An extended abstract [DRS24] of this article appears in the proceedings of ASIACRYPT 2024. This is the full version.

Tightly-Secure Group Key Exchange with Perfect Forward Secrecy

Emanuele Di Giandomenico[1], Doreen Riepel[2], and Sven Schäge[1]

[1] Eindhoven University of Technology, Eindhoven, Netherlands
{e.di.giandomenico,s.schage}@tue.nl
[2] UC San Diego, USA
doreen.riepel@gmail.com

Abstract. In this work, we present a new paradigm for constructing Group Authenticated Key Exchange (GAKE). This result is the first tightly secure GAKE scheme in a strong security model that allows maximum exposure attacks (MEX) where the attacker is allowed to either reveal the secret session state or the long-term secret of all communication partners. Moreover, our protocol features the strong and realistic notion of (full) perfect forward secrecy (PFS), that allows the attacker to actively modify messages before corrupting parties. We obtain our results via a series of tightly secure transformations. Our first transformation is from weakly secure KEMs to unilateral authenticated key exchange (UAKE) with weak forward secrecy (WFS). Next, we show how to turn this into an UAKE with PFS in the random oracle model. Finally, and as one of our major novel conceptual contributions, we describe how to build GAKE protocols from UAKE protocols, also in the random oracle model. We apply our transformations to obtain two practical GAKE protocols with tight security. The first is based on the DDH assumption and features low message complexity. Our second result is based on the LWE assumption. In this way, we obtain the first GAKE protocol from a post-quantum assumption that is tightly secure in a strong model of security allowing MEX attacks.

1 Introduction

Group Authenticated Key Exchange (GAKE) is the generalization of two-party key exchange to the group setting. It allows N group members to compute a common symmetric session key over an insecure network. This key can then be used to exchange messages among the group members that are protected via efficient symmetric cryptography. As such GAKE protocols form an important building block in any form of group-based communication.

Proving security of GAKE protocols is in general much more challenging than for classical AKE protocols. In many AKE security proofs the two parties participating in the protocol can simply be guessed upfront resulting in a polynomial security loss $\binom{N}{2}$. For GAKE protocols this strategy quickly becomes infeasible with growing group size t since there are $\binom{N}{t}$ possible groups that could now run the GAKE protocol. For superlogarithmical t this number already grows superpolynomial and guessing the group upfront becomes inefficient. Moreover, each of the existing GAKE protocols have one of the following downsides.

Vulnerabilty Against Quantum Attacks. The vast majority of GAKE protocols rely on classical security assumptions that are related to the discrete logarithm assumption. The underlying problems are known to be solvable efficiently by quantum computers [Sho94]. For long-term security a shift towards post-quantum-based security assumptions is necessary. However, relying on post-quantum assumptions often introduces new challenges like non-perfect correctness in lattice cryptography. Thus PQ-based security assumptions cannot be used as a drop-in to classical protocols and new techniques are necessary.

Realistic Security Models. Most GAKE protocols consider relatively weak security definitions that only consider attackers that may corrupt the long-term keys of parties while disallowing that the (ephemeral) state material stored by parties between moves will ever be revealed. So, in case an attacker manages to obtain the state information of a *single* group member all security guarantees may be lost. A stronger and much more realistic notion of security considers so-called maximum exposure attacks (MEX) that allow the attacker to also reveal the secret states of the group members while carefully excluding trivial attacks [LLM07]. These models are considered standard in the case of two-party key exchange.

Non-Tight Security Proofs. A tight security proof allows for highly efficient and theoretically-sound instantiations of the system parameters. In particular, the proofs – and thus the system parameters – are independent of the number of parties, sessions, or the number of attacker queries. Providing schemes with tight security proofs for asymmetric cryptography is challenging [BJLS16], in particular for key exchange [CC17]. Only recently tightly secure AKE protocols have been proposed for strong security in the two-party case [JKRS21, HJK+21, PWZ23a]. The only tightly secure GAKE protocol [PQR22] relies on signatures (which are generally less efficient in the PQ-setting) and also does not protect against MEX attacks.

1.1 Contribution

In this work, we tackle these challenges and present the first, tightly secure GAKE protocol that is secure under post-quantum assumptions under a strong, realistic notion of security. To this end, we develop a new paradigm for constructing GAKE schemes. To explain it intuitively, consider the well-known ring-based Burmester-Desmedt (BD) protocol [BD95]. In this protocol, each party P_i first sends $k_i = g^{x_i}$ for some randomly drawn ephemeral secret x_i. In the next round, each party sends $K_i = (k_{i+1}/k_{i-1})^{x_i}$ where all indices are taken mod t for group size t. The final group key is produced as

$$K = k_{i-1}^{tx_i} K_i^{t-1} K_{i+1}^{t-2} \dots K_{i-2} = g^{x_1x_2+x_2x_3+\dots+x_tx_1} \ .$$

This protocol is only passively secure but serves as a guiding principle in many constructions. Using digital signatures over all messages sent, this protocol can be made actively secure (though it remains highly vulnerable to state-reveal attacks). While it is elegant, we believe that it rather disguises the core principles that make it work. We therefore present a more conceptual perspective to the design of GAKE protocols that to the best of our knowledge is novel. This allows us to identify the parts that can be improved considerably.

Novel Conceptual Perspective on GAKE. Assume we have t parties $P_1, \dots, P_t$ organized in a ring. Essentially we view a GAKE protocol as consisting of two phases. In the first phase, adjacent parties compute a common session key via a two-party protocol. To make this secure against active attacks, the neighboring parties will at some point (implicitly or explicitly) authenticate each other. In particular, for each i, P_i authenticates P_{i-1} and P_{i+1}. In the basic BD protocol (which is only passively secure) this step simply consists of sending k_i. Actively secure protocols that rely on the BD protocol, typical add authentication via other means like digital signatures. More concretely, each party will also sign each message they send. The shared key with party P_{i+1} can then be computed as $G_{i,i+1} = (k_{i+1})^{x_i}$. Likewise, the shared key computed with P_{i-1} can then be computed as $G_{i-1,i} = (k_{i-1})^{x_i}$. The second phase of the protocol consists of distributing the derived key material to the other parties. Now, a key insight is that, in order to not hand these keys over to an impersonating attacker, they are only given to parties that P_i has authenticated before. In particular, $G_{i,i+1}$ is only given to P_{i-1} and $G_{i-1,i}$ is only given to P_{i+1}. In the BD protocol this is done simultaneously via simply publishing $G_{i,i+1}/G_{i-1,i}$. This can be thought as a simple symmetric encryption of $G_{i,i+1}$ (respectively $G_{i-1,i}$) via the key $G_{i-1,i}$ (respectively $G_{i,i+1}$).

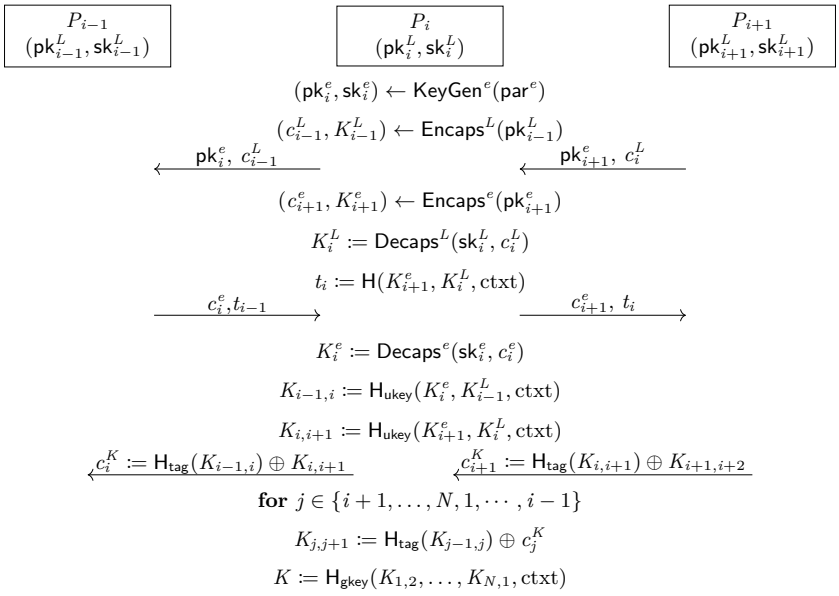

Fig. 1: The idea of the complete construction presented in Section 6, assuming, without loss of generality, that the predecessor and the successor of party i are $i-1, i+1$, respectively, and that the parties verify the tag received. The superscripts e and L stand for "ephemeral" and "long-term" respectively. H_* with different subscripts $*$ are (independent) random oracles.

Now observe that from the knowledge of all the K_j, each party P_i can now easily compute $K = g^{x_1x_2+x_2x_3+\ldots+x_tx_1} = G_{1,2}G_{2,3}\cdots G_{t,1}$. It first computes $G_{i,i+1}$ using k_{i+1} and x_i. Next, it can step-wisely compute the next value $G_{i+1+c,i+2+c}$ from K_{i+1+c} and $G_{i+c,i+1+c}$ for any $c = 0, 1, \ldots, t-1$. The BD protocol essentially computes this process in an algebraically elegant and efficient fashion.

Having this perspective in mind, we make several conceptual changes to the design that enable better efficiency. First, we do not require that P_i authenticates P_{i+1} *and* vice versa. Crucially, we observe that only one direction is enough. This is because the group members are organized in a ring: if each member authenticates its predecessor only, all parties will be authenticated eventually. In general, reciprocal authentication among neighbors seems wasteful. In addition to that, this change will now allow us to avoid using AKE protocols but instead rely on unilaterally authenticated key exchange (UAKE) where only a single party is authenticated. Overall this saves bandwidth and computational complexity. Let us clarify: each party will, as before, compute two shared keys, one with its predecessor and one with its successor. However, only the predecessor will be authenticated. Thus, in the second phase, parties will now only distribute the symmetric key that they share with their successor to their authenticated predecessor (and not vice versa). The second change that we make is that we consider the symmetric encryption scheme used more generally in the second phase. To this end, we use a simple random oracle-based symmetric encryption system, where the sharing of key $K_{i,i+1}$ to P_i now proceeds as $h(K_{i-1,i}) \oplus K_{i,i+1}$. In this way, each party essentially only encrypts to one party – its predecessor. This scheme is very simple and fast and has strong security properties. Unfortunately, currently there is no efficient and tightly secure post-quantum secure digital signature scheme that is suitable to implementing authentication efficiently.[3] To obtain a tightly secure GAKE protocol overall — even in the PQ-setting — we will thus deviate from the use of digital signature schemes and instead rely on authentication via KEMs, similar to previous work on tight AKE in the 2-party setting [JKRS21, PWZ23a, PRZ24]. We provide an illustration of our protocol from

[3] The signature schemes introduced in [PW22, HLWG23] do provide (almost) tight security but are too inefficient for practical applications.

the view of one party in Figure 1. Typically post-quantum signatures are considerably larger in size than KEM ciphertexts. Essentially, the mechanism that we use for authentication will require P_i to send an encapsulated key to its predecessor (encrypted with pk_{i-1}). This key will be decapsulated by P_{i-1} and is then used to derive a MAC key, which in turn is used to provide integrity protection for all the messages sent and received by P_{i-1}. Since P_i knows the encapsulated key as well, it can recompute the MAC. The security properties of the KEM guarantee that the MAC can only be computed correctly by the predecessor if it indeed has the corresponding secret long-term key. In our instantiations, we rely on the recent tightly secure KEM from [PWZ23a] that is secure under lattice assumptions and the DDH based scheme introduced in [JKRS21] that both fulfill the notion of OW-PCVA-CR [PRZ24], a very weak notion of KEM security.

We proceed as follows. First, we present a construction of an UAKE scheme with weak perfect forward secrecy (WFS) that is constructed from a KEM. Next, we present a transformation from a WFS-secure UAKE to an UAKE that provides full PFS in the random oracle model (ROM). Whereas WFS only provides security guarantees against long-term key corruptions in case the attacker has *not* modified the sent messages, full PFS also guarantees security in the presence of active attackers that modify messages. Finally, and as our main contribution, we present a transformation from PFS-secure UAKE to PFS-secure GAKE, again in the ROM. Security holds even under MEX attacks where the attacker may adaptively reveal state information and adaptively corrupt parties. Remarkably, all our transformations preserve the tightness of the security proof so that the final GAKE will tightly reduce to the security of the KEM. When instantiated with the PQ-secure scheme of [PWZ23a] this results in the first tightly secure GAKE scheme under lattice assumptions in a very strong model of security. When instantiated under the DDH assumption, our protocol only requires to send 5 group elements and two bitstrings of length 256 bits per party. In comparison, the tightly secure protocol of [PQR22] requires to send 2 group elements and 4 exponents (when relying on generic group model bounds of Schnorr signatures), or 2 group elements and 6 exponents (when instantiating the signature scheme with [DGJL21]).

1.2 Security Model

To provide a strong notion of security that reflects full PFS and security against state-reveal attacks, we present a new security definition. We remark that providing security notions for GAKE has in the past proven error-prone. This is due to the number of subcases that one has to consider in the proof. In this work, we take a new avenue that simplifies the development of such a definition.

The central idea is to strongly rely on a corresponding security definition for two parties. This definition is now used more generically to develop the GAKE definition. To this end, we take the strong definition of [JKRS21] as a starting point. This definition features an attack table that defines when certain query combinations of the attacker are deemed non-trivial. In our new definition a similar attack table is (almost) generically utilized at the end of our security experiment to evaluate if, for any of the tested sessions, the attacker has performed a trivial attack. However, we need to be careful since the application of the checks in the [JKRS21] table do not only depend on the holder of the tested session itself, but also on its peer.

Our formulation of GAKE security thus essentially re-applies this table to all the peers of the considered tested session that are currently participating in the GAKE run. In this way, we can reduce the problem of analyzing trivial attacks for a session and all its peers to the problem of analyzing trivial attacks for this session and a single peer. We note that the semantics of these tables define when an attack is valid. In this sense they encode properties of non-trivial attacks. By setup all other attacks are deemed trivial. So to make this useful in the group setting we require that for all of the pairs of parties, the attacker hasn't performed a trivial attack.

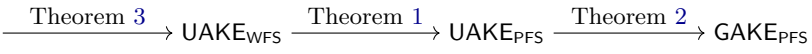

Fig. 2: Logical implication sequence from KEM to GAKE$_{\mathsf{PFS}}$ with intermediate steps.

We note that our GAKE definition holds for any polynomial-sized groups, in particular for groups of size 2. This implies a definition for classical AKE as well. However, conceptually the exposition of our algorithms is structured into rounds, where every party has to apply the same algorithms. This allows us to specify algorithms independent of classical roles like initiator or responder. However, when proving the security of our GAKE from the underlying UAKE we have to relate the UAKE roles of initiator and responder to the behaviour of two adjacent parties in the GAKE using terms like predecessor and successor.

1.3 Related Work

There has been considerably less research activity on GAKE than on classical two-party AKE. A nice overview of the existing notions of group AKE can be found in [PRSS21].

The protocol from [DB05] is similar to BD. It relies on DDH and signatures to achieve PFS. It supports dynamic groups and only requires two rounds. We note that while we require three rounds to get PFS, it was shown in [Kra05] for 2-party AKE that if the underlying protocol is only implicitly authenticated (e.g., via KEMs), then a protocol cannot achieve PFS in two rounds.

The protocol in [ADGK19] can be thought of as a lattice-based variant of the BD protocol that is secure under the Ring-LWE assumption. Correspondingly it is passively secure and needs additional authentication mechanisms for active security. To this end, the authors propose the application of signatures. It is generally unclear how to do this in an efficient and tightly secure manner in the post-quantum setting. The security model of [ADGK19] does not allow the attacker to reveal secret state information.

The work in [PQR22] focuses on tight security. It also takes the BD protocol as a basis and presents a tight proof in a security model that does not allow the attacker to reveal secret state information. The construction applies the efficient Schnorr signature scheme to protect the protocol against active attacks and achieve authentication. We deviate from these two approaches by considering tight security in strong models that allow MEX attacks. Moreover, we use a novel authentication mechanism that relies on KEMs instead of signatures. This allows us to obtain efficient instantiations based on previous works. Our instantiation in the classical setting considers the highly efficient DDH-based scheme introduced in [JKRS21]. In the post-quantum setting we can apply the recent scheme of [PWZ23a] that is based on the LWE assumption. However, from the description of the scheme in [PWZ23a] it is not immediately clear if it can be applied to our transformation when used in the group setting. The problem is that the correctness of the scheme is only shown to hold with probability $(1 - z)$ where $z = \mathsf{negl}(\kappa)$. This can be problematic when bounding the probability that *all* $N = \mathsf{poly}(\kappa)$ KEM applications that are required in a run of the GAKE protocol provide correctness because $(1-\mathsf{negl}(\kappa))^N$ is only overwhelming if z is statistically small. Fortunately, we can show that z is indeed statistically small [PWZ23a].

