Partitioned Group Password-Based Authenticated Key Exchange

Dario Fiore[1], María Isabel González Vasco[2], Claudio Soriente[3]

[1] IMDEA Software Institute, Madrid, Spain;
dario.fiore@imdea.org
[2] MACIMTE, Universidad Rey Juan Carlos, Madrid, Spain;
mariaisabel.vasco@urjc.es
[3] Telefónica Research, Barcelona, Spain;
claudio.soriente@telefonica.com

Abstract. Group Password-Based Authenticated Key Exchange (GPAKE) allows a group of users to establish a secret key, as long as *all of them* share the same password. However, in existing GPAKE protocols as soon as one user runs the protocol with a non-matching password, all the others abort and no key is established. In this paper we seek for a more flexible, yet secure, GPAKE and put forward the notion of *partitioned* GPAKE. Partitioned GPAKE tolerates users that run the protocol on different passwords. Through a protocol run, any subgroup of users that indeed share a password, establish a session key, factoring out the "noise" of inputs by users holding different passwords. At the same time any two keys, each established by a different subgroup of users, are pair-wise independent if the corresponding subgroups hold different passwords. We also introduce the notion of *password-privacy* for partitioned GPAKE, which is a kind of *affiliation hiding* property, ensuring that an adversary should not be able to tell whether any given set of users share a password. Finally, we propose an efficient instantiation of partitioned GPAKE building on an unforgeable symmetric encryption scheme and a PAKE by Bellare et al. [6]. Our proposal is proven secure in the random oracle/ideal cipher model, and requires only two communication rounds.

1 Introduction

Password Authenticated Key Exchange (PAKE) [6,10] allows two parties sharing a (short, low-entropy) password to agree on a session key, even in presence of active adversaries. Group PAKE (GPAKE) is the natural extension to PAKE that empowers groups of more than two users to establish a session key, given that they share a common password. Constant-round GPAKE protocols exist in both the standard (e.g., [2]) and the random oracle model (e.g., [3]).

To the best of our knowledge, no GPAKE protocol tolerates users holding different passwords. That is, prior to engaging in a GPAKE protocol, users must identify the purported group members claiming to hold *the* password. The GPAKE protocol, then, allows to prove knowledge of the password (and to establish a session key). However, if just one user engages in a protocol execution with a different password, she is regarded as an active adversary and causes the users to abort (even if all other users do share the same password). The same principle is actually applied in (non-password) group key exchange protocols [23]; whenever authentication fails, users typically abort the protocol execution and no joint key is established among those who successfully authenticate each others.

In this paper, we pursue a more flexible design for group key exchange, where *robustness with respect to authentication failures* is provided. Focusing on the password scenario, we consider a setting in which users are not aware in advance of who else in the group of participants may actually share a password with them. When the protocol is over, any subset of users that indeed share a password, establish a session key (with corresponding correct partner identifier – pid – and session identifier – sid), factoring out the "noise" of inputs by users holding different passwords. Passwords, therefore, induce a partition on the set of users.

Once the protocol is over, users in each subset of this partition obtain a shared secret key, whereas keys established by different subsets are pair-wise independent.[4]

By definition, an execution of partitioned GPAKE does not "abort" when it involves parties with a different password. Rather, it factors out messages by those parties and allows to compute a shared key among users who do share a password. We remark that the lack of abort does not weaken the security against online attacks. In both our protocols and a traditional GPAKE with aborts, a user can indeed tell if some parties have input a different password by checking if they have been included in the partner identifier. Therefore, as in the case of aborts, it is up to the specific application scenario to decide whether the other party should be granted another attempt.

Partitioned GPAKE finds natural application in ad-hoc scenarios. For example, in an Internet-of-Things (IoT) swarm, devices belonging to the same user may need to establish a shared key (assuming that all devices of a given user have been initialized with the same password). Moreover, in a multi-user scenario, different IoT swarms belonging to different users will co-exist and key establishment in one swarm should not affect the others.

Also, partitioned GPAKE may surprisingly be used in scenarios where sharing a password is not desirable. For example, password reuse across accounts is considered bad practice because of the consequences that a security breach at a service provider may have for others. By using partitioned GPAKE several service providers may identify those individuals who re-use the same password across accounts. Of course, this usage of partitioned GPAKE is only possible when servers actually store the passwords for authentication (or a deterministic function of them).

Related work. Our goal is to accomplish secure and efficient designs for GPAKE which are in addition resilient to authentication failures. With a similar (yet more general) motivation, a notion of *robustness* has been defined for group key establishment protocols: a key establishment protocol is defined as *robust* if it runs to completion even if some players "fail" during a protocol execution [4,11,17,19,24]. At this, failures are generic and not necessarily linked to authentication, which is our main concern here. Rather, they reflect the situation in which one of the involved participants actually disappears before completing all interaction phases described in the protocol specification. Many of these proposals are inefficient, while others make strong assumptions on the failure probabilities of users, on the existence of synchronization measures or on the communication network itself. The most efficient ones [24,19] actually assume to have at hand reliable and authenticated broadcast channels.

GPAKE protocols, including our work, share some similarities with Affiliation-Hiding Authenticated Key Exchange (AH-AKE) [18,25] protocols. AH-AKE allows users affiliated with the same authority (i.e., holding a credential issued by that authority) to establish a secret key, without disclosing their affiliation to eavesdroppers or users holding non-matching credentials. Some AH-AKE are realized using pseudonyms and are often linkable, i.e., the involvement of a given user can be recognized across multiple sessions [5,12,18], yet there are also unlinkable AH-AKE [20]. Affiliation-Hiding Authenticated Group Key Agreement protocols (AH-GAKE) [16,17,26] are the natural extension of AH-AKE to the multiparty setting. These proposals call for some heavy infrastructure involving authorities and publicly available revocation lists. In contrast, GPAKE focuses on a simpler and somewhat more realistic scenario where users simply hold passwords, rather than credentials, and there is no revocation.[5] We note that the affiliation hiding property of AH-GAKE prevents an adversary from telling whether two users share an affiliation. In our setting, affiliation is password-defined, and borrowing from AH we define *password-privacy* (see Section 2.3) that strengthens the privacy provisions of partitioned GPAKE protocols.

Contributions. We introduce here the new notion of *partitioned* GPAKE, aiming at designs suited for scenarios where the specific group of users sharing a password is not known a priori. We thus augment

[4] A user not sharing her password with any other user is assigned to a singleton subset and therefore obtains a key only known to her.

