4 are identical. This immediately implies that (1) $\Pr[\mathsf{F}_{3,s}] = \Pr[\mathsf{F}_{4,s}]$, and (2) that the conditional probabilities $\Pr[1 \leftarrow (\mathcal{A} \rightleftharpoons \mathsf{G}_3(s)) \mid \neg\mathsf{F}_{3,s}]$ and $\Pr[1 \leftarrow (\mathcal{A} \rightleftharpoons \mathsf{G}_4(s)) \mid \neg\mathsf{F}_{4,s}]$ are the same, and hence the claim follows, because $p_i^{\neg\mathsf{F}}(s) = \Pr[1 \leftarrow (\mathcal{A} \rightleftharpoons \mathsf{G}_i(s)) \mid \neg\mathsf{F}_{i,s}] * \Pr[\mathsf{F}_{i,s}]$ To argue this, note that the only difference between G_3 and G_4 is that in G_3, x_i's are (n, t)-secret-sharing of a random value whereas in G_4 x_i's are (n, t)-secret-sharing of zero. Therefore unless adversary knows $t + 1$ shares of x_i's, the view of the adversary in G_3 is identical to its view in G_4. Now, adversary gets to know $t' < t$ shares of x simply by corrupting t' servers. However server queries could possibly leak information about x_i's as in $d_{\alpha,L,j}$ value. But in any server query, if adversary does not use the legitimate password, $d_{\alpha,L,j}$ is masked with a random value in both G_3 and G_4; otherwise if adversary uses the legitimate password, unless event F happens, the number of distinct servers which adversary contacts is bounded by $t - t'$. Thus, unless event F happens, the maximum number of x_i shares that is effectively used in either game is bounded by t.

Claim 9 *Games G_4 and G_5 are indistinguishable under DDH assumption. Concretely, for any secret s, if $T_{\mathcal{A}} \leq T_{ddh} - T_U - [3 + (14n+12)q_S + (5n+12)q_U]t_{exp}$ then $|p_4^{\neg\mathsf{F}}(s) - p_5^{\neg\mathsf{F}}(s)| \leq \epsilon_{ddh}$ and $|\Pr[\mathsf{F}_{4,s}] - \Pr[\mathsf{F}_{5,s}]| \leq \epsilon_{ddh}$.*

Proof. Given DDH challenge (A, B, C), the reduction $\mathcal{R}_{4,5}$, embeds $y \leftarrow A$, $c_{s,L} \leftarrow B$ and $c_{s,R} \leftarrow C \cdot s$. If (A, B, C) is a DDH tuple, then $c_s = (c_{s,L}, c_{s,R})$ is a valid ElGamal encryption of s and hence $\mathcal{R}_{4,5}(A, B, C) \equiv \mathsf{G}_4$. However if (A, B, C) is a random tuple, then $c_s = (c_{s,L}, c_{s,R})$ is distributed uniformly at random in G^2 and hence $\mathcal{R}_{4,5}(A, B, C) \equiv \mathsf{G}_5$. Since the event F can be tested by the reduction $\mathcal{R}_{4,5}$ and the reduction can also observe if $\mathcal{A}$ outputs 1, it follows that the probability difference between an event that ($\mathcal{A}$ outputs 1 and $\neg\mathsf{F}$) is at most ϵ_{ddh} in these two games. By the same token, the probability difference between just the event F in these two games is also bounded by ϵ_{ddh}.

Claim 10 *Games* G_5 *and* G_6 *are indistinguishable under DDH assumption. Concretely for any secret* s*, if* $q_S \leq q_P^S/n$*,* $q_U \leq q_P^U$*, and* $T_{\mathcal{A}} \leq T_{ddh} - T_U - T_S - [3 + (14n + 17)q_S + (5n + 12)q_U]t_{exp}$*, then* $|p_5(s) - p_6(s)| \leq nq_S\epsilon_{ddh} + 2\epsilon_{ZK}^S + q_S\epsilon_{SS}^U$ *and* $|\Pr[\mathsf{F}_{5,s}] - \Pr[\mathsf{F}_{6,s}]| \leq nq_S\epsilon_{ddh} + 2\epsilon_{ZK}^S + q_S\epsilon_{SS}^U$.