Recent works on AKE aim at achieving tighter security reductions in the QROM [HKSU20, PWZ23b, PRZ24]. The first AKE protocol proven secure in the QROM [HKSU20] suffers from a square-root security loss in the random oracle model. This was improved in [PWZ23b] that provides a QROM proof with a loss only linear in the number of users. The resulting scheme only provides weak forward secrecy. Very recently, via an additional key confirmation move [PRZ24], this was lifted to a protocol that provides perfect forward secrecy, also with a linear loss in the number of users.

Another interesting work related to ours is the authentication compiler of Katz and Yung that constructs actively secure GAKE from a passively secure one. Essentially the paper proposes to authenticate all messages with digital signatures schemes as in the BD protocol.

However, their analysis does not account for attacks that reveal ephemeral states. Also they do not specifically consider tight reductions. Our result, in contrast, uses authentication based on KEMs that provides efficient instantiations in the DH setting and the PQ-setting. At the same time, our solutions are tightly secure.

In 1999, Mayer and Yung have proposed a construction of group AKE from two-party AKE [MY99]. The model that they use is comparatively weak and does not consider attacks that reveal state information. Also, they rely on key exchange with mutual authentication that – when used in a ring setting – requires each party to be authenticated twice. Our solution based on UAKE, authenticates parties only once and is thus more efficient, while featuring tight security. Similarly, the work presented in [ABGS07] considers a compiler from AKE to GAKE. Again, the security model is weaker than ours and does not allow to reveal state information in the GAKE. As in [MY99] the compiler requires the computationally more complex notion of AKE whereas we solely require UAKE.

UAKE protocols and their security notions were previously studied in [DF17] and [MTC13], where the former proposes a 2-round forward-deniable and forward-secure UAKE from KEMs that is very similar to ours and the latter focuses on universal composability (UC) security. Further, [IY22] studies anonymity of UAKE. The main focus of these works is to study UAKE protocols themselves, whereas we use UAKE as a building block for GAKE. Hence, our security notion is tailored to be as weak as possible to enable our transformation, which makes it presumably weaker than (or incomparable to) the ones given in these works.

We mention that our notion of security covers key compromise impersonation (KCI) security for GAKE as introduced by [GBG09]. Whereas [GBG09] can be thought of as an analogue of the security notion introduced in [Kra05], our notion rather generalizes the stronger notion of [LLM07].

Finally, we remark that GAKE is generally related to Group Continous Key Agreement, a notion that has gained much interest [ACJM20, CCG+18, ACDT20, KPPW+21] in the last years. More formally, the authors of [BDG+22] provide initial results showing that weakly-secure variants of these primitives are indeed equivalent. We believe that this relationship will become much clearer in the future where we expect GAKE to be an essential primitive used in the setup phase of CGKA protocols to establish key material for the first time. It is thus very helpful that our protocol provides security even in case the attacker obtains secret state information. This seems helpful in CGKA constructions to achieve the intricate notions of post-compromise security that CGKA protocols try to guarantee in a provably secure way.

2 Preliminaries

For a positive integer N, let $[N] \coloneqq \{1, \ldots, N\}$. For a set S, let $|S|$ be the cardinality of S; moreover, $s \leftarrow S$ denotes that s is sampled uniformly at random from S. We use the abbreviation $[\![B]\!]$ to represent the bit set to 1 when the boolean statement B is true, and 0 otherwise.

By $y \leftarrow \mathcal{A}(x)$, we denote that on input $x \in X$, the probabilistic algorithm $\mathcal{A}$ returns $y \in Y$. Otherwise, by $y \coloneqq \mathcal{A}(x)$, we denote that on input x, the deterministic algorithm $\mathcal{A}$ returns y. By $\mathcal{A}^{\mathrm{O}}$, we denote that the algorithm $\mathcal{A}$ has access to oracle O. We say that probabilistic algorithm $\mathcal{A}$ has min-entropy μ if for all outputs $y' \in Y$ we have $\Pr[y = y' \colon y \leftarrow \mathcal{A}(x)] \leq 2^{-\mu}$.

Following [Sho04], we use code-based games. An adversary is a probabilistic polynomial time algorithm. Let G be a game, for an adversary $\mathcal{A}$, $\mathsf{G}^{\mathcal{A}} \Rightarrow 1$ denotes that the output of game G running with adversary $\mathcal{A}$ is 1. All the games that will be introduced later have two fixed oracles, Initialize and Finalize, which can be queried at most once, as the first and last query respectively. The output of the game is the output of the Finalize query.

3 Unilateral Authenticated Key Exchange

We will first define unilateral authenticated key exchange (UAKE), which is a two-party protocol where only one party authenticates to the other. We will only focus on two-message protocols (but note that the syntax can be extended trivially).

Syntax. A two-message unilateral authenticated key exchange $\mathsf{UAKE} := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Beg}, \mathsf{Der}_R, \mathsf{Der}_B)$ consist of five polynomial-time algorithms:

- $\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$: The probabilistic setup algorithm Setup takes as input the security parameter κ in unary and returns global system parameters par that implicitly define message space $\mathcal{T}$, the public key space $\mathcal{PK}$, the secret key space $\mathcal{SK}$ and the key space $\mathcal{K}$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{par})$: The probabilistic key generation algorithm KeyGen takes as input the parameters par and returns a public key $\mathsf{pk} \in \mathcal{PK}$ and a secret key $\mathsf{sk} \in \mathcal{SK}$.
- $(M_1, \mathsf{st}) \leftarrow \mathsf{Beg}(\mathsf{pk})$: The probabilistic initial algorithm Beg takes as input a public key pk and returns a message $M_1 \in \mathcal{T}$ and a state st.
- $(M_2, K) \leftarrow \mathsf{Der}_R(\mathsf{sk}, M_1)$: The probabilistic derivation for the responder algorithm Der_R takes as input a secret key sk and a message M_1 and returns a message $M_2 \in \mathcal{T}$ and a key $K \in \mathcal{K}$.
- $K := \mathsf{Der}_B(\mathsf{pk}, M_2, \mathsf{st})$: The deterministic derivation for the initiator algorithm Der_B takes as input a secret key pk, a message M_2 and a state st and returns a key $K \in \mathcal{K}$.

Note that only the party I save a state information, even if only the party R has long-term keys. Then, R can derive immediately the session key K after receiving the message of I, cf. also Figure 3 below.

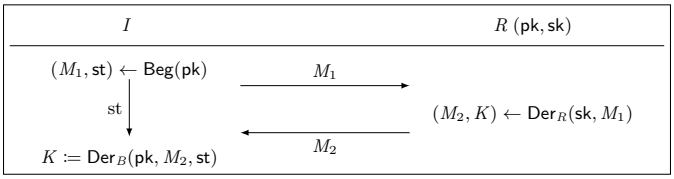

Fig. 3: Syntax of a two-message unilateral key exchange protocol.

Definition 1 (Correctness of UAKE). *We say that* UAKE *is* ρ*-correct, if for any* $\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$ *we have:*

$$\Pr\left[K = K' \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{par}), \\ (M_1, \mathsf{st}) \leftarrow \mathsf{Beg}(\mathsf{pk}), \\ (M_2, K) \leftarrow \mathsf{Der}_R(\mathsf{sk}, M_1), \\ K' := \mathsf{Der}_B(\mathsf{pk}, M_2, \mathsf{st}) \end{array}\right] \geq \rho \ ,$$

and the probability is over the random coins consumed by the algorithms of UAKE*.*

Now we introduce the definition of min-entropy for UAKE. This is extremely useful for the theorems we will introduce.

Definition 2 (Min-Entropy of UAKE). *We say that* UAKE *has min-entropy* μ *if:*

- *It has key min-entropy* $\mu' \geq \mu$*: for any* $\mathsf{pk}' \in \mathcal{PK}$ *we have* $\Pr[\mathsf{pk} = \mathsf{pk}' \colon (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{par})] \leq 2^{-\mu'}$ *for some* par*.*
- *It has min-entropy* $\mu'' \geq \mu$ *of* Beg*: for any* $M_1' \in \mathcal{T}$ *we have* $\Pr[M_1 = M_1' \colon (M_1, \mathsf{st}) \leftarrow \mathsf{Beg}(\mathsf{pk})] \leq 2^{-\mu''}$ *for some* $\mathsf{pk} \in \mathcal{PK}$*.*
- *It has min-entropy* $\mu''' \geq \mu$ *of* Der_R*: for any* $M_2' \in \mathcal{T}$ *we have* $\Pr[M_2 = M_2' \colon (M_2, K) \leftarrow \mathsf{Der}_R(\mathsf{sk}, M_1)] \leq 2^{-\mu'''}$ *for some* $\mathsf{sk} \in \mathcal{SK}$ *and* $M_1 \in \mathcal{T}$*.*

3.1 Security for UAKE

We consider N parties $P_1, \ldots, P_N$ (for an easier notation sometimes we use n to refer to P_n) with long-term key pairs $(\mathsf{pk}_n, \mathsf{sk}_n)$, $n \in [N]$. An interaction between two parties is called session, and to each session are associated an identification number sID and variables defined on sID.

- ini[sID] $\in [N]$ denotes the initiator of the session.
- res[sID] $\in [N]$ denotes the responder of the session.
- type[sID] $\in \{$"In", "Re"$\}$ denotes if the initiator or the responder computes the session key.
- I[sID] denotes the message sent by the initiator.
- R[sID] denotes the message sent by the responder.
- state[sID] denotes the state.
- sKey[sID] denotes the session key. In case a party does not accept, this variable will be set to **rej**.

Moreover, we use the following boolean values to store which queries the adversary made.

- corr[n] denotes if long-term secret key of party P_n has been given to the adversary.
- revState[sID] denotes if the state has been given to the adversary.
- peerCorr[sID] denotes if the peer of the session is corrupted and its long-term key has already been given to the adversary at the time the session key is derived.

Let us now define what it means for two sessions to be (partially) matching. As in two-party key exchange, the notion of partially matching sessions is used to define if two parties have communicated with each other and so revealing session secrets of the first will also reveal secrets of the second. Partially matching sessions take into account that parties have to accept at distinct points in time: the party responsible for sending the last message will accept independent of whether the attacker modifies the last message on transit or not. We will later use our general methodology and derive a definition of partially matching for GAKE protocols that is intuitively based on a repeated application of the two-party notion. This simplifies the exposition greatly.

Definition 3 (Partially matching session for UAKE). *The session sID is partially matched with session sID* if the following conditions are satisfied.*

1. *The sessions have the same initiator and responder,* $(ini[sID], res[sID]) = (ini[sID^*], res[sID^*])$.
2. *The messages of the initiator are identical,* $I[sID] = I[sID^*]$.
3. *The types of the sessions are distinct,* $type[sID] \neq type[sID^*]$.
4. *The type of sID is "In",* $type[sID] =$ *"In".*

Definition 4 (Matching session for UAKE). *Two sessions sID and sID* are matching if the following conditions are satisfied.*

1. *The sessions have the same initiator and responder,* $(ini[sID], res[sID]) = (ini[sID^*], res[sID^*])$.
2. *The messages of the initiator are identical,* $I[sID] = I[sID^*]$.
3. *The messages of the responder are identical,* $R[sID] = R[sID^*]$.
4. *The type of the sessions are distinct,* $type[sID] \neq type[sID^*]$.

OW-Security. We define security in a one-way game, where the adversary has to compute the session key of a target session of its choice. The full game is given in Figure 4. The adversary can create parties using the KeyGeneration oracle. It can send and relay messages between the parties using oracles $\mathsf{Beg}, \mathrm{S}_{\mathsf{Der}_R}, \mathrm{S}_{\mathsf{Der}_B}$. It can reveal the state of sessions and corrupt parties via Rev-State and Corrupt, respectively. Further, we allow the adversary to check for session keys using oracle Check. If the session is fresh when the oracle is queried and the key is correct, then we set a flag attFound which will also be the final output of the game, i.e., the adversary wins whenever this flag is set.

In order to rule out trivial attacks, we use attack tables as introduced in [JKRS21] to describe which queries the adversary is allowed to make for an attack. We give two tables, one

Fig. 4: Games OW-UAKE-G$_X$ for a two-message UAKE. Adversary $\mathcal{A}$ has access to oracles O := {INITIALIZE, KEYGENERATION, S_{Beg}, S_{Der_R}, S_{Der_B}, REV-STATE, CORRUPT, CHECK, FINALIZE}. Helper procedure UVALID$_X$ verifies the validity of attacks against the UAKE protocol. Attack tables are given in Table 1.

capturing perfect forward secrecy (cf. Table 1a) and the other one capturing weak forward secrecy (cf. Table 1b) for UAKE protocols.

Each table is parameterized by an initiator i^* and responder r^* session. Note that only the responder has a public key, hence we only need to consider corruptions of that party. The initiator on the other hand holds a state, however, we do not allow the adversary to reveal the state for any session it wants to attack. Even though this could be achieved for some cases, we will see that it is not necessary to allow this attack for our following transformations. Similarly, we also omit a session key reveal oracle. This way, we relax the security definition as much as possible to allow for most efficient instantiations, while still achieving the strongest target notion for the final group AKE protocol.

We explain Table 1a in more detail, the other table can be read in a similar way. Line (0) captures that if a protocol has not sufficient entropy, then the protocol should not be

$\mathcal{A}$ gets (i^*, r^*)	corr$[r^*]$	peerCorr[sID*]	type[sID*]	revState[sID*]	$\exists$sID $\in \mathfrak{M}$(sID*) : revState[sID]	$\lvert\mathfrak{M}(\text{sID}^*)\rvert$	$\exists$sID $\in \mathfrak{P}$(sID*) : revState[sID]	$\lvert\mathfrak{P}(\text{sID}^*)\rvert$
(0) **multiple partially matching sessions**	-	-	-	-	-	-	-	>1
(1) **(–, long-term)**	-	-	-	**F**	**F**	1	-	-
(2) **(–, long-term)**	-	-	"Re"	**F**	n/a	0	**F**	1
(3) **(–, long-term)**	-	**F**	"In"	**F**	n/a	0	n/a	0

(a) Attack table for perfect forward secrecy

$\mathcal{A}$ gets (i^*, r^*)	corr$[r^*]$	peerCorr[sID*]	type[sID*]	revState[sID*]	$\exists$sID $\in \mathfrak{M}$(sID*) : revState[sID]	$\lvert\mathfrak{M}(\text{sID}^*)\rvert$	$\exists$sID $\in \mathfrak{P}$(sID*) : revState[sID]	$\lvert\mathfrak{P}(\text{sID}^*)\rvert$
(0) **multiple partially matching sessions**	-	-	-	-	-	-	-	>1
(1) **(–, long-term)**	-	-	-	**F**	**F**	1	-	-
(2) **(–, long-term)**	-	-	"Re"	**F**	n/a	0	**F**	1
(3) **(–, long-term)**	**F**	-	"In"	**F**	n/a	0	n/a	0

(b) Attack table for weak forward secrecy

Table 1: Attack tables for UAKE protocols. The difference lies in the time when the peer can be corrupted in an actively attacked session. An attack is regarded as an AND conjunction of variables with specified values as shown in the each line, where "–" means that this variable can take arbitrary value, "**F**" means "false", "n/a" indicates that there is no state which can be revealed as no (partially) matching session exists.

considered secure. If this is the case, it should be possible for an adversary to create a session that has multiple (partially) matching sessions, so whenever this happens, we consider it a valid attack which lets the adversary win directly. Line (1) is for sessions that have a matching session. These can be of type "In" or "Re" and in this case we allow the adversary to reveal the responder's long-term key, but (as explained above) we never allow to reveal the initiator's state. Line (2) captures partially matching sessions which are always of type "Re". For those, we also allow the responder to be corrupted. Line (3) captures sessions that do not have any (partially) matching partner session. Since Table 1a looks at perfect forward secrecy, we allow to reveal the responder's secret key *after* the session key has been computed, which is captured by variable peerCorr. We only consider sessions of type "In" since the initiator has no long-term secrets and sessions of type "Re" can never be secure when being actively attacked.

For completeness, we give the table for weak forward security of UAKE protocols in Table 1b. It is very similar to Table 1a, the only difference is that the peer cannot be corrupted at all if the adversary was active.

Definition 5. *We define the game* $\mathsf{OW\text{-}UAKE\text{-}G_X}$ *for* $\mathsf{X} \in \{\mathsf{PFS}, \mathsf{WFS}\}$ *as in Figure 4. The advantage of an adversary* $\mathcal{A}$ *against* UAKE *in this game is defined as*

$$\mathrm{Adv}^{\mathsf{X}}_{\mathsf{UAKE}}(\mathcal{A}) \coloneqq \Pr[\mathsf{OW\text{-}UAKE\text{-}G}^{\mathcal{A}}_{\mathsf{X}} \Rightarrow 1] \ .$$

3.2 From WFS to PFS Secure UAKE

We construct a UAKE protocol $\mathsf{UAKE}_{\mathsf{PFS}}$ with perfect forward secrecy from a UAKE protocol $\mathsf{UAKE}_{\mathsf{WFS}}$ with weak forward secrecy and two hash functions H, $\mathsf{H}_{\mathsf{ukey}}$. The idea is that the session key of $\mathsf{UAKE}_{\mathsf{WFS}}$ will be used twice: to derive the $\mathsf{UAKE}_{\mathsf{PFS}}$ session key and to compute a key confirmation hash which is sent together with M_2. An illustration of the protocol is given in Figure 5.