[5] In a scenario where users are "affiliated" to a group by the knowledge of a password, revocation is only possible if all non-revoked members update the group password.

the correctness requirements from standard GPAKE, while the usual key-secrecy guarantees must also be attained. Further, we define *password-privacy* as a kind of affiliation-hiding property. Once the security model under consideration has been made explicit in Section 2, we give in Section 3 a design of a *partitioned* GPAKE building on an unforgeable symmetric encryption scheme and a PAKE by Bellare et al. [6]. We further prove it attains key secrecy and password-privacy in the ideal cipher/random oracle model. Our proposal is simple and efficient, as it requires only two rounds and, during the first one, each participant essentially broadcasts the (ideal-cipher) encryption of a single group element, regardless of the number of users engaged in the protocol execution. Our main efficiency advantage comes from the fact that users must not run a preliminary phase in order to recognize their partners on the actual GPAKE. In the concluding section, we further comment on possible variants of our design that may be proven secure without idealized assumptions, at the price of losing efficiency.

2 Security Model and Security Goals

Similar to previous work, we assume a public password dictionary $\mathcal{D} \subseteq \{0,1\}^*$ to be efficiently recognizable and of constant or polynomial size. In particular, we assume that a polynomially bounded adversary is able to enumerate $\mathcal{D}$. The set $\mathcal{S} = \{U_1, \ldots, U_N\}$ of users is partitioned in $l \geq 2$ disjoint subsets, such that $\mathcal{S} = \mathcal{U}_1 \cup \mathcal{U}_2 \cdots \cup \mathcal{U}_l$. All users in $\mathcal{U}_\delta$, for $\delta = 1, \ldots, l$ share a common password $pw^\delta \in \mathcal{D}$, with $pw^\delta \neq pw^\gamma$ given $\delta \neq \gamma \in \{1, \ldots, l\}$. For the sake of simplicity, we assume all passwords are chosen uniformly at random from $\mathcal{D}$, and are represented by bitstrings of the same size (denoted by k).

2.1 Communication Model and Adversarial Capabilities

Protocol instances. Users are modeled as probabilistic polynomial time (ppt) Turing machines. Each user $U \in \mathcal{S}$ may execute a polynomial number of protocol *instances* in parallel and we use Π_i^j to refer to the j-th instance of user i, which can be considered as a process executed by U_i. To each instance we assign seven variables:

- used_i^j indicates whether this instance is being or has been used for a protocol run;
- state_i^j keeps the state information needed during the protocol execution;
- term_i^j indicates if the execution has terminated;
- sk_i^j stores the session key once it is accepted by Π_i^j. Before acceptance, it stores a distinguished NULL value;
- sid_i^j denotes a (possibly public) session identifier that can serve as an identifier for the session key sk_i^j;
- pid_i^j stores the set of identities of those users that Π_i^j establishes a key with—including U_i himself; [6]
- acc_i^j indicates if the protocol instance was successful, i. e., the user accepted the session key.

For more details on the usage of the variables we refer to the work of Bellare et al. in [6].

Communication network. Arbitrary point-to-point connections among the users are assumed to be available. The network is, however, non-private and fully asynchronous. More specifically, it is controlled by the adversary, who may delay, insert and delete messages at will.

[6] This will be defined dynamically. In previous work on PAKE, pid is set a priori and stores the set of identities of those users who claim to share a password. In our setting, this is unknown since users engage in a protocol execution without knowing *which* password the others hold.

Adversarial capabilities. We restrict to probabilistic polynomial time (ppt) adversaries. The capabilities of an adversary $\mathcal{A}$ are made explicit through a number of *oracles* allowing $\mathcal{A}$ to communicate with protocol instances run by the users:

- $\mathsf{Send}(U_i, j, M)$. Sends message M to the instance Π_i^j of U_i and returns the reply generated by this instance. If $\mathcal{A}$ queries this oracle with an unused instance Π_i^j and M being the set of users $\{U_{i_1}, \ldots, U_{i_\mu}\} \subseteq \mathcal{S}$, engaging in the protocol (including U_i), then the flag used_i^j is set, and the first protocol message of Π_i^j for initializing a protocol run involving $\{U_{i_1}, \ldots, U_{i_\mu}\}$ is returned.
- $\mathsf{Execute}(\{\Pi_{i_1}^{j_1}, \ldots, \Pi_{i_\mu}^{j_\mu}\})$. Executes a complete protocol run among the specified unused instances of the respective users. The adversary obtains a transcript of all messages sent over the network. A query to the Execute oracle is supposed to reflect passive eavesdropping. In particular, no password online-guess can be implemented with this oracle.
- $\mathsf{Reveal}(U_i, j)$. Yields the session key sk_i^j (if this has been defined).
- $\mathsf{Test}(U_i, j)$. Only one query of this form is allowed for an active adversary $\mathcal{A}$. Provided that sk_i^j is defined, (i. e. $\mathsf{acc}_i^j = \mathsf{true}$ and $\mathsf{sk}_i^j \neq \text{NULL}$), $\mathcal{A}$ can issue this query at any time when being activated. Then with probability 1/2 the session key sk_i^j and with probability 1/2 a uniformly chosen random session key is returned.
- $\mathsf{Corrupt}(U_i)$. Returns the password pw_i held by U_i.

2.2 Correctness and Key Secrecy

Correctness. Our definition of correctness extends the standard one in GPAKE. Namely, without active adversarial interference, it should be the case that users holding matching passwords end up establishing a common session key as intended and assigning the same name (sid) to it. Furthermore, messages from users with non-matching password should not disrupt session key computations.

Definition 1 (Correctness). *Let $\mathcal{D}$ be a dictionary and $\mathcal{S}$ be a set of users as described earlier. Then, a* partitioned *group password-based key establishment protocol* P *is* correct *if in the presence of a passive adversary $\mathcal{A}$ —i. e., $\mathcal{A}$ only uses the* Execute *oracle—a single execution of the protocol among $U_{i_1}, \ldots, U_{i_\mu}$ involves μ instances $\Pi_{i_1}^{j_1}, \ldots, \Pi_{i_\mu}^{j_\mu}$ and ensures that with overwhelming probability all instances:*

- *accept, i. e.,* $\mathsf{acc}_{i_1}^{j_1} = \cdots = \mathsf{acc}_{i_\mu}^{j_\mu} = \mathsf{true}$*;*
- *users belonging to the same subset $\mathcal{U}_\tau$ of the password-induced partition on $\mathcal{S}$ have accepted the same session key associated with the common session and partner identifier, that is*

$$\forall s, r \in \{1, \ldots, \mu\} \text{ whenever } U_{i_s}, U_{i_r} \in \mathcal{U}_\tau,$$

it holds

$$\mathsf{sk}_{i_s}^{j_s} = \mathsf{sk}_{i_r}^{j_r} \neq \text{NULL}, \ \mathsf{sid}_{i_s}^{j_s} = \mathsf{sid}_{i_r}^{j_r} \text{ and } \mathsf{pid}_{i_s}^{j_s} = \mathsf{pid}_{i_r}^{j_r} \neq \text{NULL}.$$

(Note that if U_{i_s} is the only user in $\mathcal{U}_\tau$, then she will end up with unique $\mathsf{pid}_{i_s}^{j_s}$, $\mathsf{sid}_{i_s}^{j_s}$, and $\mathsf{sk}_{i_s}^{j_s}$.)