Proof. Consider games $\bar{\mathsf{G}}_5$ and $\bar{\mathsf{G}}_6$ in which the proofs output by $\mathcal{P}^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}]$ in the $\mathsf{Init}^\diamond$ oracle are replaced by simulated proofs output by $\mathcal{S}[\mathcal{L}_S^{\mathsf{st}_0}]$. Since $\Pi^{\mathcal{H}}[\mathcal{L}_S^{\mathsf{st}_0}]$ is $(T_S, q_P^S, q_H^S, \epsilon_{ZK}^S, \epsilon_{SS}^S)$-SSZK, as long as the number of $\mathsf{Server}^\diamond$ oracle queries are less than q_P^S/n, any adversarial algorithm has at most ϵ_{ZK}^S chance of distinguishing between G_5 and $\bar{\mathsf{G}}_5$ and similarly between G_6 and $\bar{\mathsf{G}}_6$.

We use a hybrid argument over nq_S server sessions to argue that $\bar{\mathsf{G}}_5$ and $\bar{\mathsf{G}}_6$ are indistinguishable. For each $i \in [0, nq_S]$, we define an intermediate game $\bar{\mathsf{G}}_5^i$ which follows $\bar{\mathsf{G}}_6$ on calls to $\mathsf{Server}_j^\diamond$ for $j \leq i$ and follows $\bar{\mathsf{G}}_5$ on calls to $\mathsf{Server}_j^\diamond$ for $j > i$. Note $\bar{\mathsf{G}}_5^0 \equiv \bar{\mathsf{G}}_5$ and $\bar{\mathsf{G}}_5^{nq_S} \equiv \bar{\mathsf{G}}_6$. Let $p_5^i = \Pr[1 \leftarrow (\mathcal{A} \rightleftharpoons \bar{\mathsf{G}}_5^i)]$. For each $i \geq 0$ we show a reduction $R_{5,6}^i$, which reduces breaking DDH assumption to distinguishing between $\bar{\mathsf{G}}_5^{i-1}$ and $\bar{\mathsf{G}}_5^i$. Let DDH challenge (A, B, C) where $A = g^\alpha$, $B = g^\beta$ and $C = g^\gamma$ be an input to $\mathcal{R}_{5,6}^i$. The reduction algorithm sets $y = A$, $a_j = B, y^{t_i} = C$. In other words, $\mathcal{R}_{5,6}^i$ assumes that $x = \alpha$ and $t_i = \beta$. If (A, B, C) is a DDH tuple ($i.e.$ $\gamma = \alpha\beta$), then $c_{\beta,R,i} = C^{r_p}(o_2)^{-1/r_2}[B^p(o_1/(c_{\tilde{p},L})^{\bar{x}_1})^{-1/r_1}]^{\bar{x}} = C^{r_p - r_{\tilde{p}}}B^{(p-\tilde{p})\bar{x}} = y^{(r_p - r_{\tilde{p}})t_i}h^{(p-\tilde{p})t_i} = (c_{p,R}/c_{\tilde{p},R})^{t_i}$. And hence $\mathcal{R}_{5,6}^i$ is equivalent to $\bar{\mathsf{G}}_5^{i-1}$ except when the adversary manages to make correct proof on wrong statement (o_1, o_2) which can happen with probability at most equal to ϵ_{SS}^U as long as the number of $\mathsf{User}^\diamond$ oracle queries are less than q_P^U. If (A, B, C) is a random tuple, $i.e.$ (α, β, γ) are distributed uniformly and independently at random, then $c_{\beta,R,i}$ can be replaced with a random variable which is independent from a_i, b_i, e_i and w_i when $r_p \neq r_{\tilde{p}}$. Therefore $\mathcal{R}_{5,6}^i$ is equivalent to $\bar{\mathsf{G}}_5^i$. So for each i, it holds that if $T_{\mathcal{R}_{5,6}^i} + T_{\mathcal{A}} < T_{ddh}$, then $|p_0^i - p_0^{i+1}| \leq \epsilon_{ddh}$. Since $T_{\mathcal{R}_{5,6}^i}$ is bounded by $T_U + T_S + [3 + (14n + 17)q_S + (5n + 12)q_U]t_{exp}$, the claim follows.

Claim 11 *Games* G_6 *and* G_7 *are indistinguishable under DDH assumption. Concretely, for any secret* s*, if* $T_{\mathcal{A}} \leq T_{ddh} - T_U - [5 + (14n + 12)q_S + (5n + 20)q_U)t_{exp}$ *then* $|\Pr[\mathsf{F}_{6,s}] - \Pr[\mathsf{F}_{7,s}]| \leq \epsilon_{ddh} + \epsilon_{UU}^S$.