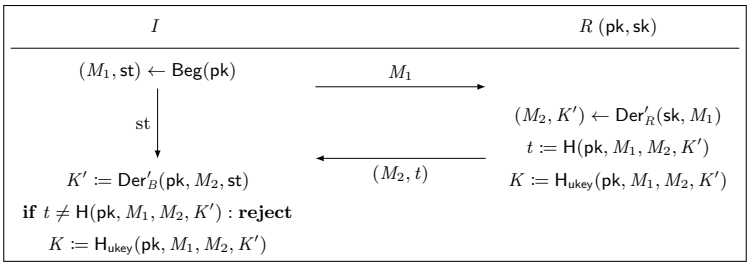

Fig. 5: Protocol $\mathsf{UAKE}_{\mathsf{PFS}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Beg}, \mathsf{Der}'_R, \mathsf{Der}'_B)$ constructed from $\mathsf{UAKE}_{\mathsf{WFS}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Beg}, \mathsf{Der}_R, \mathsf{Der}_B)$ and random oracles $\mathsf{H}, \mathsf{H}_{\mathsf{ukey}}$.

Observe that the construction does not introduce any new primitives at all. Hence, correctness is preserved from the underlying $\mathsf{UAKE}_{\mathsf{WFS}}$.

Lemma 1. *If* $\mathsf{UAKE}_{\mathsf{WFS}}$ *has correctness* $\rho = 1 - 1/2^v$ *for some* $v \in \Omega(\kappa)$ *then* $\mathsf{UAKE}_{\mathsf{PFS}}$ *has overwhelming correctness at least* ρ*.*

Theorem 1 ($\mathsf{UAKE}_{\mathsf{WFS}}$ **to** $\mathsf{UAKE}_{\mathsf{PFS}}$)**.** *Let* $\mathsf{UAKE}_{\mathsf{WFS}}$ *be* $(1 - 1/2^v)$*-correct and with min-entropy* μ*. Let* ζ *be the lower bound for the dimensions of the tag space and key space. For any adversary* $\mathcal{A}$ *against* $\mathsf{OW\text{-}UAKE\text{-}G}_{\mathsf{PFS}}$ *with protocol* $\mathsf{UAKE}_{\mathsf{PFS}}$*, there exists an adversary* $\mathcal{B}$ *against* $\mathsf{OW\text{-}UAKE\text{-}G}_{\mathsf{WFS}}$ *with protocol* $\mathsf{UAKE}_{\mathsf{WFS}}$ *such that*

$$\mathrm{Adv}^{\mathsf{PFS}}_{\mathsf{UAKE}_{\mathsf{PFS}}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{WFS}}_{\mathsf{UAKE}_{\mathsf{WFS}}}(\mathcal{B}) + \frac{N^2 + S^2 + Sq_{RO}}{2^\mu} + \frac{S + q_{RO}^2 + q_{Ch}}{2^\zeta} + \frac{2S}{2^v} \ ,$$

where N *is the number of queries that* $\mathcal{A}$ *and* $\mathcal{B}$ *make to the key generation oracle,* S *is the number of sessions that* $\mathcal{A}$ *and* $\mathcal{B}$ *create, and* q_{RO} *and* q_{Ch} *are the number of random oracle and check queries that* $\mathcal{A}$ *makes. Further, the running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

We give the full proof in Appendix A and we want to give a brief intuition here. It is indeed very similar to the proof for AKE in [PRZ24], adapted to the UAKE case. For this, note that the only difference between the weak and perfect forward security is that in the latter the adversary $\mathcal{A}$ is allowed to query corrupt after the session is completed even if there is no matching session. We will show that due to the key confirmation tag, $\mathcal{A}$ can never complete a session for which the peer was not corrupted. More specifically, $\mathcal{A}$ has to forge a valid tag t. For this, it has to compute the underlying UAKE key and query it to the random oracle. Hence, we can construct a reduction which extracts the key and wins game $\mathsf{UAKE}_{\mathsf{WFS}}$.

4 Group Authenticated Key Exchange

We define group authenticated key exchange (GAKE), which is an N-party protocol, with $N > 2$, where all parties authenticate to each other. We consider three-round broadcast protocols where each round corresponds to a message broadcast. The execution is exemplified in Figure 6.

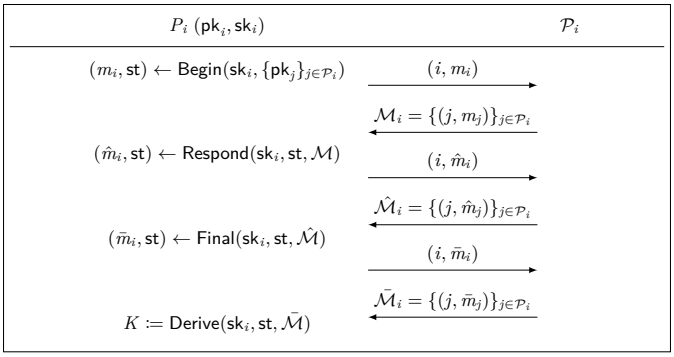

Fig. 6: Execution of the GAKE algorithms in a protocol run.

We indicate with $\mathbf{P} = \{P_1, \ldots, P_N\}$ the set of all potential members. Before the first run of the protocol, each party $P_n \in \mathbf{P}$ runs the algorithm KeyGen to get their own long-term public and secret keys $(\mathsf{pk}_n, \mathsf{sk}_n)$.

Our GAKE protocol allows all parties in a group $\mathbf{P}' \subseteq \mathbf{P}$ to establish a common secret group key. For a party P_n, we define $\mathcal{P}_n \coloneqq \mathbf{P}' \setminus \{P_n\}$ the set of the peers from the point of view of P_n. The following provides a detailed explanation of how our GAKE protocol works and offers proper syntax.

Syntax. A group authenticated key exchange protocol $\mathsf{GAKE} \coloneqq (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Begin}, \mathsf{Respond}, \mathsf{Final}, \mathsf{Derive})$ consists of six polynomial-time algorithms:

- $\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$: The probabilistic setup algorithm Setup takes as input the security parameter κ in unary and returns global system parameters par that implicitly define message space $\mathcal{T}$, the public key space $\mathcal{PK}$, the secret key space $\mathcal{SK}$ and the key space $\mathcal{K}$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{par})$: The probabilistic key generation algorithm KeyGen takes as input the parameters par and returns a public key $\mathsf{pk} \in \mathcal{PK}$ and a secret key $\mathsf{sk} \in \mathcal{SK}$.
- $(m, \mathsf{st}) \leftarrow \mathsf{Begin}(\mathsf{sk}, \{\mathsf{pk}_j\}_{j\in\mathcal{P}})$: The probabilistic first round algorithm Begin takes as input a secret key sk and a set of public keys $\{\mathsf{pk}_j\}_{j\in\mathcal{P}}$ of the peers $\mathcal{P} \subset \mathbb{N}$ and returns a message $m \in \mathcal{T}$ and a state st.
- $(\hat{m}, \mathsf{st}) \leftarrow \mathsf{Respond}(\mathsf{sk}, \mathsf{st}, \mathcal{M})$: The probabilistic second round algorithm $\mathsf{Respond}$ takes a secret key sk, a state st, and a set $\mathcal{M}$ of extended messages $\mathcal{M} = \{(i, m)\}$. Each extended message is a pair consisting of an index $i \in \mathbb{N}$ and a message $m \in \mathcal{T}$. The algorithm returns a message $\hat{m} \in \mathcal{T}$ and an updated state st.
- $(\bar{m}, \mathsf{st}) \leftarrow \mathsf{Final}(\mathsf{sk}, \mathsf{st}, \hat{\mathcal{M}})$: The probabilistic third round algorithm Final takes a secret key sk, a state st, and a set of extended messages $\hat{\mathcal{M}} = \{(i, \hat{m})\}$ with index $i \in \mathbb{N}$ and message $\hat{m} \in \mathcal{T}$, and returns a message $\bar{m} \in \mathcal{T}$ and an updated state st.
- $K \coloneqq \mathsf{Derive}(\mathsf{sk}, \mathsf{st}, \bar{\mathcal{M}})$: The deterministic derivation algorithm Derive takes a secret key sk, a state st and a set of extended messages $\bar{\mathcal{M}} = \{(i, \bar{m})\}$ with index $i \in \mathbb{N}$ and $\bar{m} \in \mathcal{T}$, and returns a group key $K \in \mathcal{K}$.

Definition 6 (Correctness GAKE). *Given N parties, we say that* GAKE *is ρ-correct, if for any* $\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$ *we have:*

$$\Pr\left[K_1 = \cdots = K_N \;\middle|\; \begin{array}{l} \forall i \in [N] : (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{par}), \\ (m_i, \mathsf{st}_i) \leftarrow \mathsf{Begin}(\mathsf{sk}_i, \{\mathsf{pk}_j\}_{j\in[N]\setminus\{i\}}), \\ (\hat{m}_i, \mathsf{st}_i) \leftarrow \mathsf{Respond}(\mathsf{sk}_i, \mathsf{st}_i, \mathcal{M}_i), \\ (\bar{m}_i, \mathsf{st}_i) \leftarrow \mathsf{Final}(\mathsf{sk}_i, \mathsf{st}_i, \hat{\mathcal{M}}_i), \\ K_i \coloneqq \mathsf{Derive}(\mathsf{sk}_i, \mathsf{st}_i, \bar{\mathcal{M}}_i) \end{array}\right] \geq \rho \ ,$$

where $\mathcal{M}_i = \{(j, m_j)\}_{j\in[N]\setminus\{i\}}$, $\hat{\mathcal{M}}_i = \{(j, \hat{m}_j)\}_{j\in[N]\setminus\{i\}}$, *and* $\bar{\mathcal{M}}_i = \{(j, \bar{m}_j)\}_{j\in[N]\setminus\{i\}}$ *and the probability is over the random coins consumed by the algorithms of* GAKE.

To each new run of the group key protocol we assign a unique identification number sID and variables which are defined relative to sID. We call such a run a session.

- holder[sID] $\in [N]$ denotes the holder of the session.
- peers[sID] $\subseteq [N]$ denotes the peers of the session.
- I[sID], R[sID], F[sID] denotes the extended message sent by the holder in the first, second and third round respectively.
- $\mathcal{M}$[sID], $\hat{\mathcal{M}}$[sID], $\bar{\mathcal{M}}$[sID] denotes the extended messages received by the holder in the first, second and third round respectively.
- state[sID] denotes the state.
- gKey[sID] denotes the group key.
- stage[sID] $\in \{2, 3, 4, 5\}$ is used to model that the algorithms of each session are executed in a specific order.

Moreover, we use the following boolean values to store which queries the adversary made.

- corr[n] denotes if long-term secret key of party P_n has been given to the adversary.
- revealed[sID] denotes if the group key has been given to the adversary.
- revState[sID] denotes if the state has been given to the adversary.
- peersCorr[sID] denotes if one of the peers is corrupted and its long-term key has already been given to the adversary at the time the group key is derived.

Let us now define what it means for two sessions to be partially matching for GAKE protocols. This follows the same motivation as in the definition for the two-party case.

Definition 7 (Partially matching session). *Two sessions sID and sID* are partially matching if the following conditions are satisfied.*

1. *The sessions have distinct holders, holder[sID] $\neq$ holder[sID*].*
2. *The extended messages in the first round are identical, $I[sID] \cup \mathcal{M}[sID] = I[sID^*] \cup \mathcal{M}[sID^*]$.*
3. *The extended messages in the second round are identical, $R[sID] \cup \hat{\mathcal{M}}[sID] = R[sID^*] \cup \hat{\mathcal{M}}[sID^*]$.*

Definition 8 (Matching session). *Two sessions sID and sID* are matching if they are partially matching and additionally, we have:*

4. *The extended messages in the third round are identical, $F[sID] \cup \bar{\mathcal{M}}[sID] = F[sID^*] \cup \bar{\mathcal{M}}[sID^*]$.*

Security Notion. We give the full description of the security game in Figure 7. In contrast to the UAKE game, this game models key indistinguishability. The interfaces are however very similar. We allow the adversary to create parties via KeyGeneration. It can create groups and send messages to its members via oracles Session_B, Session_R, Session_F, Der. Here, we not only allow to reveal the state and long-term keys of parties, but also session keys. Security is captured by the Test oracle which can be queried multiple times. All queries are answered with the same bit b.

Similar to UAKE, we use an attack table to describe valid attacks. Here, we only define the stronger notion of perfect forward secrecy in Table 2 since this is our target notion. One can define weak forward secrecy by modifying the table similar to the UAKE notions. Intuitively, we aim for the strongest notion possible where the adversary is allowed to query *either* the long-term key *or* the secret state of any party in the group, even if the corresponding session of any group member will later be queried to Test.

We now describe Table 2 in more detail. As for UAKE, we let the adversary win directly if the protocol does not have sufficient entropy. Further, we iterate over all peers of a session to detect trivial attacks. More specifically, we look at the holder of the session and then at each group member individually. Depending on whether this group member has the same view as the holder, which we determine by the checking whether they have a (partially) matching session, we allow the adversary to reveal long-term keys or states.

GAMES GAKE-$\mathsf{G}_{\mathsf{PFS},b}$

INITIALIZE
00 $\text{cnt}_1 \coloneqq 0$ //session counter
01 $N \coloneqq 0$
02 $\mathcal{S} \coloneqq \varnothing$ //set of test sessions
03 $\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$
04 **return** par

KEYGENERATION(par)
05 N ++
06 $(\mathsf{pk}_N, \mathsf{sk}_N) \leftarrow \mathsf{KeyGen}(\mathsf{par})$
07 **return** pk_N

SESSION$_\text{B}(i, \mathcal{P}_i)$
08 **if** $\mathcal{P}_i \not\subseteq [N]$: **return** $\perp$ //subset of known pks
09 **if** $i \notin [N]$: **return** $\perp$ //key material exists
10 **if** $i \in \mathcal{P}_i$: **return** $\perp$ //all public keys distinct
11 cnt_1 ++
12 sID $\coloneqq \text{cnt}_1$
13 holder[sID] $\coloneqq i$
14 peers[sID] $\coloneqq \mathcal{P}_i$
15 $(m_i, \mathsf{st}) \leftarrow \mathsf{Begin}(\mathsf{sk}_i, \{\mathsf{pk}_j\}_{j \in \mathcal{P}_i})$
16 I[sID] $\coloneqq \{(i, m_i)\}$ //store first extended message
17 state[sID] $\coloneqq$ st
18 stage[sID] $\coloneqq 2$
19 **return** (sID, m_i)

SESSION$_\text{R}$(sID, $\mathcal{M}$)
20 **parse** $\{(j, m_j)\}_{j \in \mathcal{P}_i} \coloneqq \mathcal{M}$
21 **if** stage[sID] $\neq 2$: **return** $\perp$ //session not created
22 **if** $|\mathcal{M}| \neq |\text{peers[sID]}|$: **return** $\perp$ //correct no. msgs.
23 **if** $\{j \mid \exists m_j \text{ s.t. } (j, m_j) \in \mathcal{M}\} \neq \text{peers[sID]}$: **return** $\perp$
//all partners have sent messages
24 $(i, \mathcal{P}_i) \coloneqq (\text{holder[sID}_1], \text{peers[sID}_1])$
25 peersCorr[sID] $\coloneqq \bigvee_{j \in \mathcal{P}_i} \text{corr}[j]$
26 $\mathcal{M}$[sID] $\coloneqq \mathcal{M}$
27 $(\hat{m}_i, \mathsf{st}) \leftarrow \mathsf{Respond}(\mathsf{sk}_i, \text{state[sID]}, \mathcal{M})$
28 R[sID] $\coloneqq \{(i, \hat{m}_i)\}$ //store second extended message
29 state[sID] $\coloneqq$ st
30 stage[sID] $\coloneqq 3$
31 **return** $(\text{sID}, \hat{m}_i)$

REVEAL(sID)
32 revealed[sID] $\coloneqq$ **true**
33 **return** gKey[sID]

SESSION$_\text{F}$(sID, $\hat{\mathcal{M}}$)
34 **parse** $\{(j, \hat{m}_j)\}_{j \in \mathcal{P}_i} \coloneqq \hat{\mathcal{M}}$
35 **if** stage[sID] $\neq 3$: **return** $\perp$
36 **if** $|\hat{\mathcal{M}}| \neq |\text{peers[sID]}|$: **return** $\perp$
37 **if** $\{j \mid \exists \hat{m}_j \text{ s.t. } (j, \hat{m}_j) \in \hat{\mathcal{M}}\} \neq \text{peers[sID]}$: **return** $\perp$
38 $(i, \mathcal{P}_i) \coloneqq (\text{holder[sID}_1], \text{peers[sID}_1])$
39 peersCorr[sID] $\coloneqq \bigvee_{j \in \mathcal{P}_i} \text{corr}[j]$
40 $\hat{\mathcal{M}}$[sID] $\coloneqq \hat{\mathcal{M}}$
41 $(\bar{m}_i, \mathsf{st}) \leftarrow \mathsf{Final}(\mathsf{sk}_i, \text{state[sID]}, \hat{\mathcal{M}})$
42 F[sID] $\coloneqq \{(i, \bar{m}_i)\}$ //store third extended message
43 state[sID] $\coloneqq$ st
44 stage[sID] $\coloneqq 4$
45 **return** $(\text{sID}, \bar{m}_i)$