Key Secrecy. Here we define the main security notion of partitioned GPAKE protocols. In order to do so, we introduce the notions of *partnering* and *freshness* to express which instances are associated in a common protocol session, and how to rule out trivial attacks, respectively.

Partnering. We adopt the notion of partnering from [9] where instances Π_i^j, Π_t^m are *partnered* if $\mathsf{sid}_i^j = \mathsf{sid}_t^m$, $\mathsf{pid}_i^j = \mathsf{pid}_t^m$ and $\mathsf{acc}_i^j = \mathsf{acc}_t^m = \mathsf{true}$. However, in [9] pid lists user instances engaging in a common protocol execution. In our scenario, pid explicits instances that engage in a common protocol execution *and* share a password. In other words, in [9] and in other GPAKE proposals, a user defines pid at the beginning of the protocol, while in our settings, a user *discovers* pid at the end of the protocol.

Note that the above notion of partnering defines an equivalence relation on the set of possible instances (namely, it is reflexive, symmetric and transitive). Further, to avoid trivial cases we assume that an instance Π_i^j always accepts the session key constructed at the end of the corresponding protocol run, if no deviation from the protocol specification occurs. Moreover, without adversarial interference all users in the same protocol session belonging to the same subset $\mathcal{U}_k$, i.e., holding the same password, should come up with the same session key, store it under the same session identifier and be aware of whom they share it with.

Freshness. This notion helps specifying under which conditions a Test-query can be executed by the adversary in the security experiment. An instance Π_i^j is called *fresh* if the adversary never made one of the following queries:

- $\mathsf{Corrupt}(U_t)$ to any U_t holding the same password as U_i (i.e., so that U_i and U_t are both in $\mathcal{U}_\tau$ for some $\tau \in \{1, \ldots, l\}$);
- $\mathsf{Reveal}(U_t, m)$ with Π_i^j and Π_t^m being partnered.

The notion of freshness allows to rule out trivial attacks. In particular, revealing a session key from an instance Π_i^j clearly yields the session key of all instances partnered with Π_i^j and, therefore, this kind of "attack" is not taken into account for the security definition. Also, note that this definition of freshness implies that corrupting users which hold a different password from the one held by the uses specified in the Test query, should be of no help to the adversary.

Key secrecy. Now that we have introduced the notions of partnering and freshness, we are ready to fully define key secrecy. As typical in password-based protocols, we observe that since the dictionary $\mathcal{D}$ has polynomial size we cannot prevent an adversary from correctly guessing a password $pw \in \mathcal{D}$ used by any of the users. Therefore, our goal is to restrict the adversary $\mathcal{A}$ to such online-verification of password guesses.

In the above setting, for a fixed group key establishment protocol $\mathcal{P}$, let $\mathsf{Succ}(\ell)$ be the probability that an adversary $\mathcal{A}$ queries Test on a fresh instance Π_i^j and guesses correctly the bit b used by the Test oracle in a moment when Π_i^j is still fresh. Now we define $\mathcal{A}'$s advantage as the function

$$\mathsf{Adv}_{\mathcal{A}}(\ell) := |2 \cdot \mathsf{Succ}(\ell) - 1|.$$

We now introduce a function ε to capture the weaknesses that may originate in the employed authentication technique; namely, as the adversary might be able to guess passwords online, ε will explicit a bound on $\mathcal{A}'$s probability of guessing a shared password.

Definition 2 (Key-secrecy). *Let $\mathcal{P}$ be a correct partitioned group password-authenticated key establishment protocol, with $\mathcal{D}$ and $\mathcal{S}$ as described above. Let $\mathcal{A}$ be a probabilistic polynomial time adversary having access to the* $\mathsf{Execute}$, Send, Reveal *and* $\mathsf{Corrupt}$ *oracles. We say that $\mathcal{P}$ provides* key secrecy*, if for every such $\mathcal{A}$, running in the experiment described in Section 2.1 and querying the* Send *oracle to at most* q *instances, the following inequality holds for some negligible function* negl *and some function ε which is at most linear in its second variable* q,

$$\mathsf{Adv}_{\mathcal{A}}(\ell) \leq \varepsilon(\ell, \mathsf{q}) + \mathsf{negl}(\ell).$$

Note that assuming passwords are selected uniformly at random and only a constant number of passwords can be checked by the adversary on each on-line attack, it holds $\varepsilon(\ell, \mathsf{q}) = \mathcal{O}\left(\frac{\mathsf{q}}{|\mathcal{D}|}\right)$.

Remark 1. Typically, in GAKE the Corrupt oracle is used to model different flavours of forward security, i.e., to establish to what extent the leakage of authentication keys compromises the security of previously agreed session keys. In our scenario, however, *corrupted* users are to be understood as adversaries who might actually be legitimate members of a different password-defined subset $\mathcal{U}_\delta$. Thus, our model implicitly states that everyone who is not in the same password-defined subset is understood as under adversarial control.

2.3 Password privacy

In this section we introduce a security notion for partitioned GPAKE protocols that we call *password-privacy*. Very intuitively, *password-privacy* ensures that an active adversary should not gain any information on the passwords used by legitimate users, so he should not even be able to tell whether a given set of users actually share the same password or not unless he has guessed the involved password(s). Basically, if we consider the partition on the set of users induced by the password assignment, then the adversary should not learn information about such partitions beyond what he may get by just making wild guesses.

It is interesting to note that such a notion is not relevant in many GPAKE proposals, as by design messages constructed from a non-matching password are typically recognized as adversarially generated and result in an abort (see for instance [1,2,3]). Indeed, in such scenario an active adversary may learn if two users U_i and U_t share the same password by starting a new session involving U_i and replaying him messages constructed by U_t in a different execution. Now, the adversary just observes whether this rouge session is aborted or not. In contrast, in partitioned GPAKE protocols, executions always succeed and at their end every participant ends up with a valid key (even if only participants sharing the same password will share the same session key).