Proof. We make a hybrid argument over q_U user sessions. We define a series of intermediary games G_6^i between G_6 and G_7 where G_6^i in the $\mathsf{User}^\diamond$ oracle calls, follows G_7 on the first i user sessions, i.e. it picks $c_{\tilde{p},R} \xleftarrow{r} G$ and hence there is no information about p revealed in these sessions. On the remaining sessions, G_6^i follows G_6. Clearly, $\mathsf{G}_6^0 \equiv \mathsf{G}_6$ and $\mathsf{G}_6^{q_U} \equiv \mathsf{G}_7$. Let $p_6^i = \Pr[1 \leftarrow (\mathcal{A} \rightleftharpoons \mathsf{G}_6^i)]$. For each $i > 0$ we construct reduction $\mathcal{R}_{6,7}^i$ which on input a DDH challenge (A, B, C) follows G_6 during $\mathsf{Init}^\diamond$ except that it chooses $y \leftarrow A$, $h_0 \leftarrow B^{r_0}$, $h_2 \leftarrow g^{r_2}$ for random $r_0, r_2 \xleftarrow{r} \mathbb{Z}_q$, then $\mathcal{R}_{6,7}^i$ follows G_6 in all $\mathsf{Server}^\diamond$ calls, but for User sessions it follows G_7 on all sessions prior to the i-th session, on the i-th session it sets $\{e_j \leftarrow w_j^{-1/r_0}\}_{j=1}^t$, $c_{\tilde{p},L} \leftarrow B$, $c_{\tilde{p},R} \leftarrow Ch^p$, $o_2 \leftarrow B^{r_2}$ and simulates Π_2; on remaining sessions $\mathcal{R}_{6,7}^i$ follows G_6. Let $A = g^\alpha$, $B = g^\beta$ and $C = g^\gamma$. In other words, we assume $r_{\tilde{p}} = \beta$. Note that if (A, B, C) is a DDH tuple then $\mathcal{R}_{6,7}^i(A, B, C) \equiv \mathsf{G}_6^{i-1}$ because if $\gamma = \alpha\beta$ then for the ith session we have $y = g^\alpha$, $e_j = h_0^{t_j/r_0} = g^{\beta t_j}$ (for $j = 1$ to t), $o_2 = g^{\beta r_2} = h_2^\beta$, $c_{\tilde{p},R} = g^{\alpha\beta}h^p$ which is distributed as in G_6^{i-1} except when the adversary manages to make correct proof on wrong statement w_j' which can happen with probability at most equal to ϵ_{UU}^S as long as the number of $\mathsf{Server}^\diamond$ oracle queries are less than q_P^S. However if (A, B, C) is a random tuple then $c_{\tilde{p},R}$ is distributed uniformly at random in G independent of all $\{e_j\}_{j=1}^t$ and $c_{\tilde{p},L}$ and thus $\mathcal{R}_{6,7}^i(A, B, C) \equiv \mathsf{G}_6^i$. Therefore for every i it holds that if $T_{\mathcal{R}_{6,7}^i} + T_{\mathcal{A}} < T_{ddh}$ then $|p_6^i - p_6^{i+1}| \leq \epsilon_{ddh}$. Since the time $T_{\mathcal{R}_{6,7}^i}$ of algorithm $\mathcal{R}_{6,7}^i$ is bounded by $T_U + [5 + (14n + 12)q_S + (5n + 20)q_U)t_{exp}$ exponentiations, and reduction $\mathcal{R}_{6,7}$ can test whether event F happens, the claim follows.

Claim 12 *Games* G_7 *and* G_8 *are indistinguishable under DDH assumption. Concretely, for any secret* s*, if* $T_{\mathcal{A}} \leq T_{ddh} - T_U - [1 + (14n + 12)q_S + (5n + 12)q_U]t_{exp}$ *then* $|\Pr[\mathsf{F}_{7,s}] - \Pr[\mathsf{F}_{8,s}]| \leq \epsilon_{ddh}$.

Proof. The claim easily follows by constructing a reduction that embeds the DDH challenge (A, B, C) as follows: $y \leftarrow A$, $c_{\tilde{p},L} \leftarrow B$ and $c_{\tilde{p},R} \leftarrow Ch^p$.