DER(sID, $\bar{\mathcal{M}}$)
46 **parse** $\{(j, \bar{m}_j)\}_{j \in \mathcal{P}_i} \coloneqq \bar{\mathcal{M}}$
47 **if** stage[sID] $\neq 4$: **return** $\perp$
48 **if** $|\bar{\mathcal{M}}| \neq |\text{peers[sID]}|$: **return** $\perp$
49 **if** $\{j \mid \exists \bar{m}_j \text{ s.t. } (j, \bar{m}_j) \in \bar{\mathcal{M}}\} \neq \text{peers[sID]}$: **return** $\perp$
50 **if** gKey[sID] $\neq \perp$: **return** $\perp$ //key already computed
51 $(i, \mathcal{P}_i) \coloneqq (\text{holder[sID}_1], \text{peers[sID}_1])$
52 peersCorr[sID] $\coloneqq \bigvee_{j \in \mathcal{P}_i} \text{corr}[j]$
53 $\bar{\mathcal{M}}$[sID] $\coloneqq \bar{\mathcal{M}}$
54 $K \coloneqq \mathsf{Derive}(\mathsf{sk}_i, \text{state[sID]}, \bar{\mathcal{M}})$
55 gKey[sID] $\coloneqq K$
56 stage[sID] $\coloneqq 5$
57 **return** ϵ

REV-STATE(sID)
58 revState[sID] $\coloneqq$ **true**
59 **return** state[sID]

CORRUPT(n)
60 corr[n] $\coloneqq$ **true**
61 **return** sk_n

TEST(sID)
62 **if** sID $\in \mathcal{S}$: **return** $\perp$
63 **if** gKey[sID] $= \perp$: **return** $\perp$
64 $\mathcal{S} \coloneqq \mathcal{S} \cup \{\text{sID}\}$
65 $K_0^* \coloneqq \text{gKey[sID]}$
66 $K_1^* \leftarrow \mathcal{K}$
67 **return** K_b^*

FINALIZE(b')
68 **for** $\text{sID}^* \in \mathcal{S}$: //multiple test sessions
69 **for** $P \in \text{peers[sID}^*]$:
70 **if** $\text{GVALID}_{\mathsf{PFS}}(\text{sID}^*, P) =$ **false** :
71 **return** 0 //no valid attack
72 **return** b'

GVALID$_{\mathsf{PFS}}$(sID*, P)
73 $i^* \coloneqq \text{holder[sID}^*]$
74 $\mathfrak{M}(\text{sID}^*, P) \coloneqq \{\text{sID} \mid \text{holder[sID}] = P \wedge I[\text{sID}] \cup \mathcal{M}[\text{sID}] = I[\text{sID}^*] \cup \mathcal{M}[\text{sID}^*]$
$\wedge\, R[\text{sID}] \cup \hat{\mathcal{M}}[\text{sID}] = R[\text{sID}^*] \cup \hat{\mathcal{M}}[\text{sID}^*] \wedge F[\text{sID}] \cup \bar{\mathcal{M}}[\text{sID}] = F[\text{sID}^*] \cup \bar{\mathcal{M}}[\text{sID}^*]\}$
75 $\mathfrak{P}(\text{sID}^*, P) \coloneqq \{\text{sID} \mid \text{holder[sID}] = P \wedge I[\text{sID}] \cup \mathcal{M}[\text{sID}] = I[\text{sID}^*] \cup \mathcal{M}[\text{sID}^*]$ //part. match. sess. to P
$\wedge\, R[\text{sID}] \cup \hat{\mathcal{M}}[\text{sID}] = R[\text{sID}^*] \cup \hat{\mathcal{M}}[\text{sID}^*]\}$
76 **if** revealed[sID*] **or** $(\exists \text{sID} \in \mathfrak{M}(\text{sID}^*, P) : \text{revealed[sID}] =$ **true**) //session or partner revealed
or $\exists \text{sID} \in \mathfrak{M}(\text{sID}^*, P)$ s.t. sID $\in \mathcal{S}$: //Test was asked for two partnered sessions
77 **return false**
78 **for** attack $\in$ Table 2
79 **if** attack $=$ **true** :
80 **return true**
81 **return false**

Fig. 7: Games GAKE-$\mathsf{G}_{\mathsf{PFS},b}$ for GAKE, where $b \in \{0, 1\}$. Adversary $\mathcal{A}$ has access to oracles $\text{O} \coloneqq \{\text{INITIALIZE}, \text{KEYGENERATION}, \text{SESSION}_\text{B}, \text{SESSION}_\text{R}, \text{SESSION}_\text{F}, \text{DER}, \text{REVEAL}, \text{REV-STATE}, \text{CORRUPT}, \text{TEST}, \text{FINALIZE}\}$. Helper procedure GVALID$_{\mathsf{PFS}}$ captures perfect forward secrecy and verifies the validity of attacks against the GAKE protocol. If there exists any test session which is not valid, the game will return 0.

$\mathcal{A}$ gets (holder, P)	$\mathrm{corr}[i^*]$	$\mathrm{corr}[P]$	$\mathrm{peersCorr}[\mathrm{sID}^*]$	$\mathrm{revState}[\mathrm{sID}^*]$	$\exists \mathrm{sID} \in \mathfrak{M}(\mathrm{sID}^*, P) : \mathrm{revState}[\mathrm{sID}]$	$\lvert\mathfrak{M}(\mathrm{sID}^*, P)\rvert$	$\exists \mathrm{sID} \in \mathfrak{P}(\mathrm{sID}^*, P) : \mathrm{revState}[\mathrm{sID}]$	$\lvert\mathfrak{P}(\mathrm{sID}^*, P)\rvert$
(0) **multiple partially matching sessions**	-	-	-	-	-	-	-	>1
(1) **(long-term, long-term)**	-	-	-	**F**	**F**	1	-	-
(2) **(state, state)**	**F**	**F**	-	-	-	1	-	-
(3) **(long-term, state)**	-	**F**	-	**F**	-	1	-	-
(4) **(state, long-term)**	**F**	-	-	-	**F**	1	-	-
(5) **(long-term, long-term)**	-	-	**F**	**F**	n/a	0	**F**	1
(6) **(state, state)**	**F**	**F**	-	-	n/a	0	-	1
(7) **(long-term, state)**	-	**F**	-	**F**	n/a	0	-	1
(8) **(state, long-term)**	**F**	-	**F**	-	n/a	0	**F**	1
(9) **(long-term, long-term)**	-	-	**F**	**F**	n/a	0	n/a	0
(10) **(state, state)**	**F**	**F**	-	-	n/a	0	n/a	0

Table 2: Attack table describing valid attacks for perfect forward secrecy. An attack is regarded as an AND conjunction of variables with specified values as shown in the each line, where "–" means that this variable can take arbitrary value, "**F**" means "false", "n/a" indicates that the result is trivially "false" because of the definition of (partially) matching sessions.

- Attacks (1)-(4) deal with matching sessions, where we essentially capture all combinations of reveal queries.
- Attacks (5)-(8) capture partially matching sessions which are the same as (1)-(4), except that we need to look at the state of those sessions in set $\mathfrak{P}$.
- Attacks (9)-(10) look at sessions where the peer does not hold a session with the same view (hence, the adversary actively modified communication). Here, we need to be more restrictive since the adversary can pick some of the states itself, in which case we cannot allow it to also reveal the long-term key.

Definition 9. *We define the game* $\mathsf{GAKE\text{-}G}_{\mathsf{PFS},b}$ *for* $b \in \{0,1\}$ *as in Figure 7. The advantage of an adversary* $\mathcal{A}$ *against* GAKE *in this game is defined as*

$$\mathrm{Adv}^{\mathsf{PFS}}_{\mathsf{GAKE}}(\mathcal{A}) := \left|\Pr[\mathsf{GAKE\text{-}G}^{\mathcal{A}}_{\mathsf{PFS},1} \Rightarrow 1] - \Pr[\mathsf{GAKE\text{-}G}^{\mathcal{A}}_{\mathsf{PFS},0} \Rightarrow 1]\right| .$$

5 GAKE from UAKE

We construct a GAKE protocol from a UAKE protocol as shown in Figure 8. Each party will run the UAKE protocol twice to generate two fresh symmetric keys. While doing this, each party uses the first UAKE run to authenticate itself to the predecessor (acting as responder in the UAKE), and the other to authenticate its successor (acting as initiator). In the second phase, each party will encrypt the key that it shares with its predecessor to its successor. In this way, shared keys will only be made available to parties that have been authenticated. In the final step, each party will step-wisely compute all the pairwisely shared keys and use them to derive the final group key.

To protect critical information from the attacker, we will encrypt the state information with the long-term key. The state information consists of all the information that needs to be pertained between rounds for the the protocol to work properly. To simplify, we will in our description encrypt all state information. Indeed some of the state information of sessions

can be derived by the attacker publicly. Encrypting the entire state information allows us to abstract and more generically describe the mechanisms we use.

Remark 1. One may view the state encryption as a weaker variant of the NAXOS trick [LLM07] that is sufficient to capture state attacks that reveal the information that needs to be stored between rounds. In contrast to long-term secrets, this information may be more easily accessible (and may not be securely erased from memory). This definition of state attacks is weaker since it cannot model compromise or manipulation of randomness. However, when considering tightness, the latter is also much harder to achieve. For example, when using the NAXOS trick, the reduction must be able to output "valid-looking" randomness for all sessions since it does not know which ones will be tested.

We will rely on perfect forward secrecy of UAKE to achieve perfect forward secrecy of GAKE, hence we refer to the protocols as $\mathsf{UAKE}_{\mathsf{PFS}}$ and $\mathsf{GAKE}_{\mathsf{PFS}}$, respectively. We will later explain why the same implication does not seem to directly hold for weak forward secrecy.

5.1 Correctness

It can be shown that the final construction has overwhelming correctness if the underlying UAKE has overwhelming correctness.

Lemma 2. *Consider the construction in Figure 8. If* $\mathsf{UAKE}_{\mathsf{PFS}}$ *has overwhelming correctness* $\rho = 1 - 1/2^v$ *for some* $v \in \Omega(\kappa)$ *and the attacker makes* q *queries overall, then* $\mathsf{GAKE}_{\mathsf{PFS}}$ *has overwhelming correctness at least* $1 - q/2^v$.

Proof. Assume $\mathsf{UAKE}_{\mathsf{PFS}}$ has overwhelming correctness $\rho = 1 - 1/2^v$ for some $v \in \Omega(\kappa)$. First, observe that by setup this is the only source of non-perfect correctness in the entire protocol construction. Also observe that if $\mathsf{UAKE}_{\mathsf{PFS}}$ has no correctness errors at all, then we will not have any correctness error in our $\mathsf{GAKE}_{\mathsf{PFS}}$ construction as well. So we only have to analyse the influence of $\mathsf{UAKE}_{\mathsf{PFS}}$ on the overall correctness. Now, since a single application of $\mathsf{UAKE}_{\mathsf{PFS}}$ has overwhelmingly high correctness ρ, a q-time application of $\mathsf{UAKE}_{\mathsf{PFS}}$ will result in a correctness of at least $(1-1/2^v)^q$. This can be lower bounded via $(1-1/2^v)^q \geq 1-q/2^v$ for some arbitrary polynomial $q = q(\kappa)$ due the Bernoulli's inequality, which shows that the resulting correctness is still overwhelming. □

5.2 Security

We now prove the security of our construction. Informally, if UAKE has perfect forward secrecy, then the resulting GAKE also has perfect forward secrecy. This is captured in the following theorem.

Theorem 2 (UAKE to GAKE). *For any adversary* $\mathcal{A}$ *against* $\mathsf{GAKE\text{-}G}_{\mathsf{PFS},b}$ *with protocol* $\mathsf{GAKE}_{\mathsf{PFS}}$ *with* N *parties that establish at most* S *sessions and issues at most* q *queries to the oracles, there exists an adversary* $\mathcal{B}$ *against* $\mathsf{OW\text{-}UAKE\text{-}G}_{\mathsf{PFS}}$ *of a protocol* $\mathsf{UAKE}_{\mathsf{PFS}}$ *with min-entropy* μ *and correctness* $(1 - 1/2^v)$, *such that*

$$\begin{aligned}\mathrm{Adv}^{\mathsf{PFS}}_{\mathsf{GAKE}_{\mathsf{PFS}}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{PFS}}_{\mathsf{UAKE}_{\mathsf{PFS}}}(\mathcal{B}) + \frac{Sq_{RS} + Nq_C}{2^\kappa} + \frac{Sq_{\mathsf{H}_{\mathsf{ukey}}}}{2^\tau} + \frac{Sq_{\mathsf{H}_{\mathsf{tag}}}}{2^\tau} \\ + \frac{Sq_{\mathsf{H}_{\mathsf{gkey}}}}{|\mathcal{K}_{\mathsf{GAKE}_{\mathsf{PFS}}}|} + \frac{N^2+S^2}{2^\mu} + \frac{N^2+3S^2}{2^\kappa} + \frac{q_{RO}^2}{2^\zeta} + \frac{q}{2^v}\ ,\end{aligned}$$

where all hash functions are modeled as random oracles and the running time of $\mathcal{B}$ *is about that of* $\mathcal{A}$.

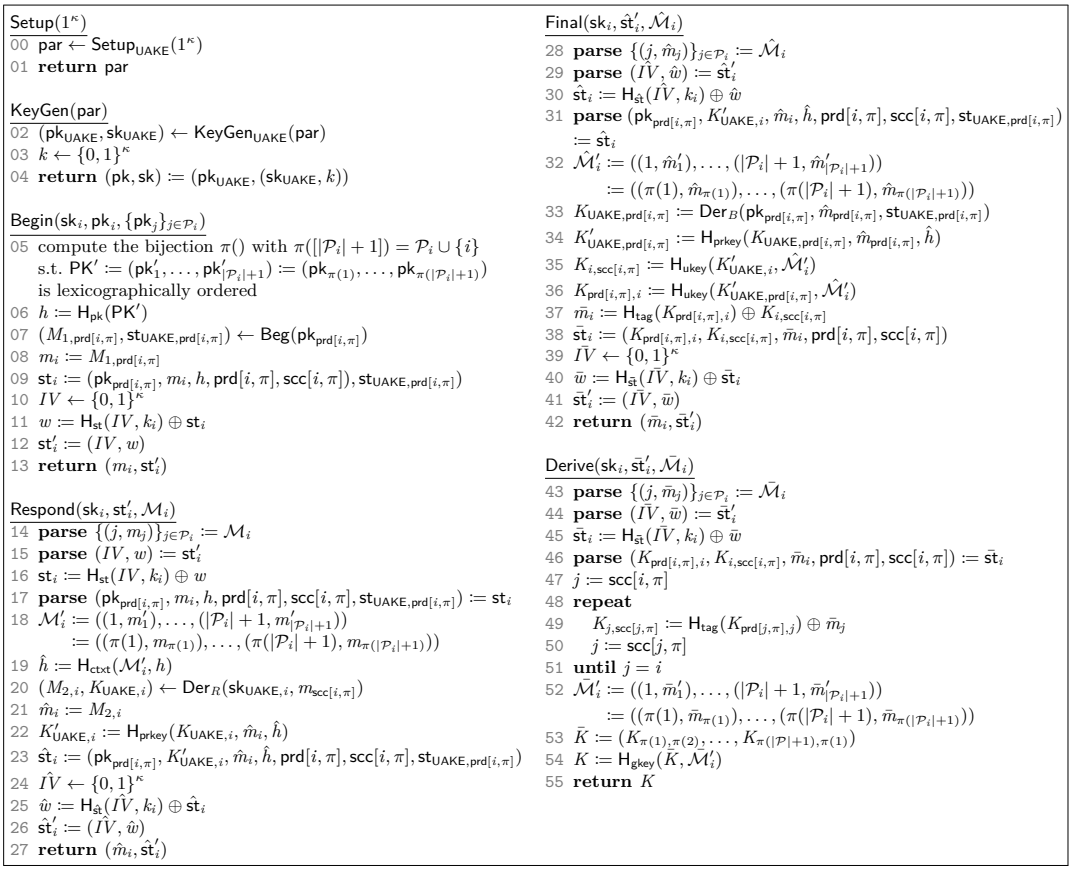

Fig. 8: Generic construction of $\mathsf{GAKE}_{\mathsf{PFS}}$ from $\mathsf{UAKE}_{\mathsf{PFS}} = (\mathsf{Setup}_{\mathsf{UAKE}}, \mathsf{KeyGen}_{\mathsf{UAKE}}, \mathsf{Beg}, \mathsf{Der}_R, \mathsf{Der}_B)$.