Our notion of password-privacy is rather inspired to that of affiliation hiding [18,25] considered in authenticated key exchange. Affiliation hiding implies that an active adversary should not be able to obtain any information on group membership through a protocol execution (without considering trivial attacks where the adversary shares the affiliation of the victims). In particular, an adversary should not be able to tell whether two users share the same affiliation or not. In our scenario, this translates into guaranteeing that no active adversary should gain information on which users do share a password, assuming he has not guessed the password used by any/some of them.

We model password-privacy with a kind of indistinguishability game where the adversary $\mathcal{A}$ interacts with a challenger. First, he chooses the victim subgroup $\mathcal{U} \subseteq \mathcal{S}$ and two partitions P_0 and P_1 of it. Then the challenger randomly selects one of the two partitions and assigns passwords (chosen uniformly at random) consistently to the corresponding subgroups. $\mathcal{A}$ wins if it can tell which of the two partitions has actually been chosen by the challenger, under the restriction that $\mathcal{A}$ cannot query the Reveal or Corrupt oracles on any of the users in $\mathcal{U}$.[7] We stress that in our game we make no assumption about the passwords of all the remaining users in $\mathcal{S} \setminus \mathcal{U}$; these passwords can be even chosen by the adversary (i.e., the adversary can simulate any of these users on its own). This reflects the fact that our definition models the privacy of passwords not known by the adversary.

Definition 3 (Password-privacy). *Let $\mathcal{P}$ be a correct partitioned* GPAKE *protocol. Consider a public dictionary $\mathcal{D}$ and (potential) set of users $\mathcal{S} = \{U_1, \ldots, U_N\}$, where N is polynomial in the security parameter ℓ.*
Let $\mathcal{A}$ be a probabilistic polynomial time adversary interacting with a challenger Ch *in the following game:*

1. *$\mathcal{A}$ selects a set of users $\mathcal{U} \subseteq \mathcal{S}$, and two partitions P_0 and P_1 of $\mathcal{U}$*
2. Ch *chooses a bit $b \in \{0,1\}$ uniformly at random and assigns a password, also chosen uniformly at random from the dictionary, for each subgroup of the partition P_b. Further, he follows the specification of $\mathcal{P}$.*

[7] Otherwise, $\mathcal{A}$ could trivially win the game through direct Corrupt calls or by testing if two users share the same session key at the end of the protocol execution through the corresponding Reveal calls.

3. $\mathcal{A}$, *equipped with* Send *and* Execute, *must output a guess* b' *and wins if* $b' = b$.[8]

We say that $\mathcal{P}$ *achieves* password-privacy *if every p.p.t.* $\mathcal{A}$ *wins the above password-privacy game with (at most) negligible probability over a random guess, provided he did not guess any password from a user in* $\mathcal{U}$. *More precisely, for every p.p.t. let* $\mathsf{Succ}(\ell)$ *be the probability that an adversary* $\mathcal{A}$ *guesses correctly the bit* b *selected by* Ch. *Now we define* $\mathcal{A}$*'s advantage as the function*

$$\mathsf{Adv}^{\mathsf{pwpri}}_{\mathcal{A}}(\ell) := |2 \cdot \mathsf{Succ}(\ell) - 1|.$$

Let q *denote the number of instances to which* $\mathcal{A}$ *has made a* Send *query. Then a protocol* $\mathcal{P}$ *has password-privacy if the following holds for some negligible function* negl *and some function* ε *which is at most most linear in* q,

$$\mathsf{Adv}^{\mathsf{pwpri}}_{\mathcal{A}}(\ell) \leq \varepsilon(\ell, \mathsf{q}) + \mathit{negl}(\ell).$$

3 Our Construction of Partitioned GPAKE

We propose an instantiation of a partitioned GPAKE that satisfies key secrecy and password-privacy. Our idea builds on the two-party password based key exchange protocol presented by Bellare, Pointcheval and Rogaway in [6], and is also inspired by an n-party *private equality test* (PET) protocol by Gelles, Ostrovsky and Winoto (see Protocol 4 of [15]), which allows a distinguished user (Alice) to detect whether the $n-1$ private inputs of her peers are actually equal.

At first glance it may seem that partitioned GPAKE may be achieved by directly combining a PET protocol with a "standard" GPAKE. It is however not clear how to end up with a correct and efficient system in that fashion. Let us first note that assuming $|\mathcal{D}| > N$ the number of possible dictionary-induced partitions of $\mathcal{S}$ is of size super-exponential in N,[9] thus directly executing a PET protocol to check on every possible partition is simply infeasible. Furthermore, constructions for n-party PET are asymmetric: upon termination a distinguished user (Alice) learns whether or not all private inputs are equal while the other parties learn nothing. Thus, once a group of users sharing the same password is identified via PET subsequent communication would be required so that each user learns who they share a password with before the actual GPAKE starts.

Our approach is easier to understand with a high-level description in two stages: during the first stage, users from $\mathcal{S}$ run pair-wise PAKE in such a way that, after the execution, any two users establish a common key, provided they both hold the same password—users holding different passwords will get as output from this stage random independent keys. Let us denote by $sk_{i,t}$ the (first stage) output two-party key that user U_i stores as shared with U_t. Indeed, if U_i and U_t hold matching passwords, the corresponding two-party keys will coincide, i.e., $sk_{i,t} = sk_{t,i}$. Otherwise, $sk_{i,t}$ and $sk_{t,i}$ will be chosen independently at random from a fixed key space. Notice that this first stage can be carried through with the same number of communication rounds as the underlying PAKE. During the second stage, each user picks a random key contribution and sends it to each other user, symmetrically encrypting it with the shared key established before. That is, U_i selects a random nonce r_i and encrypts it for each $t \neq i$ using $sk_{i,t}$. To conclude each user computes the final key by combining the key contributions that she can decrypt (including her own). Note this second stage is completed with one only communication round.

[8] We remark here that $\mathcal{A}$ does not have the Reveal oracle when confronting the password-privacy game. Indeed, it would trivially allow him to learn whether two users share a password or not (moreover, simply by executing a session involving all users, he would learn through reveal queries the password induced partition in $\mathcal{S}$ — for the case $\mathcal{S}$ polynomial).

[9] It is actually B_N, the so-called N-th Bell number. The following recursive formula is satisfied: $B_N = \sum_{k=0}^{n} \binom{N}{k} B_k$, see [13].

The security guarantees of each proposal following the structure depicted above, clearly depends on the concrete choices for the underlying PAKE and symmetric encryption scheme. Indeed, for achieving password-privacy the PAKE underlying our proposal should provide assurance that the message flow from the first stage does not leak any information on the initial password-induced partition. Furthermore, we must also require that the second-stage encryptions of (random) values using the same keys are not linkable. In the next section, we describe the specific building blocks that we adopted to build a secure protocol following the above idea. Aiming at an efficient design, we also select a PAKE with the nice feature that each user must only broadcast one message during the first stage. This message enables her to establish an independent two-party key with any other user sharing the same password.

3.1 Tools

Bellare, Pointcheval and Rogaway PAKE. Our main building block is the EKE2 PAKE presented by Bellare et al. in [6] which is secure in the so-called *ideal cipher* model (see [8]). In this model, it is assumed that there exists a publicly accessible random block cipher with a k-bit key and a n-bit input/output,[10] that is chosen uniformly at random among all block ciphers of this form. Further it is needed to assume that *ideal* random function exists, namely, we will model a hash function H used in the key derivation process as a random oracle [7]. These two models can actually be proven equivalent, as evidenced in [14].

Informally, EKE2 can be described as a Diffie-Hellman key exchange where message flows are encrypted using the password as secret key, and the two-party key is the random oracle image of the Diffie-Hellman triplet concatenated with the users names. Further, session identifiers are constructed concatenating the message flow.

Unforgeable encryption. For the choice of our second building block, a symmetric encryption scheme Π, we will select a construction which fulfills a very strong notion of unforgeability; namely, we should not even allow the adversary to produce *any* new valid ciphertext without the private key. Such property is defined in [22] as *existential unforgeability*; we adapt Definition 5 from that paper here;

Definition 4. *Let $\Pi = (\mathsf{KEYGEN}, \mathsf{ENC}, \mathsf{DEC})$ be a private-key encryption scheme. Let ℓ be the security parameter and $\mathcal{A}$ be any pptm algoritm. Define*

$$\mathsf{Adv}^{\mathit{exist}}_{\mathcal{A},\Pi}(\ell) = Pr\left[sk \leftarrow \mathsf{KEYGEN}(1^\ell); y \leftarrow \mathcal{A} : \mathsf{DEC}_{sk}(y) \neq \perp\right].$$

At this, y is produced by the adversary $\mathcal{A}$ which may use an encryption oracle $\mathcal{E}_{sk}$, yet y must not have been directly returned by $\mathcal{E}_{sk}$. We say that Π is $(t, p, b; \delta)$-secure in the sense of existential unforgeability if for any adversary $\mathcal{A}$ which runs in time at most t and asks at most p queries to the encryption oracle, these totaling at most b bits[11], we have $\mathsf{Adv}^{\mathit{exist}}_{\mathcal{A},\Pi}(\ell) \leq \delta(\ell)$.

If δ is negligible in ℓ, we will simply say that Π is an unforgeable encryption scheme.

Furthermore, in Theorem 1 of [22] it is proven that unforgeability along with chosen plaintext security implies adaptive chosen ciphertext security. For our generic construction we will make use of a symmetric key encryption scheme Π secure in this sense, thus, we may assume that the adversary will not be able to produce any valid ciphertext, nor to gain any information on the plaintexts underlying encrypted values.

3.2 Our construction

Now we are ready to present our concrete construction, which is depicted in Figure 1. Its main building blocks are:

[10] In our construction, k is the bit size of passwords and n is the number of bits needed to represent elements in G

[11] Obviously t, p and b, as well as δ must be seen as functions on ℓ

– a hash function H, wich will be modelled as a random oracle; we assume it to range on $\{0,1\}^d$, for d polynomial in the security parameter ℓ,
– a private key encryption scheme $\Pi = (\mathsf{KEYGEN}, \mathsf{ENC}, \mathsf{DEC})$, assumed to be secure in the the sense of existential unforgeability and achieving chosen ciphertext security (see [22] and section 4 above). For each choice of the security parameter, we will denote by $\mathcal{P}$ and $\mathcal{C}$ the corresponding polynomial sized plaintext and ciphertext spaces, and assume $\mathcal{P}$ to be an additive group.[12] Furthermore, we will assume KEYGEN selects keys uniformly at random from the range of the random oracle H. [13]
– an ideal cipher $\mathcal{E} : \mathcal{D} \times G \longmapsto \hat{G}$, where $\mathcal{D}$ is the password dictionary, G is a cyclic group of order q (polynomial in ℓ) and $\hat{G}$ is a finite set of q elements.

Round 1

Broadcast. Each U_i chooses uniformly at random a value $x_i \in \{1, \ldots, q-1\}$ and broadcasts

$$M_i^1 := (U_i, Y_i), \quad \text{where} \quad Y_i := \mathcal{E}_{pw_i}(g^{x_i})$$

Computation. For every received message (U_t, Y_t), the user U_i sets

$$\mathsf{sid}_{i,t} = U_i||Y_i||U_t||Y_t.$$

Further, if $\mathcal{E}_{pw_i}^{-1}(Y_t) = X_t \neq \perp$, U_i sets

$$\mathsf{sk}_{i,t} := H(U_i\|U_t\|X_i\|X_t\|X_t^{x_i}),$$

otherwise U_i selects $\mathsf{sk}_{i,t}$ uniformly at random in the range of H.
As a result, for every U_t holding the same password as U_i, user U_i defines a two-party key

$$\mathsf{sk}_{i,t} := H(U_i\|U_t\|g^{x_i}\|g^{x_t}\|g^{x_i x_t}),$$

and a matching session identifier[a].

Round 2:

Broadcast. Each user U_i selects u.a.r. $r_i \in \mathcal{P}$, and sends, for each $t \neq i$,

$$M_{it}^2 := (U_i, \mathsf{sid}_{i,t}, a_{it}) := (U_i, \mathsf{sid}_{i,t}, \mathsf{ENC}_{\mathsf{sk}_{i,t}}(r_i))$$

Computation. For every received message $(U_t, \mathsf{sid}_{t,i}, a_{ti})$, user U_i computes $c_{it} := \mathsf{DEC}_{\mathsf{sk}_{it}}(a_{ti})$ and sets $\mathsf{pid} = \{i\} \cup \{t : c_{it} \neq \{r_i, \perp\}\}$. Further, for each $t \in \mathsf{pid}, t \neq i$, it sets $r_t^* := c_{it}$ and also $r_i^* := r_i$.