Let us first sketch the proof. We proceed in a series of games. In the first steps, we make sure that all outputs of the sessions are distinct by relying on the min-entropy of the UAKE. Next, we exclude collisions in the random oracles. Finally, we assume that all UAKE runs feature correctness. Each time this accounts only for a statistically small change in the success probability of the attacker. Next we make the state of each stage independent of all the initial values and secret parameters. At the same time, we ensure consistency by adapting the random oracle used for state encryption to generate an output - on the fly - that is used to encrypt the state. In fact, the reduction creates it such that it can successfully appear to have encrypted the state beforehand. In this step, we essentially exploit that the hash function in the state encryption is modeled as a random oracle. After that, we change how the keys $K'_{\mathsf{UAKE},i}$ are computed and make them independent of all previously computed values while guaranteeing consistency with all the queries of the attacker. Again, we exploit that the underlying hash function is used as a random oracle. We now have that uKey$'$ is now independent of uKey. Now we partition the space of all sessions into two categories.

1) The set of sessions that still, when tested, could amount to a subset S of valid attacks (with respect to the predicate GVALID).
2) The set of sessions for which we already have certainty, that they can never amount to a valid attack in S.

Specifically, we will show that sessions of type 1) imply the existence of an algorithm that breaks the underlying UAKE scheme. Put differently, the attacks in 1) will correspond to an attacker that can compute the UAKE key in a non-trivial way. For the remaining sessions we will next apply an argument which guarantees that the keys $K'_{\mathsf{UAKE},\mathsf{prd}[\mathrm{holder[sID]},\pi]}$ and $K'_{\mathsf{UAKE},\mathrm{holder[sID]}}$ are indistinguishable from random. This accounts for an additional

statistically small change in the success probability. As a result, we now only consider group keys that are indistinguishable from random (with overwhelming probability).

Remark 2. One might wonder whether we can use our WFS-secure UAKE (without the additional hash) to construct WFS-secure GAKE. However, if UAKE is only WFS-secure, the resulting GAKE protocol does not seem to be WFS-secure. This is because the active adversary can always compute one UAKE key, wait until all parties have sent their third messages and recompute the second UAKE key to send its own message. This is not possible when using UAKE with key confirmation because the adversary will not be able to authenticate towards its predecessor. Hence, the predecessor will not send the third message and prevents the above attack.

Proof (Theorem 2). Let $\mathcal{A}$ be an adversary against $\mathsf{GAKE\text{-}G}_{\mathsf{PFS},b}$ with protocol $\mathsf{GAKE}_{\mathsf{PFS}}$ as defined in Figure 8. We consider the sequence of games in Figures 9 to 11.

Game $\mathsf{G}_{0,b}$. This is the same as $\mathsf{GAKE\text{-}G}_{\mathsf{PFS},b}$, except for small changes. We store the unencrypted state in an additional variable state′[sID] and do not decrypt it explicitly. This is only conceptual. We also implicitly exclude collisions and if a collision happens at any time in the game, the experiment aborts. We also make sure that key pairs and messages are distinct. Using the fact that $\mathsf{UAKE}_{\mathsf{PFS}}$ has min-entropy μ, the upper bound for key collisions is $N^2/2^\mu$ and for message collisions, it is $S^2/2^\mu$. Moreover, we assume that values k_n, for $n \in [N]$, and $IV, \hat{IV}, \bar{IV}$ (at most S for each one) are distinct, and this is provided with probability at most $(N^2+3S^2)/2^\kappa$. In the end, we aim for all random oracle outputs to be unique. Assuming ζ is the lower bound for all dimensions of the random oracle outputs, collisions are excluded with a probability of at most $q_{RO}^2/2^\zeta$, where $q_{RO} \leq q$. All the probabilities above also follow from the birthday bound. Finally, in this step we abort if any of the UAKE runs of the challenger do not feature correctness. However, as analysed before this only happens with probability $q/2^v$.

We get

$$\left|\Pr[\mathsf{GAKE\text{-}G}^{\mathcal{A}}_{\mathsf{PFS},0} \Rightarrow 1] - \Pr[\mathsf{GAKE\text{-}G}^{\mathcal{A}}_{\mathsf{PFS},1} \Rightarrow 1]\right| \leq$$

$$\left|\Pr[\mathsf{G}^{\mathcal{A}}_{0,0} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}}_{0,1} \Rightarrow 1]\right| + \frac{N^2+S^2}{2^\mu} + \frac{q_{KG}^2+3S^2}{2^\kappa} + \frac{q_{RO}^2}{2^\zeta} + \frac{q}{2^v} \ .$$

Game $\mathsf{G}_{1,b}$. In games $\mathsf{G}_{1,b}$, we make the state of each stage independent of all the initial values and secret parameters. At the same time, we ensure consistency by adapting the random oracle $\mathsf{H}_{\mathsf{st}^*}$ to generate an output - on the fly - that is used to encrypt the state (actually the reduction creates it such that it can successfully appear to have encrypted the state beforehand). Now the initial values are computed in the Rev-State oracle and the long-term secret values k_n are computed in the Corrupt oracle. We raise flag BAD_{IV} in line 27 (Fig. 11) and abort if the Rev-State oracle chooses an initial value IV that was issued, together with the secret key of the corresponding holder of the session, to the $\mathsf{H}_{\mathsf{st}^*}$ oracle before. The probability that BAD_{IV} is raised for one specific IV is at most $q_{RS}/2^\kappa$, where q_{RS} indicates the number of queries issued to the Rev-State oracle and $q_{RS} \leq q$. An union bound gives us

$$\Pr[\mathrm{BAD}_{IV}] \leq \frac{Sq_{RS}}{2^\kappa} \ . \tag{1}$$

We also raise flag BAD_k in line 38 (Fig. 11) and abort if, for the chosen secret value k_n computed by the Corrupt oracle, there exist an initial value IV such that both were issued to the $\mathsf{H}_{\mathsf{st}^*}$ oracle before. The probability that BAD_k is raised for one specific k_n is at most $q_C/2^\kappa$, where q_C indicates the number of queries issued to the Corrupt oracle and $q_C \leq q$. Again, an union bound gives us

$$\Pr[\mathrm{BAD}_k] \leq \frac{Nq_C}{2^\kappa}. \tag{2}$$

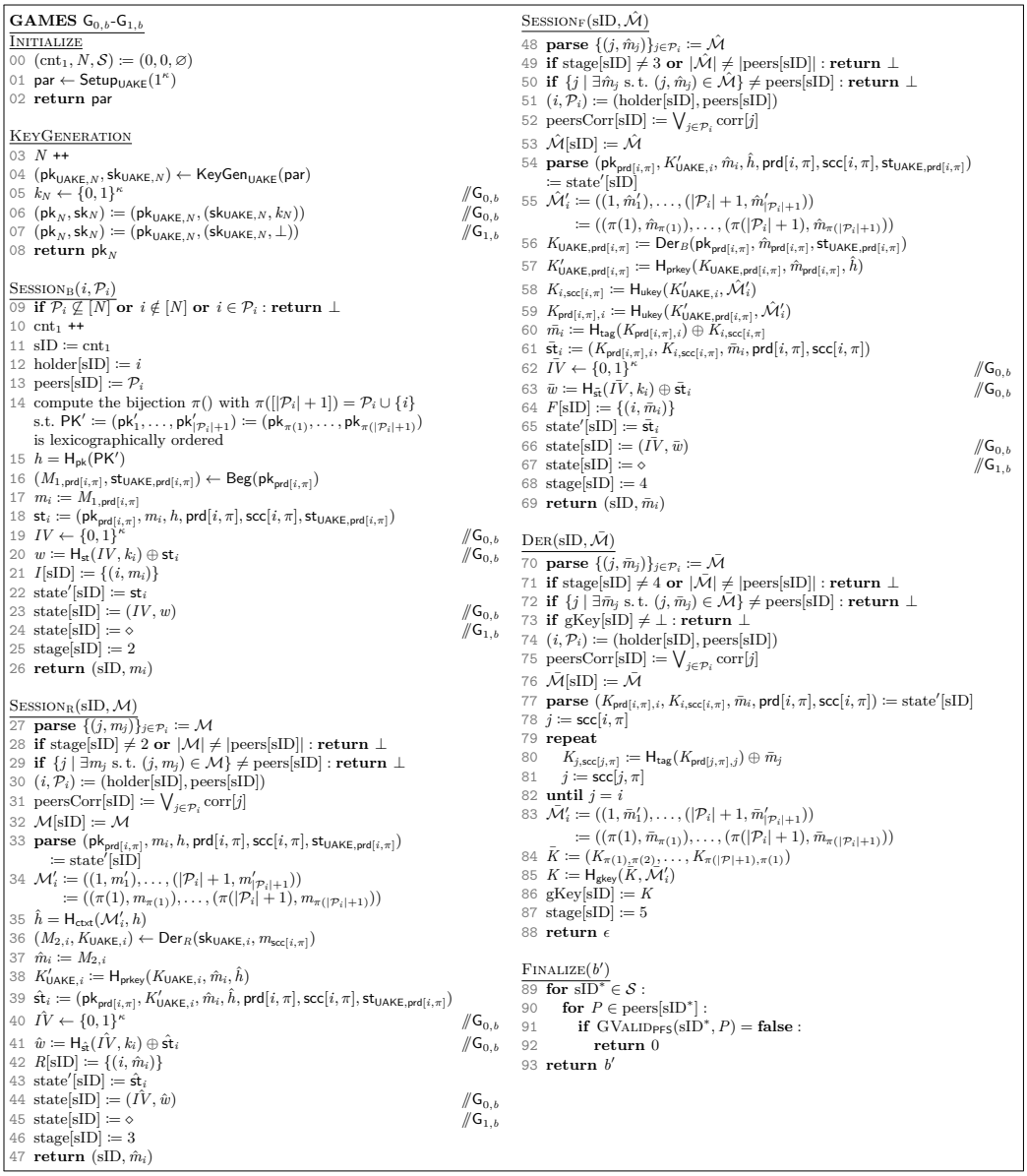

Fig. 9: Games $\mathsf{G}_{0,b}, \mathsf{G}_{1,b}$ for the proof of Theorem 2. Adversary $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\textsc{Initialize}, \textsc{Session}_\mathrm{B}, \textsc{Session}_\mathrm{R}, \textsc{Session}_\mathrm{F}, \textsc{Der}, \textsc{Reveal}, \textsc{Rev-State}, \textsc{Corrupt}, \textsc{Test}, \mathsf{H}_{\mathsf{st}^*}, \mathsf{H}_{\mathsf{pk}}, \mathsf{H}_{\mathsf{gkey}}, \mathsf{H}_{\mathsf{ukey}}, \mathsf{H}_{\mathsf{ctxt}}, \mathsf{H}_{\mathsf{tag}}\, \mathsf{H}_{\mathsf{prkey}}, \textsc{Finalize}\}$, where oracles $\textsc{Test}$ and $\textsc{Reveal}$ are defined as in Figure 7 and all other oracles are defined in Figure 11.

Then, from equations 1 and 2 we have

$$|\Pr[\mathsf{G}^{\mathcal{A}}_{1,b} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}}_{0,b} \Rightarrow 1]| \leq \frac{Sq_{RS} + Nq_C}{2^\kappa} \ .$$

$\textsc{Game}\ \mathsf{G}_{2,b}$. This is a bridging step. We essentially change how the keys $K'_{\mathsf{UAKE},i}$ are computed and make them independent of all previously computed values while guaranteeing consistency with all the queries of the attacker. We also introduce two helper variables, uKey and uKey$'$, that store the keys $K_{\mathsf{UAKE},i}$ and $K'_{\mathsf{UAKE},i}$ for later use if needed.

$$\Pr[\mathsf{G}^{\mathcal{A}}_{2,b} \Rightarrow 1] = \Pr[\mathsf{G}^{\mathcal{A}}_{1,b} \Rightarrow 1] \ .$$

Fig. 10: Game $\mathsf{G}_{2,b}$ for the proof of Theorem 2. Adversary $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\textsc{Initialize}, \textsc{Session}_\mathrm{B}, \textsc{Session}_\mathrm{R}, \textsc{Session}_\mathrm{F}, \textsc{Der}, \textsc{Reveal}, \textsc{Rev-State}, \textsc{Corrupt}, \textsc{Test}, \mathsf{H}_{\mathsf{st}^*}, \mathsf{H}_{\mathsf{pk}}, \mathsf{H}_{\mathsf{gkey}}, \mathsf{H}_{\mathsf{ukey}}, \mathsf{H}_{\mathsf{ctxt}}, \mathsf{H}_{\mathsf{tag}}\ \mathsf{H}_{\mathsf{prkey}}, \textsc{Finalize}\}$, where oracles $\textsc{Test}$ and $\textsc{Reveal}$ are defined as in Figure 7 and all other oracles are defined in Figure 11.

$\textsc{Game}\ \mathsf{G}_{3,b}$. In the previous game we have already changed the way uKey$'$ is computed. In particular, it is now independent of uKey while the simulation is still consistent. This holds for all sessions. Now we partition the space of all sessions into two categories.

1) The set of sessions that still, when tested, could amount to a subset S of valid attacks (with respect to the predicate GVALID).
2) The set of sessions for which we already have certainty, that they can never amount to a valid attack in S.

Specifically, we will show that sessions of type 1) imply the existence of an algorithm that breaks the underlying UAKE scheme. Intuitively, the attacks in 1) will correspond to an attacker that can compute the UAKE key in a non-trivial way. Essentially, the conditions

require sID to be a session that has successfully computed an UAKE key in $\textsc{Session}_{\mathrm{R}}$ (or $\textsc{Session}_{\mathrm{F}}$) via a successive run of Beg and Der_B (or Der_R). Technically, in this step, we abort if the attacker has raised flag BAD in lines 68 and 73 (Fig. 11). We now need to show that whenever this happens, we can immediately break the security of the underlying UAKE. The flag BAD is only set in case some preliminary conditions are fulfilled. This is the case if there is a specific query $w = (K, m, h)$ to the random oracle $\mathsf{H}_{\mathsf{prkey}}$ such that:

i) the query w has never been queried before to $\mathsf{H}_{\mathsf{prkey}}$;
ii) there exists a session sID that has helper variable uKey – which is either $K_{\mathsf{UAKE},\mathrm{holder[sID]}}$ of the holder of that session (case (a)) or $K_{\mathsf{UAKE},\mathsf{prd}[\mathrm{holder[sID]},\pi]}$ of the predecessor (case (b)) – be equal to K;
iii) the hash of the context so far $\hat{h}[\mathrm{sID}] = \mathsf{H}_{\mathsf{ctxt}}(\mathcal{M}_i[\mathrm{sID}], h[\mathrm{sID}])$ is equal to h;
iv) the message m is equal to the message $m_{\mathrm{holder[sID]}}$ in case (a) and $m_{\mathsf{prd}[\mathrm{holder[sID]},\pi]}$ in case (b).

Now, the flag will only be set if additionally in case (a), we have that there is a non-trivial attack when testing sID with peer $\mathsf{scc}[\mathrm{holder[sID]}, \pi]$ and in case (b) there is a non-trivial attack with peer $\mathsf{prd}[\mathrm{holder[sID]}, \pi]$. All these conditions map to a specific non-trivial attack that transfers to an underlying attack on $\mathsf{UAKE}_{\mathsf{PFS}}$: if there is a non-trivial attack on $\mathsf{GAKE}_{\mathsf{PFS}}$ under these conditions we have that any of the lines (0), (1), (2), or (3) in Table 1a are fulfilled. We show this by analyzing what – under the conditions i), ii), iii), iv) respectively – a non-trivial attack on the GAKE protocol means. Importantly, we show that under these conditions, attacks on the GAKE protocol will *always* result in an attack on the UAKE protocol.

Let us begin by considering the reduction in Figure 12 and 13. Observe that the output distribution of the queries $\textsc{Initialize}$, $\textsc{KeyGeneration}$, $\textsc{Session}_{\mathrm{B}}$, $\textsc{Session}_{\mathrm{R}}$, $\textsc{Session}_{\mathrm{F}}$, $\textsc{Corrupt}$, $\textsc{Der}$, $\textsc{Test}$, $\textsc{Finalize}$ and all outputs of the queries to the random oracle are distributed like in Game 2, except for $\mathsf{H}_{\mathsf{prkey}}$. Thus, we need to formally show that $\mathsf{H}_{\mathsf{prkey}}$, and $\textsc{Rev-State}$ are distributed like in Game 2 unless the GAKE attacker breaks the underlying UAKE security game.

Let us begin with $\textsc{Rev-State}$. The introduced changes will complete the state of the GAKE with the state of the underlying UAKE if needed. By the modifications made in the last games, we have ensured that in no other place the state of the underlying UAKE is required. Also, observe that the format of the full state of the GAKE protocol means that only the last part of it refers to the underlying UAKE. So by appropriate projections, the state $\mathrm{state}'[\mathrm{sID}]$ can give information on both the underlying UAKE state and the remaining state of the GAKE.

Now, let us have a look at the remaining query $\mathsf{H}_{\mathsf{prkey}}$. It makes the underlying UAKE decide whether there has actually been a non-trivial attack on session sID^*. To this end, it calls the $\textsc{Check}$ query of the UAKE. We will in the following detail which subset of the attacks on the GAKE will correspond to an attack on the UAKE by comparing the attack tables that both schemes use. The query $\mathsf{H}_{\mathsf{prkey}}$ uses the $\textsc{Check}$ query of the UAKE to identify if the input session key is consistent with some session $\mathrm{sdcnt}_1[\mathrm{sID}]$ or $\mathrm{sdcnt}_2[\mathrm{sID}]$. Let us begin our analysis of the conditions in the attack tables: first, we consider all GAKE attacks that would imply an attack on the UAKE according to line (0) in Table 1a. Such an attack happens if, in any UAKE run, there is a receiver session such that there are multiple initiator sessions that share the first message with the receiver (partially matching sessions). However, since we have already excluded collisions in the first message due to its high entropy this is impossible.

Let us now consider Table 2. Consider lines (0) to (8). The lines all imply that there are two neighbors that have matching sessions since a partially matching session in GAKE implies a fully matching session in the underlying UAKE. Since the underlying UAKE has high entropy, we have that the GAKE attack of line (0) can be excluded with overwhelming probability. We remark that, for the GAKE to have partially matching sessions, two sessions must agree on the first *two* messages. However, each communication partner participates in

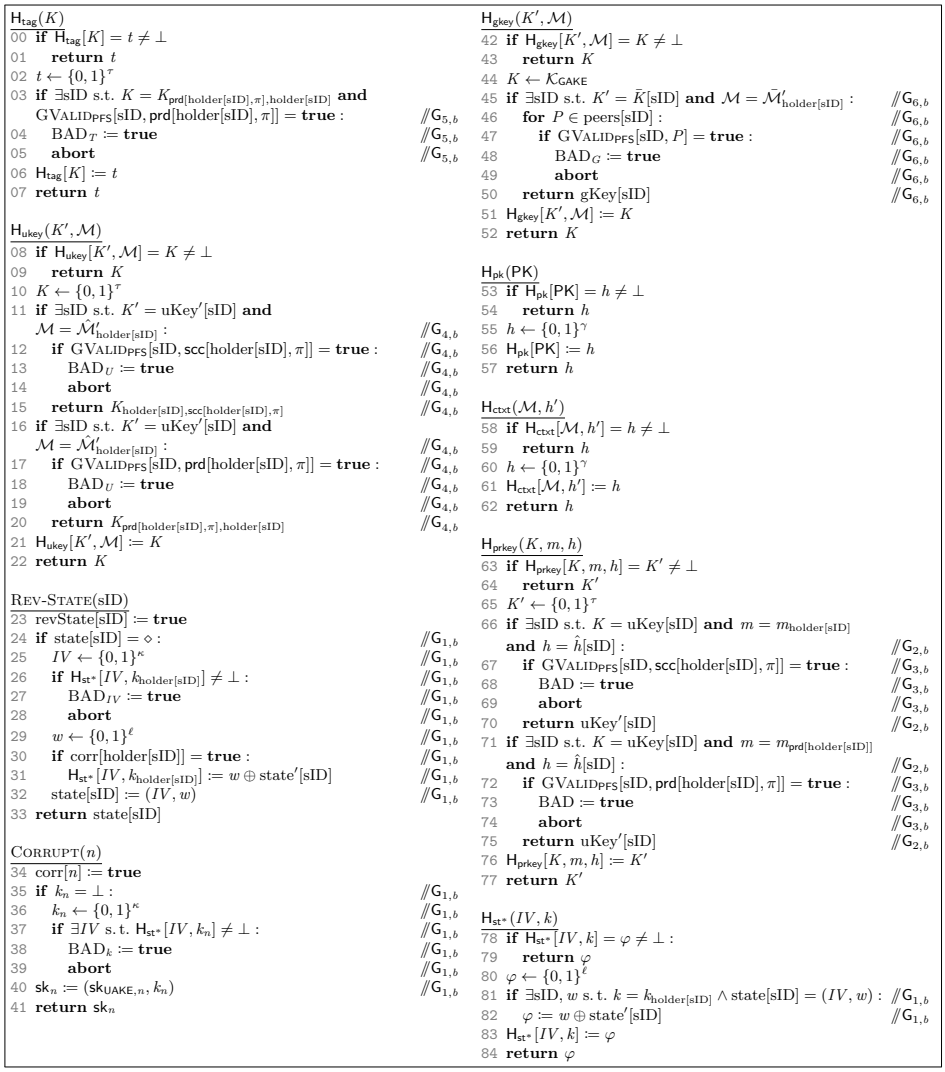

Fig. 11: Additional oracles for games $\mathsf{G}_{0,b}$-$\mathsf{G}_{2,b}$ in Figure 9 and Figure 10 where $\mathsf{H}_{\mathsf{st}^*} \in \{\mathsf{H}_{\mathsf{st}}, \mathsf{H}_{\hat{\mathsf{st}}}, \mathsf{H}_{\bar{\mathsf{st}}}\}$.

generating one of the messages via a contribution of high entropy messages. But, due to the previous modifications of the security game where we excluded collisions, we will not see the same message twice. So we can consider the lines (1) to (8) in the GAKE table that will transfer to sessions of the UAKE that have a single matching session. By definition of non-trivial GAKE attacks, we have that for both, the tested session and the partnered sessions, it must hold that state and long-term key will never be revealed at the same time. Both scenarios can be simulated efficiently without immediately presenting a trivial attack according to the UAKE definition. The PFS notion guarantees that we cover corruptions of the long-term key. (For this, WFS would actually be sufficient at this point in the proof.) Importantly now, albeit for non-tested sessions revealing the state of the GAKE can be transferred to revealing states in the UAKE, now the state of the UAKE will never be revealed.

The underlying argument is that by revealing the state of the GAKE, the attacker only obtains the encrypted UAKE state $w = \mathsf{H}_{\mathsf{st}}(IV, k) + \mathsf{st}$ (or $\hat{w} = \mathsf{H}_{\hat{\mathsf{st}}}(\hat{IV}, k) + \hat{\mathsf{st}}$ or $\bar{w} = \mathsf{H}_{\bar{\mathsf{st}}}(\bar{IV}, k) + \bar{\mathsf{st}}$). Moreover, the adversary can only obtain (with overwhelming probability) the current underlying UAKE state (st, $\hat{\mathsf{st}}$, or $\bar{\mathsf{st}}$) at all, by obtaining not only the long-term key k via a CORRUPT query but also the initialization vector (IV, $\hat{IV}$, or $\bar{IV}$) via a

Rev-State query. We emphasize that by the changes introduced in the previous games, there is indeed no other way for the attacker to obtain the UAKE state: since it is modeled as a random oracle, the output of $\mathsf{H}_{\mathsf{st}^*}$ does not reveal anything about its inputs. It is drawn as a uniformly random output and independent of any other values. Additionally, k will never be used for any other purpose than computing encrypted states. As it is part of the long-term key it will never be accessible via state revealing. The initialization vectors will likewise never be used for any other purpose besides state encryption. Moreover, they will only be stored in the state of sessions and are not part of the long-term key.

Now let us consider the remaining lines in the GAKE table, (9) and (10). These lines indicate no partially matching session exists for the tested session. However, since the underlying UAKE protocol has only two moves, this can either transfer to the existence of a partially matching session in the UAKE table – line (2) – or the lack of it – line (3). We note that in both cases we can exploit the PFS property of the UAKE in case the attacker corrupts the holder of the test session.

Now consider lines (9) and (10) more closely and let us restrict our attention to the sub-case where in the resulting UAKE table we have partially matching conversations, i.e. the first message has not been modified on transit. With the same arguments as before on state encryption, the conditions in the GAKE table for both lines guarantee that the attacker will never see the secret state of any of the sessions. This in particular holds for the computations of each GAKE session that correspond to the computations of the responder in the underlying UAKE protocol. So this particular sub-case of attacks on the GAKE protocol will always transfer to an attack according to line (2).

Finally, consider the remaining sub-class of attacks that are characterised by the lack of sessions that do not have partially matching sessions with the tested session in the underlying UAKE, i.e. they do not share the first message. The last line of the UAKE table essentially says that in case no oracle shares the first message with the tested oracle, then all other oracles might as well be fully under the control of the attacker. We do not require the peer to be corrupted and have no other requirements on the remaining oracles (since they are not fully nor partially matching the tested oracle). Also, the two lines in the GAKE table make sure that there is no corruption of the holder of the tested session at all. So in particular, there is no "early" corruption where peersCorr[sID*] is true. Due to the state encryption this immediately guarantees that the UAKE state of the tested session is never revealed. Thus, this sub-case transfers to the sub-case (3) of the UAKE table.

To win, the adversary must successfully guess a crucial input to the computation of $K_{\mathsf{UAKE},i}$ or $K'_{\mathsf{UAKE},i}$ and thus would be able to compute these keys directly. In particular, the attacker has generated an input to the random oracle $\mathsf{H}_{\mathsf{prkey}}$ that constitutes a valid attack. This now immediately reduces to the OW-UAKE-G security game.

$$|\Pr[\mathsf{G}^{\mathcal{A}}_{3,b} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}}_{2,b} \Rightarrow 1]| \leq \Pr[\mathrm{BAD}] \leq \mathrm{Adv}^{\mathsf{PFS}}_{\mathsf{UAKE}_{\mathsf{PFS}}}(\mathcal{B}) \ .$$

Game $\mathsf{G}_{4,b}$. We raise flag BAD_U in lines 13 and 18 (Fig. 11) and abort if there exists a session sID such that a (prime) predecessor UAKE key $K'_{\mathsf{UAKE},\mathsf{prd}[\mathrm{holder}[\mathrm{sID}],\pi]}$ or a (prime) successor UAKE key $K'_{\mathsf{UAKE},\mathrm{holder}[\mathrm{sID}]}$ have been queried (in a valid attack) with the predecessor or, respectively, the successor as peer. The probability that BAD_U is raised for a specific derived key is at most $q_{\mathsf{H}_{\mathsf{ukey}}}/2^\tau$, where $q_{\mathsf{H}_{\mathsf{ukey}}}$ are queries issued to random oracle $\mathsf{H}_{\mathsf{ukey}}$ and $q_{\mathsf{H}_{\mathsf{ukey}}} \leq q$. An union bound, over the number of sessions, gives us

$$|\Pr[\mathsf{G}^{\mathcal{A}}_{4,b} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}}_{3,b} \Rightarrow 1]| \leq \Pr[\mathrm{BAD}_U] \leq \frac{S q_{\mathsf{H}_{\mathsf{ukey}}}}{2^\tau}.$$

Game $\mathsf{G}_{5,b}$. We raise flag BAD_T in line 04 (Fig. 11) and abort if there is a session sID such that a derived predecessor key $K_{\mathsf{prd}[\mathrm{holder}[\mathrm{sID}],\pi],\mathrm{holder}[\mathrm{sID}]}$ is issued in a valid attack. The probability that BAD_T is raised for a specific $K_{\mathsf{prd}[\mathrm{holder}[\mathrm{sID}],\pi],\mathrm{holder}[\mathrm{sID}]}$ is at most $q_{\mathsf{H}_{\mathsf{tag}}}/2^\tau$, where $q_{\mathsf{H}_{\mathsf{tag}}}$ are queries issued to random oracle $\mathsf{H}_{\mathsf{tag}}$ and $q_{\mathsf{H}_{\mathsf{tag}}} \leq q$. Again, an union bound

Fig. 12: Adversary $\mathcal{B}$ against $\mathsf{OW\text{-}UAKE\text{-}G}_{\mathsf{PFS}}$ for the proof of Theorem 2. It has access to oracles $\widetilde{\mathrm{O}} := \{\textsc{Initialize}, \textsc{KeyGeneration}, \mathrm{S}_{\mathsf{Beg}}, \mathrm{S}_{\mathsf{Der}_R}, \mathrm{S}_{\mathsf{Der}_B}, \textsc{Rev-State}, \textsc{Corrupt}, \textsc{Check}, \textsc{Finalize}\}$. The remaining oracles are described in Figure 13.

gives us

$$|\Pr[\mathsf{G}^{\mathcal{A}}_{5,b} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}}_{4,b} \Rightarrow 1]| \leq \Pr[\mathrm{BAD}_T] \leq \frac{S q_{\mathsf{H}_{\mathsf{tag}}}}{2^\tau} \ .$$

Game $\mathsf{G}_{6,b}$. We raise flag BAD_G in line 48 (Fig. 11) and abort if there exists a session sID such that a sorted key $\bar{K}$ is issued in a valid attack for all the peers of the session. The probability that BAD_G is raised for a specific $\bar{K}$ is at most $q_{\mathsf{H}_{\mathsf{gkey}}}/|\mathcal{K}_{\mathsf{GAKE}_{\mathsf{PFS}}}|$, where $q_{\mathsf{H}_{\mathsf{gkey}}}$ are queries issued to random oracle $\mathsf{H}_{\mathsf{gkey}}$ and $q_{\mathsf{H}_{\mathsf{gkey}}} \leq q$. An union bound gives us

$$|\Pr[\mathsf{G}^{\mathcal{A}}_{6,b} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}}_{5,b} \Rightarrow 1]| \leq \Pr[\mathrm{BAD}_G] \leq \frac{S q_{\mathsf{H}_{\mathsf{gkey}}}}{|\mathcal{K}_{\mathsf{GAKE}_{\mathsf{PFS}}}|} \ .$$

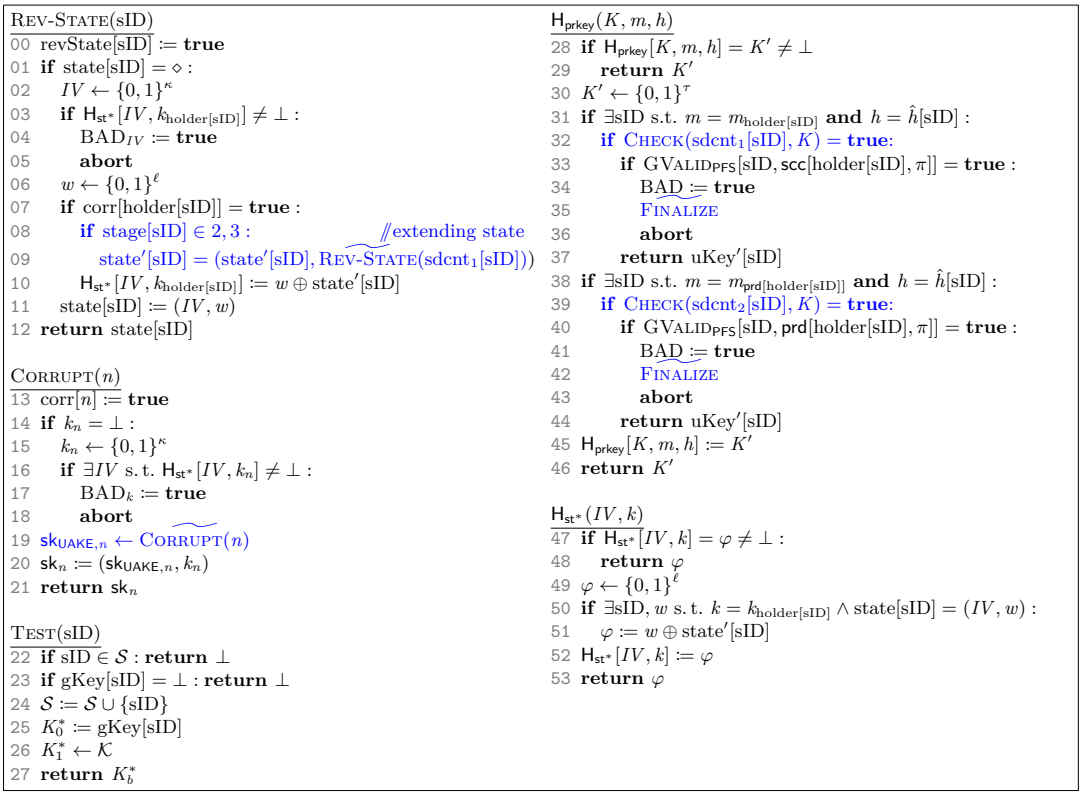

Fig. 13: Additional oracles for adversary $\mathcal{B}$ described in Figure 12.

Now, observe that the attacker does not have any advantage in distinguishing the session key from a random key since for all non-trivial attacks the session key is a random key by the modifications that we made in the sequence of games.

Combining all probabilities, we obtain the bound stated in Theorem 2. □

6 Final GAKE Protocol

We first describe how to construct UAKE from KEMs. Then we give the final construction which uses all transformations from KEM to GAKE.

6.1 UAKE from KEMs

We introduce the syntax for key encapsulation mechanisms and provide the definitions of correctness and min-entropy. The latter is extremely useful for the initial step of the security proof of the UAKE based on KEMs.

SYNTAX. A key encapsulation mechanism $\mathsf{KEM} := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ consist of four polynomial-time algorithms:

- $\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$: The probabilistic setup algorithm Setup takes as input the security parameter κ in unary and returns global system parameters par that implicitly define ciphertext space $\mathcal{C}$, the public key space $\mathcal{PK}$, the secret key space $\mathcal{SK}$ and the key space $\mathcal{K}$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{par})$: The probabilistic key generation algorithm KeyGen takes as input the parameters par and returns a public key $\mathsf{pk} \in \mathcal{PK}$ and a secret key $\mathsf{sk} \in \mathcal{SK}$.
- $(\mathsf{ct}, K) \leftarrow \mathsf{Encaps}(\mathsf{pk})$: The probabilistic encapsulation algorithm Encaps takes as input a public key pk and returns a ciphertext $\mathsf{ct} \in \mathcal{C}$ and a key $K \in \mathcal{K}$.