Session Key/session identifier definition. User U_i sets $\mathsf{acc}_i :=$true, derives the (subgroup) key as the addition

$$\mathsf{sk}_i := \sum_{l \in \mathsf{pid}} r_l^*,$$

and also the session identifier[b]

$$\mathsf{sid}_i = \{\mathsf{sid}_{i,t}||a_{i,t}\}_{t \in \mathsf{pid}_i}||\mathsf{pid}_i.$$

[a] assuming $i < t$, i.e., users inputs are displayed ordered in the two party session identifiers.
[b] again, here some prefixed ordering is assumed in order to attain consistency of the sid's.

Fig. 1. An efficient construction (ROM + Ideal Cipher)

[12] given the security parameter ℓ, we should write $\mathcal{P}_\ell$ and $\mathcal{C}_\ell$. We however drop the subscripts for simplicity.
[13] As a result, from the key secrecy of the Bellare-Pointcheval-Rogaway PAKE it follows that two-party keys output from it are (computationally) indistinguishable from keys output by KEYGEN.

Theorem 1. *Let $\Pi = (\mathsf{KEYGEN}, \mathsf{ENC}, \mathsf{DEC})$ be a symmetric encryption scheme which is both unforgeable and chosen plaintext semantically-secure. Then, the protocol from Figure 1 is a correct partitioned password based group key agreement achieving key secrecy as defined in Definition 2 and password-privacy as defined in Definition 3 under the computational Diffie-Hellman assumption in group G in the random oracle/ideal cipher model.*

Proof.

Correctness. In an honest execution of the protocol, it is easy to verify that all participants in the protocol will terminate by accepting and computing the same session identifier and session key as those who hold the same password.

Key Secrecy. The proof is set up in terms of several experiments or games, where a challenger interacts with the adversary confronting him with a counterfeit Test-challenge in the spirit of Definition 2. From game to game the challenger's behavior somehow deviates from the previous, with the corresponding impact on $\mathcal{A}$'s success probability. Following standard notation, we denote by $\mathsf{Adv}(\mathcal{A}, G_i)$ the advantage of the adversary when confronted with Game i. The security parameter is denoted by ℓ.

In the sequel, we denote by q_{exe} the number of $\mathsf{Execute}$ calls made by the adversary. Also q will denote the number of instances to which the adversary has made a Send query, thus, it is the number of instances that have suffered on-line attacks. Similarly, q_{ro} will denote the number of queries $\mathcal{A}$ makes to the hash function H.

Game 0. This first game corresponds to a real attack, in which all the parameters are chosen as in the actual scheme. By definition, $\mathsf{Adv}(\mathcal{A}, G_0) = \mathsf{Adv}(\mathcal{A})$.

Game 1.

We assume that the hash function H is simulated as a Random Oracle. Namely, every time a new query α is asked, the simulator selects u.a.r a value h_α from the range of H and stores the pair α, h_α in a table (from now on, the $\mathsf{H\text{-}list}$). Should the value α be queried again, the simulator will look in the $\mathsf{H\text{-}list}$ and forward h_α as answer.

Furthermore, we explicit here the ideal cipher simulation. For a given password pw, the simulator will maintain an $\mathsf{IC}_{pw}\text{-}\mathsf{list}$ in which for every query (pw, g) he stores a different value $\hat{g}$ which is selected uniformly at random in $\hat{G}$.[14] Similarly, he also maintains a list capturing the decryption calls done to the ideal cipher, which we may denote $\mathsf{IC}_{pw}^{-1}\text{-}\mathsf{list}$. Thus, a bijection $\sigma_{pw} : G \longmapsto \hat{G}$ and ist inverse are actually explicited by the two lists $\mathsf{IC}_{pw}\text{-}\mathsf{list}, \mathsf{IC}_{pw}^{-1}\text{-}\mathsf{list}$.

The Random Oracle and the Ideal Cipher assumptions are made explicit by assuming

$$\mathsf{Adv}(\mathcal{A}, G_1) = \mathsf{Adv}(\mathcal{A}, G_0).$$

Game 2. In this game, we exclude certain collisions of values chosen uniformly at random in different sessions. Namely, this game aborts in case the same exponent in Round 1 or the same random contribution in Round 2 is selected in different sessions by two (non-necessarily distinct) honest users. Similarly we exclude the event that an H-collision occurs at the time of extracting different two-party keys or session identifiers at the end of Round 1 in different protocol executions.

It is not hard to see that the difference between the two games

$$\left|\mathsf{Adv}(\mathcal{A}, G_2) - \mathsf{Adv}(\mathcal{A}, G_1)\right|$$

is bounded by the probability of "partial collisions" on independent transcripts, which is in turn bounded by

[14] The simulator selects $\hat{g}$ u.a.r. from $\hat{G}$; should the selection be already on the $\mathsf{IC}_{pw}\text{-}\mathsf{list}$, it is discarded and a new value is selected.

$$\frac{q_{\mathsf{ro}}^2}{2^d} + N^2\left(\frac{1}{|G|} + \frac{1}{|\mathcal{P}|}\right)$$

where $\mathcal{P}$ is the plaintext space for Π, from where the nonces are selected in Round 1.

Game 3. Consider the event that $\mathcal{A}$ queries the random oracle on the 5-tuple

$$(U_i \| U_t \| X_i \| X_t \| Z)$$

such that both the values X_i and X_t were generated by the simulator during the game and $Z = X_t^{x_i}$, (essentially, if $\mathcal{A}$ queries the oracle on a valid CDH tuple). If such event (that we call Bad) happens, the simulation is aborted. Clearly,

$$|\mathsf{Adv}(\mathcal{A}, G_3) - \mathsf{Adv}(\mathcal{A}, G_2)| \leq P(\mathsf{Bad}).$$

It is easy to see that for any adversary $\mathcal{A}$ that cause the Bad event to happen it is possible to construct another adversary $\mathcal{B}$ against the CDH assumption. The reduction is rather straightforward. $\mathcal{B}$, on input g^x, g^y, chooses a random index $q^* \leftarrow \{1, \ldots, q_{\mathsf{exe}}\}$ and two user indices $i, t \leftarrow \{1, \ldots, N\}$ also at random. Then, in the q^*-th protocol execution requested by the adversary $\mathcal{B}$ sets $X_i = g^x$ and $X_t = g^y$ for the users i and t respectively. Finally, in the end of the game, it selects one random entry from $\mathsf{H\text{-}list}$ such that among the ones with $X_i = g^x$ and $X_t = g^y$, and returns the last value Z of the tuple. Clearly, if the event Bad occurs, and $\mathcal{B}$ guessed correctly the indices i, t and q^*, then $\mathcal{B}$ found a solution for the CDH problem. Otherwise, if any of the guess was wrong, $\mathcal{B}$ aborts. It is not hard to see that

$$P(\mathsf{Bad}) \times \frac{1}{q_{\mathsf{exe}}} \frac{1}{N^2} \frac{1}{q_{\mathsf{ro}}} \leq \mathsf{Adv}_{G,g}^{\mathsf{CDH}}(\ell),$$

where $\mathsf{Adv}_{G,g}^{\mathsf{CDH}}(\ell)$ is the probability that $\mathcal{B}$ has of winning a computational Diffie-Hellman challenge over G with generator g.