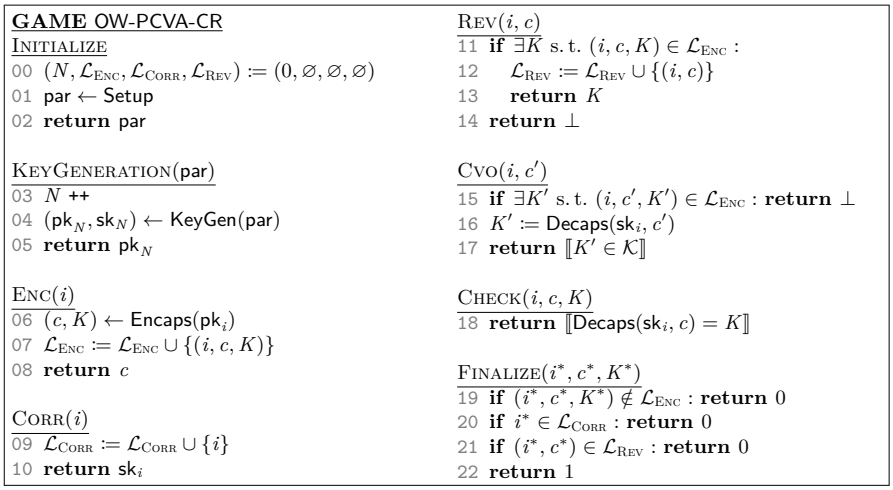

Fig. 14: Game OW-PCVA-CR for KEM. Adversary $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\text{INITIALIZE}, \text{KEYGENERATION}, \text{ENC}, \text{CVO}, \text{REV}, \text{CHECK}, \text{CORR}, \text{FINALIZE}\}$.

– $K := \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$: The deterministic decapsulation algorithm Decaps takes as input a secret key sk and a ciphertext ct and returns a key $K \in \mathcal{K}$.

Definition 10 (Correctness KEM). *We say that* KEM *is ρ-correct, if for any* $\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$ *we have:*

$$\Pr[K = K' \mid (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{par}), (\mathsf{ct}, K) \leftarrow \mathsf{Encaps}(\mathsf{pk}), K' = \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})] \geq \rho.$$

Definition 11 (Min-Entropy of KEM). *We say that* KEM *has min-entropy μ if:*
– *It has key min-entropy $\mu' \geq \mu$: for all $\mathsf{pk}' \in \mathcal{PK}$ we have* $\Pr[\mathsf{pk} = \mathsf{pk}' : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{par})] \leq 2^{-\mu'}$ *for some* par.
– *It has ciphertext min-entropy $\mu'' \geq \mu$: for all $\mathsf{ct}' \in \mathcal{C}$ we have* $\Pr[\mathsf{ct} = \mathsf{ct}' : (\mathsf{ct}, K) \leftarrow \mathsf{Encaps}(\mathsf{pk})] \leq 2^{-\mu''}$ *for some* $\mathsf{pk} \in \mathcal{PK}$.

SECURITY NOTION FOR KEM. We recall the security notion recently used to analyze two-party key exchange with key confirmation from [PRZ24], which is a multi-user version of one-way security with corruptions under plaintext checking and ciphertext validity attacks.

Definition 12. *The game* OW-PCVA-CR *is defined as in Figure 14. The advantage of an adversary $\mathcal{A}$ against* KEM *in this game is defined as*

$$\mathrm{Adv}^{\mathsf{OW\text{-}PCVA\text{-}CR}}_{\mathsf{KEM}}(\mathcal{A}) := \Pr[\mathsf{OW\text{-}PCVA\text{-}CR}^{\mathcal{A}} \Rightarrow 1] \ .$$

UAKE CONSTRUCTION. We construct a UAKE protocol from two KEMs as shown in Figure 15. Let us first prove correctness.

Lemma 3. *Consider the construction in Figure 8. If both* KEM^e *and* KEM^L *have overwhelming correctness of at least $\rho = 1 - 1/2^v$ for some $v \in \Omega(\kappa)$ and the attacker makes q queries overall, then* $\mathsf{UAKE}_{\mathsf{WFS}}$ *has overwhelming correctness at least* $1 - q/2^{v-2}$.

Proof. We give a crude bound. The proof closely follows the proof of the main construction. Assume each of the two KEM schemes has at least overwhelming correctness $\rho = 1 - 1/2^v$ for some $v \in \Omega(\kappa)$. Observe that by setup the KEMs are the only source of incorrectness in the entire protocol construction. Next, observe that if we condition the KEMs to have no correctness error at all, then we will not have a correctness error in $\mathsf{UAKE}_{\mathsf{WFS}}$ as well. So we only have to analyse the influence of the KEMs on the overall correctness. Now, since a single application of a single KEM has overwhelmingly high correctness ρ the probability for both

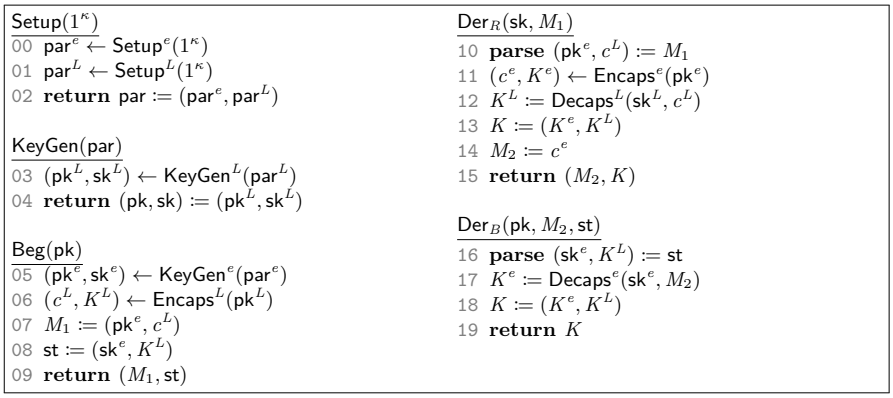

Fig. 15: Generic construction of $\mathsf{UAKE}_{\mathsf{WFS}}$ from KEM^e and KEM^L.

KEMs to simultaneously have no correction error is lower bounded by $\rho' = \rho^2 \geq 1 - 1/2^{v-2}$. A q-time call of the KEMs will thus result in a correctness of $(1 - 1/2^{v-2})^q$. This can be lower bounded via $(1 - 1/2^{v-2})^q \geq (1 - q/2^{v-2})$ for some arbitrary polynomial $q = q(\kappa)$ which shows that the resulting correctness is still overwhelming. □

Theorem 3 (KEM to $\mathsf{UAKE}_{\mathsf{WFS}}$). *For any adversary $\mathcal{A}$ against $\mathsf{OW\text{-}UAKE\text{-}G}_{\mathsf{WFS}}$ with protocol $\mathsf{UAKE}_{\mathsf{WFS}}$ with N parties that establish at most S sessions and issues at most q queries to the oracles, there exist adversaries $\mathcal{B}$ and $\mathcal{C}$ against $\mathsf{OW\text{-}PCVA\text{-}CR}$ security of KEM^e and KEM^L with respectively min-entropy μ^e and μ^L and correctness $(1 - 1/2^v)$ for both, such that*

$$\mathrm{Adv}^{\mathsf{WFS}}_{\mathsf{UAKE}_{\mathsf{WFS}}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{OW\text{-}PCVA\text{-}CR}}_{\mathsf{KEM}^e}(\mathcal{B}) + \mathrm{Adv}^{\mathsf{OW\text{-}PCVA\text{-}CR}}_{\mathsf{KEM}^L}(\mathcal{C}) + \frac{2S^2}{2^{\mu^e}} + \frac{N^2 + S^2}{2^{\mu^L}} + \frac{q}{2^{v-2}} \ ,$$

where the running time of $\mathcal{B}$ and $\mathcal{C}$ is about that of $\mathcal{A}$.

We provide the proof in Appendix B.

6.2 Putting Things Together

The protocol given in Figure 16 is our final GAKE protocol based on KEMs. The idea is the same showed previously in Figure 1. It is a multi-party three-round protocol. We collect the bounds from Theorems 1 to 3 in the following corollary.

Corollary 1 (KEMs to GAKE). *For any adversary $\mathcal{A}$ against $\mathsf{GAKE\text{-}G}_{\mathsf{PFS}}$ with protocol $\mathsf{GAKE}_{\mathsf{PFS}}$ with N parties that establish at most S sessions and issues at most q queries to the oracles, there exist adversaries $\mathcal{B}$ and $\mathcal{C}$ against $\mathsf{OW\text{-}PCVA\text{-}CR}$ security of KEM^e and KEM^L with respectively min-entropy μ^e and μ^L and correctness $(1 - 1/2^v)$ for both, such that*

$$\begin{aligned}\mathrm{Adv}^{\mathsf{PFS}}_{\mathsf{GAKE}_{\mathsf{PFS}}}(\mathcal{A}) \leq\ & \mathrm{Adv}^{\mathsf{OW\text{-}PCVA\text{-}CR}}_{\mathsf{KEM}^e}(\mathcal{B}) + \mathrm{Adv}^{\mathsf{OW\text{-}PCVA\text{-}CR}}_{\mathsf{KEM}^L}(\mathcal{C}) \\ & + \frac{Sq_{RS} + Nq_C + N^2 + 3S^2}{2^\kappa} + \frac{3Sq_{RO} + 2q_{RO}^2 + S + q_{Ch}}{2^\zeta} \\ & + \frac{3(N^2 + S^2)}{2^{\mu^L}} + \frac{2S^2 + Sq_{RO}}{2^{\mu^e}} + \frac{5q + 2S}{2^v} \ ,\end{aligned}$$

where ζ is the lower bound for all dimensions of the random oracle outputs and $q_{RO}, q_{Ch}, q_{RS}, q_C$ are the number of random oracle, check, reveal state and corrupt queries that $\mathcal{A}$ makes. Further, the running time of $\mathcal{B}$ and $\mathcal{C}$ are about that of $\mathcal{A}$.

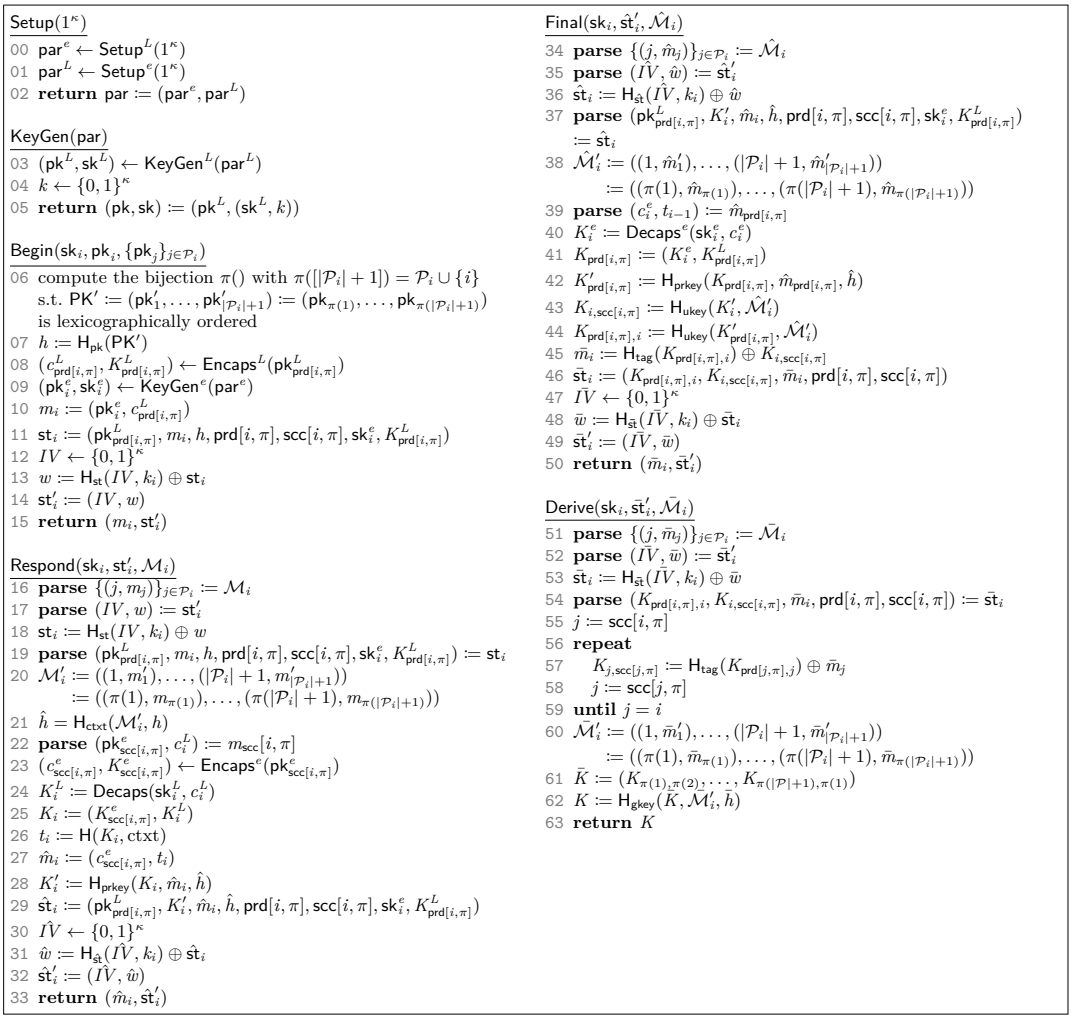

Fig. 16: Generic construction of GAKE from $\mathsf{KEM}^e = (\mathsf{Setup}^e, \mathsf{KeyGen}^e, \mathsf{Encaps}^e, \mathsf{Decaps}^e)$ and $\mathsf{KEM}^L = (\mathsf{Setup}^L, \mathsf{KeyGen}^L, \mathsf{Encaps}^L, \mathsf{Decaps}^L)$. We use the helper function $\mathsf{prd}[i,\pi] := \pi((\pi^{-1}(i)-2 \bmod (|\mathcal{P}_i|+1))+1)$ and $\mathsf{scc}[i,\pi] := \pi((\pi^{-1}(i) \bmod (|\mathcal{P}_i|+1))+1)$ for a bijection $\pi : [|\mathcal{P}_i|+1] \to \mathcal{P}_i \cup \{i\}$. For an intuitive overview see also Figure 1.

Instantiations from DDH and LWE. Let us now show how we can instantiate the underlying KEM with existing constructions. The first implementation is the DDH-based KEM introduced in [JKRS21]. As shown in [PWZ23a, PRZ24] this scheme achieves OW-PCVA-CR security. Ciphertexts consist of two group elements, whereas public keys consist of a single group element. Overall we thus need to transfer two ciphertext and one ephemeral public key in the UAKE protocol with WFS. The UAKE protocol with PFS adds to this a MAC which accounts for a bitstring of size 256 bits. Finally, in the second phase, we exchange symmetric ciphertexts of size 256 bits. In total, this accounts for 5 group elements and 2 strings of size 256 bits.

In comparison, the tightly secure protocol by [PQR22] requires to send 2 group elements for the underlying BD protocol. They use signature schemes over each BD message for authentication. Unfortunately, at the time their paper was published no efficient signature scheme was known that fulfilled the security notion that they require in a tightly secure way. This is why they used Schnorr signatures while providing a proof of tight security *in the generic group model*. However, at the same time, the work in [FJS14] shows that Schnorr signatures cannot provide tight security under any non-interactive security assumption. This

indicates that the results of [PQR22] will lose its tight security guarantees when leaving the GGM model and reducing to non-interactive security assumptions. Based on this instantiation, their protocol accounts for overall 2 group elements and 4 exponents in $\mathbb{Z}_p$ where p is the group order. However, the recent signature scheme in [DGJL21] can now be used as a drop-in for their protocol to obtain a proof under the DDH assumption that does not rely on the generic group model. This signature scheme has signatures that consist of 3 elements in $\mathbb{Z}_p$. When using it in [PQR22] this accounts for 6 elements in $\mathbb{Z}_p$ and 2 group elements. We can now instantiate all schemes in elliptic curve groups with group element representation of around 256 bits.

As a result, we can see that our protocol is more efficient when reducing to non-interactive security assumptions and not relying on GGM proofs. At the same time, we stress that we prove security in a much stronger model that allows the attacker to reveal secret states.

Our second instantiation uses the recent scheme by [PWZ23a] that is based on the LWE assumption. It is thus secure in the PQ-setting. The construction relies on a double encryption approach and is given in Appendix C along with an analysis that shows that its correctness bounds are suitable for our transformation.