Game 4. Consider the event that $\mathcal{A}$ queries the random oracle on the 5-tuple

$$(U_i \| U_t \| X_i \| X_t \| Z)$$

such that the value X_t was generated by the simulator during the game, pw^* is the (random) password held by U_t whereas X_i is such that $\mathcal{A}$ made a query to the ideal cipher on input (pw^*, X_i) to get Y_i, and a Send query with the input (U_i, Y_i). If such event (that we call $\mathsf{B}ad^*$) happens, the simulation is aborted. Clearly,

$$|\mathsf{Adv}(\mathcal{A}, G_4) - \mathsf{Adv}(\mathcal{A}, G_3)| \leq P(\mathsf{Bad}^*).$$

Now, the probability of the event Bad^* is bounded by the probability of a password guess, which is $\mathcal{O}\left(\frac{\mathsf{q}}{|D|}\right)$.

We remark that from this point on in the simulation all H-queries of the form $(U_i \| U_t \| X_i \| X_t \| X_i^{x_t})$ where users U_i and U_t share a password pw^* are either fully generated by the adversary or fully generated by the challenger. This means that for any two users sharing a password pw^* which has not been guessed by the adversary the corresponding two-party keys will be indistinguishable from random.

Game 5. This game deals with adversaries that only modify messages in Round 2, for the tested instance Π_i^j. Precisely, consider any pair of users (U_i, U_t) for which the adversary had not made a random oracle query as the ones "excluded" in the previous two games. Then if in the second part of the protocol execution the adversary $\mathcal{A}$ sends a valid message M_{it}^2 that decrypts correctly and is not a replay, then the game aborts.

With a simple reduction to the unforgeability of the encryption scheme Π, it is possible to show that

$$|\mathsf{Adv}(\mathcal{A}, G_5) - \mathsf{Adv}(\mathcal{A}, G_4)| \leq \mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{exist}}(\ell)$$

where

$$\mathsf{Adv}^{\mathsf{exist}}_{\mathcal{A},\Pi}(\ell) \leq \delta(\ell),$$

for some negligible function δ.

Game 6. Now, we modify the Execute and Send simulation in that we construct messages a_{it} as encryptions of 0, i.e., $a_{it} := \mathsf{ENC}_{\mathsf{sk}_{it}}(0)$. Precisely, this change is for all pairs of users (U_i, U_t) for which the adversary had not made a random oracle query as the ones excluded in the previous games. By relying on the CCA-security of Π one can argue that

$$|\mathsf{Adv}(\mathcal{A}, G_6) - \mathsf{Adv}(\mathcal{A}, G_5)| \leq \mathsf{Adv}^{\mathsf{CCA}}_{\mathcal{A},\Pi}(\ell)$$

After making this last change, the session key of a fresh session is completely random and independent from the simulated protocol transcript. Therefore, $\mathsf{Succ}(\ell) = 1/2$, and the proof follows by putting together the bounds between the games.

Password Privacy. The security proof for password privacy proceeds very similarly to the one of key secrecy given above. The main idea is that after applying similar game changes as for key secrecy, the protocol messages become independent of the users passwords.

The games are defined as follows.

Game 0. This first game corresponds to a real attack, in which all the parameters are chosen as in the actual scheme. By definition, $\mathsf{Adv}(\mathcal{A}, G_0) = \mathsf{Adv}(\mathcal{A})$.

Game 1. This is the same as Game 1 in the proof of Theorem 1. It simply makes explicit the simulation of the random oracle and the ideal cipher.

$$\mathsf{Adv}(\mathcal{A}, G_1) = \mathsf{Adv}(\mathcal{A}, G_0).$$

Game 2. This is the same as Game 2 in the proof of Theorem 1, and thus

$$\left|\mathsf{Adv}(\mathcal{A}, G_2) - \mathsf{Adv}(\mathcal{A}, G_1)\right| \leq \frac{q^2_{\mathsf{ro}}}{2^d} + N^2\left(\frac{1}{|G|} + \frac{1}{|\mathcal{P}|}\right)$$

where $\mathcal{P}$ is the plaintext space for Π, from where the nonces are selected in Round 1.

Game 3. This is the same as Game 3 in the proof of Theorem 1, and thus

$$|\mathsf{Adv}(\mathcal{A}, G_3) - \mathsf{Adv}(\mathcal{A}, G_2)| \leq q_{\mathsf{exe}} \cdot N^2 \cdot q_{\mathsf{ro}} \cdot \mathsf{Adv}^{\mathsf{CDH}}_{G,g}(\ell)$$

Game 4. This proceeds similarly to Game 4 in the proof of Theorem 1. Let us consider the event that $\mathcal{A}$ queries the random oracle on the 5-tuple

$$(U_i \| U_t \| X_i \| X_t \| Z)$$

such that the value X_t was generated by the simulator during the game, pw^*_t is the (random) password held by U_t [15] whereas X_i is such that $\mathcal{A}$ made a query to the ideal cipher on input (pw^*_t, X_i) to get Y_i, and a Send query with the input (U_i, Y_i). If such event (that we call Bad^*) happens, the simulation is aborted. The difference between this and the previous game lies in the occurrence of event Bad^* whose probability is bounded by that of a password guess. Therefore,

$$|\mathsf{Adv}(\mathcal{A}, G_4) - \mathsf{Adv}(\mathcal{A}, G_3)| \leq P(\mathsf{Bad}^*) = \mathcal{O}\left(\frac{\mathsf{q}}{|D|}\right).$$

[15] Note that it could happen that no other user holds this same password.

Game 5. This is the same as Game 5 in the proof of Theorem 1, and thus

$$|\mathsf{Adv}(\mathcal{A}, G_5) - \mathsf{Adv}(\mathcal{A}, G_4)| \leq \mathsf{Adv}^{\mathsf{exist}}_{\mathcal{A},\Pi}(\ell)$$

where

$$\mathsf{Adv}^{\mathsf{exist}}_{\mathcal{A},\Pi}(\ell) \leq \delta(\ell),$$

for some negligible function δ.

Game 6. Finally, in this game the challenger modifies the Execute and Send simulation by constructing messages a_{it} as encryptions of randomly selected values R_{it}, i.e., $a_{it} := \mathsf{ENC}_{\mathsf{sk}_{it}}(R_{it})$ while the session key is still computed using the randomly sampled values r_i. Based on the CCA-security of Π one can argue that

$$|\mathsf{Adv}(\mathcal{A}, G_6) - \mathsf{Adv}(\mathcal{A}, G_5)| \leq \mathsf{Adv}^{\mathsf{CCA}}_{\mathcal{A},\Pi}(\ell)$$

Furthermore, after making this last change, the protocol messages in the simulation are independent of the password choices, and in particular are distributed identically in both the cases when the users in $\mathcal{U}$ share the same password pw^* or have each a different password. So, the probability that the adversary succeeds in correctly guessing the bit b in this game is $1/2$ – $\mathsf{Succ}(\ell) = 1/2$. Therefore, by putting together the various bounds of the game differences we have that the chances of $\mathcal{A}$ to win the password-privacy game are only negligibly above

$$\frac{1}{2} + \mathcal{O}\left(\frac{\mathsf{q}}{|\mathcal{D}|}\right).$$

4 Final Remarks

It may be worth exploring our design idea aiming at a construction that can be proven secure without idealized assumptions. Having efficiency in mind, our choice of the basic PAKE is optimal, yet using a different scheme for the first stage could remove the need for the random oracle/ideal cipher assumption. A good candidate for such a construction would be the scheme of Katz-Vaikuntanathan [21], which is also one-round and in the (more realistic) common reference string model. Still, the underlying computational load per user would be much higher, as random oracles are "substituted" by smooth projective hash functions. In addition the message complexity from the first round would be much larger, as each user may actually be forced to send a dedicated message to every other user. All in all, we believe it is indeed of interest to try and find different constructions following the main idea presented here.

References

1. Michel Abdalla, Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. (password) authenticated key establishment: From 2-party to group. In Salil P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 499–514. Springer, 2007.
2. Michel Abdalla, Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Password-based Group Key Exchange in a Constant Number of Rounds. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography – PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 427–442. Springer, 2006.
3. Michel Abdalla and David Pointcheval. A scalable password-based group key exchange protocol in the standard model. In *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'06)*, pages 332–347, 2006.

4. Yair Amir, Cristina Nita-Rotaru, John L. Schultz, Jonathan Robert Stanton, Yongdae Kim, and Gene Tsudik. Exploring robustness in group key agreement. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS 2001), Phoenix, Arizona, USA, April 16-19, 2001*, pages 399–408. IEEE Computer Society, 2001.
5. Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana K. Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *2003 IEEE Symposium on Security and Privacy (S&P)*, pages 180–196, 2003.
6. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In *Advances in Cryptology – EUROCRYPT 2000*, pages 139–155, 2000.
7. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.
8. John Black and Phillip Rogaway. Ciphers with arbitrary finite domains. In Bart Preneel, editor, *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, volume 2271 of *Lecture Notes in Computer Science*, pages 114–130. Springer, 2002.
9. Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. Secure Group Key Establishment Revisited. *International Journal of Information Security*, 6(4):243–254, 2007.
10. Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using diffie-hellman. In *Advances in Cryptology - EUROCRYPT 2000*, pages 156–171, 2000.
11. Christian Cachin and Reto Strobl. Asynchronous group key exchange with failures. In Soma Chaudhuri and Shay Kutten, editors, *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 357–366. ACM, 2004.
12. Claude Castelluccia, Stanislaw Jarecki, and Gene Tsudik. Secret handshakes from ca-oblivious encryption. In *10th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 293–307, 2004.
13. J.H. Conway and R. Guy. *The Book of Numbers*. Copernicus Series. Springer New York, 1998.
14. Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2008.
15. Ran Gelles, Rafail Ostrovsky, and Kina Winoto. Multiparty proximity testing with dishonest majority from equality testing. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 537–548. Springer, 2012.
16. Stanislaw Jarecki, Jihye Kim, and Gene Tsudik. Authentication for paranoids: Multi-party secret handshakes. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, volume 3989 of *Lecture Notes in Computer Science*, pages 325–339, 2006.
17. Stanislaw Jarecki, Jihye Kim, and Gene Tsudik. Robust group key agreement using short broadcasts. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 411–420. ACM, 2007.
18. Stanislaw Jarecki, Jihye Kim, and Gene Tsudik. Beyond secret handshakes: Affiliation-hiding authenticated key exchange. In Tal Malkin, editor, *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8-11, 2008. Proceedings*, volume 4964 of *Lecture Notes in Computer Science*, pages 352–369. Springer, 2008.
19. Stanislaw Jarecki, Jihye Kim, and Gene Tsudik. Flexible robust group key agreement. *IEEE Trans. Parallel Distrib. Syst.*, 22(5):879–886, 2011.
20. Stanislaw Jarecki and Xiaomin Liu. Unlinkable secret handshakes and key-private group key management schemes. In *Applied Cryptography and Network Security (ACNS)*, pages 270–287, 2007.
21. Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. *J. Cryptology*, 26(4):714–743, 2013.
22. Jonathan Katz and Moti Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In *Fast Software Encryption (FSE)*, pages 284–299, 2000.

23. Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. In *Advances in Cryptology — CRYPTO'03*, pages 110–125, 2003.
24. Jihye Kim and Gene Tsudik. Survival in the wild: Robust group key agreement in wide-area networks. In Pil Joong Lee and Jung Hee Cheon, editors, *Information Security and Cryptology - ICISC 2008, 11th International Conference, Seoul, Korea, December 3-5, 2008, Revised Selected Papers*, volume 5461 of *Lecture Notes in Computer Science*, pages 66–83. Springer, 2008.
25. Mark Manulis, Bertram Poettering, and Gene Tsudik. Affiliation-hiding key exchange with untrusted group authorities. In *Applied Cryptography and Network Security (ACNS)*, pages 402–419, 2010.
26. Gene Tsudik and Shouhuai Xu. A flexible framework for secret handshakes. In George Danezis and Philippe Golle, editors, *Privacy Enhancing Technologies, 6th International Workshop, PET 2006, Cambridge, UK, June 28-30, 2006, Revised Selected Papers*, volume 4258 of *Lecture Notes in Computer Science*, pages 295–315. Springer, 2006.