On Extending the Results to the QROM. Since the proof for our protocol heavily relies on the ROM, we expect a proof in the quantum random oracle model (QROM) to be more complicated and to not achieve as tight bounds. In the 2-party setting, the first tightly-secure lattice-based AKE was given in the ROM [PWZ23a] which served as a basis for follow-up works [PWZ23b, PRZ24] which improved QROM bounds. However, the latter still have a loss linear in the number of users. We are hopeful that our protocol (without the state-encryption trick) will be able to achieve similar bounds and that any future improvement in the 2-party case is likely to be useful for our setting. Additionally, the currently best tightly-secure instantiation for a KEM in the standard model [HLWG23] is "not quite practical at the moment" due to the NIZK it requires. We therefore believe that a ROM proof is an important first step and consider tight(er) security in the QROM (or standard model) as an important future direction.

Acknowledgements

Emanuele Di Giandomenico and Sven Schäge were supported by the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme in the scope of the CONFIDENTIAL6G project under Grant Agreement 101096435. The contents of this publication are the sole responsibility of the authors and do not in any way reflect the views of the EU. Doreen Riepel was supported in part by Mihir Bellare's KACST grant.

References

ABGS07. Michel Abdalla, Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. (Password) authenticated key establishment: From 2-party to group. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 499–514. Springer, Berlin, Heidelberg, February 2007.

ACDT20. Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. Security analysis and improvements for the IETF MLS standard for group messaging. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 248–277. Springer, Cham, August 2020.

ACJM20. Joël Alwen, Sandro Coretti, Daniel Jost, and Marta Mularczyk. Continuous group key agreement with active security. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 261–290. Springer, Cham, November 2020.

ADGK19. Daniel Apon, Dana Dachman-Soled, Huijing Gong, and Jonathan Katz. Constant-round group key exchange from the ring-LWE assumption. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 189–205. Springer, Cham, 2019.

BD95. Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system (extended abstract). In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 275–286. Springer, Berlin, Heidelberg, May 1995.

BDG+22. Alexander Bienstock, Yevgeniy Dodis, Sanjam Garg, Garrison Grogan, Mohammad Hajiabadi, and Paul Rösler. On the worst-case inefficiency of CGKA. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 213–243. Springer, Cham, November 2022.

BJLS16. Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 273–304. Springer, Berlin, Heidelberg, May 2016.

CC17. Katriel Cohn-Gordon and Cas Cremers. Mind the gap: Where provable security and real-world messaging don't quite meet. Cryptology ePrint Archive, Report 2017/982, 2017.

CCG+18. Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1802–1819. ACM Press, October 2018.

DB05. Ratna Dutta and Rana Barua. Constant round dynamic group key agreement. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC 2005*, volume 3650 of *LNCS*, pages 74–88. Springer, Berlin, Heidelberg, September 2005.

DF17. Yevgeniy Dodis and Dario Fiore. Unilaterally-authenticated key exchange. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 542–560. Springer, Cham, April 2017.

DGJL21. Denis Diemert, Kai Gellert, Tibor Jager, and Lin Lyu. More efficient digital signatures with tight multi-user security. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 1–31. Springer, Cham, May 2021.

DRS24. Emanuele Di Giandomenico, Doreen Riepel, and Sven Schäge. Tightly-secure group key exchange with perfect forward secrecy. In *ASIACRYPT*, LNCS. Springer, Singapore, December 2024.

FJS14. Nils Fleischhacker, Tibor Jager, and Dominique Schröder. On tight security proofs for Schnorr signatures. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 512–531. Springer, Berlin, Heidelberg, December 2014.

GBG09. M. Choudary Gorantla, Colin Boyd, and Juan Manuel González Nieto. Modeling key compromise impersonation attacks on group key exchange protocols. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 105–123. Springer, Berlin, Heidelberg, March 2009.

HJK+21. Shuai Han, Tibor Jager, Eike Kiltz, Shengli Liu, Jiaxin Pan, Doreen Riepel, and Sven Schäge. Authenticated key exchange and signatures with tight security in the standard model. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 670–700, Virtual Event, August 2021. Springer, Cham.

HKSU20. Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. Generic authenticated key exchange in the quantum random oracle model. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 389–422. Springer, Cham, May 2020.

HLWG23. Shuai Han, Shengli Liu, Zhedong Wang, and Dawu Gu. Almost tight multi-user security under adaptive corruptions from LWE in the standard model. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 682–715. Springer, Cham, August 2023.

IY22. Ren Ishibashi and Kazuki Yoneyama. Post-quantum anonymous one-sided authenticated key exchange without random oracles. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 35–65. Springer, Cham, March 2022.

JKRS21. Tibor Jager, Eike Kiltz, Doreen Riepel, and Sven Schäge. Tightly-secure authenticated key exchange, revisited. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 117–146. Springer, Cham, October 2021.

KPPW+21. Karen Klein, Guillermo Pascual-Perez, Michael Walter, Chethan Kamath, Margarita Capretto, Miguel Cueto, Ilia Markov, Michelle Yeo, Joël Alwen, and Krzysztof Pietrzak. Keep the dirt: Tainted TreeKEM, adaptively and actively secure continuous group key agreement. In *2021 IEEE Symposium on Security and Privacy*, pages 268–284. IEEE Computer Society Press, May 2021.

Kra05. Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer, Berlin, Heidelberg, August 2005.

LLM07. Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007*, volume 4784 of *LNCS*, pages 1–16. Springer, Berlin, Heidelberg, November 2007.

MTC13. Ueli Maurer, Björn Tackmann, and Sandro Coretti. Key exchange with unilateral authentication: Composable security definition and modular protocol design. Cryptology ePrint Archive, Report 2013/555, 2013.

MY99. Alain J. Mayer and Moti Yung. Secure protocol transformation via "expansion": From two-party to groups. In Juzar Motiwalla and Gene Tsudik, editors, *ACM CCS 99*, pages 83–92. ACM Press, November 1999.

PQR22. Jiaxin Pan, Chen Qian, and Magnus Ringerud. Signed (group) Diffie-Hellman key exchange with tight security. *Journal of Cryptology*, 35(4):26, October 2022.

PRSS21. Bertram Poettering, Paul Rösler, Jörg Schwenk, and Douglas Stebila. SoK: Game-based security models for group key exchange. In Kenneth G. Paterson, editor, *CT-RSA 2021*, volume 12704 of *LNCS*, pages 148–176. Springer, Cham, May 2021.

PRZ24. Jiaxin Pan, Doreen Riepel, and Runzhi Zeng. Key exchange with tight (full) forward secrecy via key confirmation. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 59–89. Springer, Cham, May 2024.

PW22. Jiaxin Pan and Benedikt Wagner. Lattice-based signatures with tight adaptive corruptions and more. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 347–378. Springer, Cham, March 2022.

PWZ23a. Jiaxin Pan, Benedikt Wagner, and Runzhi Zeng. Lattice-based authenticated key exchange with tight security. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 616–647. Springer, Cham, August 2023.

PWZ23b. Jiaxin Pan, Benedikt Wagner, and Runzhi Zeng. Tighter security for generic authenticated key exchange in the QROM. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part IV*, volume 14441 of *LNCS*, pages 401–433. Springer, Singapore, December 2023.

Sho94. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

Sho04. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004.

A Proof of Theorem 1

Proof. Let $\mathcal{A}$ be an adversary against $\mathsf{UAKE}_{\mathsf{PFS}}$. We consider a sequence of games G_0 to G_3, as shown in Figure 17.

Game G_0. This is the original $\mathsf{OW\text{-}UAKE\text{-}G}_{\mathsf{PFS}}$ game, but we also implicitly exclude collisions. That is, if the same public key or message is chosen twice, the experiment aborts. Using the fact that UAKE has min-entropy μ, the upper bound for key collisions is $N^2/2^\mu$ and for message collisions, it is $S^2/2^\mu$. In the end, we also aim for random oracle outputs to be unique. Assuming ζ is the lower bound for the dimensions of the tag space and key space,

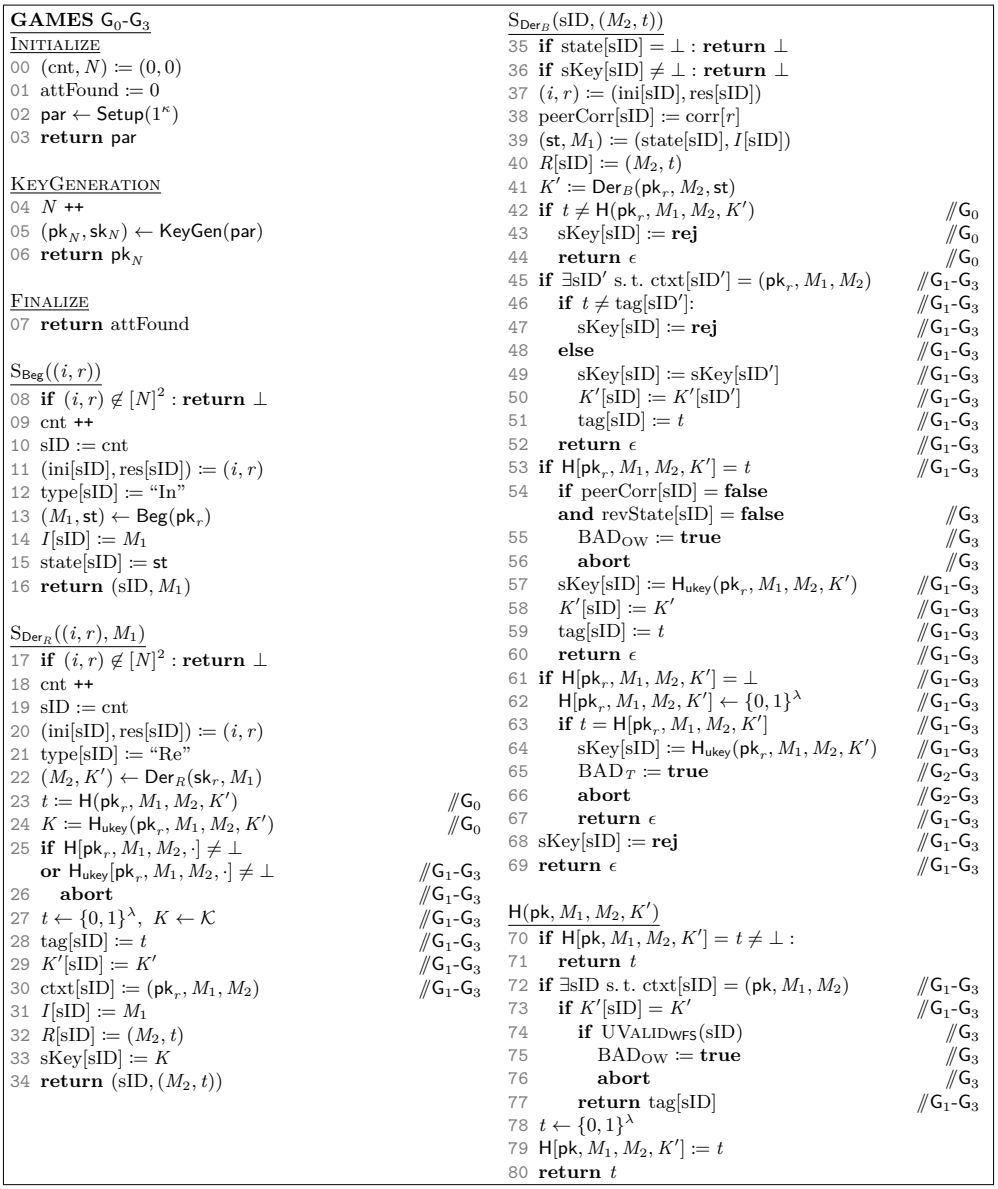

Fig. 17: Games G_0-G_3 for the proof of Theorem 1. $\mathcal{A}$ has access to oracles $\mathrm{O} \coloneqq \{\mathrm{S}_{\mathsf{Beg}}, \mathrm{S}_{\mathsf{Der}_R}, \mathrm{S}_{\mathsf{Der}_B}, \text{REV-STATE}, \text{CORRUPT}, \text{CHECK}, \mathsf{H}, \mathsf{H}_{\mathsf{ukey}}\}$, where $\mathsf{H}_{\mathsf{ukey}}$ is simulated as H, using variable sKey[sID] instead. Further, oracles REV-STATE, CORRUPT and CHECK are as in Figure 4.

collisions are excluded with a probability of at most $q_{RO}^2/2^\zeta$, where q_{RO} is the number of random oracle queries. All the probabilities follow from the birthday bound. Hence, we have

$$\mathrm{Adv}^{\mathsf{PFS}}_{\mathsf{UAKE}_{\mathsf{PFS}}}(\mathcal{A}) \leq \Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] + \frac{N^2 + S^2}{2^\mu} + \frac{q_{RO}^2}{2^\zeta} \ .$$

GAME G_1. Game G_1 aborts when $\mathrm{S}_{\mathsf{Der}_R}$ computes a message M_2 and there already exists a random oracle entry with $(\mathsf{pk}_r, M_1, M_2, \cdot)$, where pk_r is the session holder's public key and M_1 is the message given as input to $\mathrm{S}_{\mathsf{Der}_R}$. Since UAKE has at least min-entropy μ, the probability that the game aborts can be upper bounded by $S \cdot q_{RO} \cdot 2^{-\mu}$ via a union bound over all sessions. If the game does not abort, the tag and key are chosen uniformly at random and stored in variables tag[sID] and sKey[sID]. We also store the intermediate UAKE key K' as

well as the context $(\mathsf{pk}_r, M_1, M_2)$ in additional variables $K'[\text{sID}]$ and ctxt[sID], respectively. These are used to simulate the random oracles and oracle $\mathrm{S}_{\mathsf{Der}_B}$ correctly. The latter changes are not noticeable to $\mathcal{A}$ unless a correctness error occurs. Hence, we get

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| \leq \frac{Sq_{RO}}{2^{\mu}} + \frac{S}{2^{v}} \ .$$

Game G_2. This game raises flag BAD_T and aborts when the adversary guesses a tag that is accepted by $\mathrm{S}_{\mathsf{Der}_B}$, but it has not queried the random oracle before. Such a tag will only be accepted with small statistical probability, namely, $2^{-\lambda}$ for each session. Hence,

$$\left|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]\right| \leq \frac{S}{2^{\lambda}} \ .$$

Game G_3. In game G_3, we raise a flag $\mathrm{BAD}_{\mathrm{OW}}$ when the adversary manages to forge a tag for a valid session (which means it must have queried the correct key to H) or when it queries $\mathsf{H}_{\mathsf{ukey}}$ on the underlying key of $\mathsf{UAKE}_{\mathsf{WFS}}$. In those cases, the game aborts. We have

$$\left|\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1]\right| \leq \Pr[\mathrm{BAD}_{\mathrm{OW}}] \ .$$

To bound $\mathrm{BAD}_{\mathrm{OW}}$, we will construct a reduction $\mathcal{B}$ and show that

$$\Pr[\mathrm{BAD}_{\mathrm{OW}}] \leq \mathrm{Adv}^{\mathsf{WFS}}_{\mathsf{UAKE}_{\mathsf{WFS}}}(\mathcal{B}) + \frac{S}{2^{v}} \ .$$

Our reduction $\mathcal{B}$ is given in Figure 18. Since $\mathcal{B}$ has access to almost the same interfaces as $\mathcal{A}$, oracles KeyGeneration, $\mathrm{S}_{\mathsf{Beg}}$, $\mathrm{S}_{\mathsf{Der}_R}$, Rev-State and Corrupt can be simulated in a straightforward way using $\mathcal{B}$'s own oracles. Care needs to be taken when simulating the initiator's derivation oracle and the random oracles which is where $\mathcal{B}$ wants to extract the solution to its UAKE game. Whenever the game G_3 performs an equivalence check of the underlying UAKE keys with the query provided by $\mathcal{A}$, $\mathcal{B}$ uses its Check oracle. It also checks (as in G_3) whether the session in question is still valid and if both these checks succeed, then this means that $\mathcal{A}$ has found the underlying UAKE key for a valid session. Hence, $\mathcal{B}$ directly stops and calls its finalize oracle.

Finally, note that in G_3, the adversary can only have guessed a correct session key. Since the adversary must specify sID for each query to Check and all sessions' contexts are distinct, each query can only be successful with probability $1/|\mathcal{K}|$. Union bound over all queries gives us

$$\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] \leq \frac{q_{Ch}}{|\mathcal{K}|} \ ,$$

which completes the proof. □

B Proof of Theorem 3

Proof. Let $\mathcal{A}$ be an adversary against protocol $\mathsf{UAKE}_{\mathsf{WFS}}$. We consider the games in Figure 19.

Game G_0. This is the same as $\mathsf{OW\text{-}UAKE\text{-}G}_{\mathsf{WFS}}$, except for small changes. We implicitly exclude collisions (if a collision happens at any time in the game, it aborts). We assume that both ephemeral and long-term key pairs and ciphertexts are distinct. Using the fact that KEM^e and KEM^L have min-entropy μ^e and μ^L respectively and due the birthday bound, the upper bound for ephemeral key collisions and for ephemeral ciphertext collisions is $2S^2/2^{\mu^L}$, and for long-term key collisions and for long-term ciphertext collisions is $(N^2 + S^2)/2^{\mu^L}$. Finally, in this step we abort if any of the KEM runs of the challenger do not feature correctness. However, as stated in Lemma 3, this only happens with probability $q/2^{v-2}$.

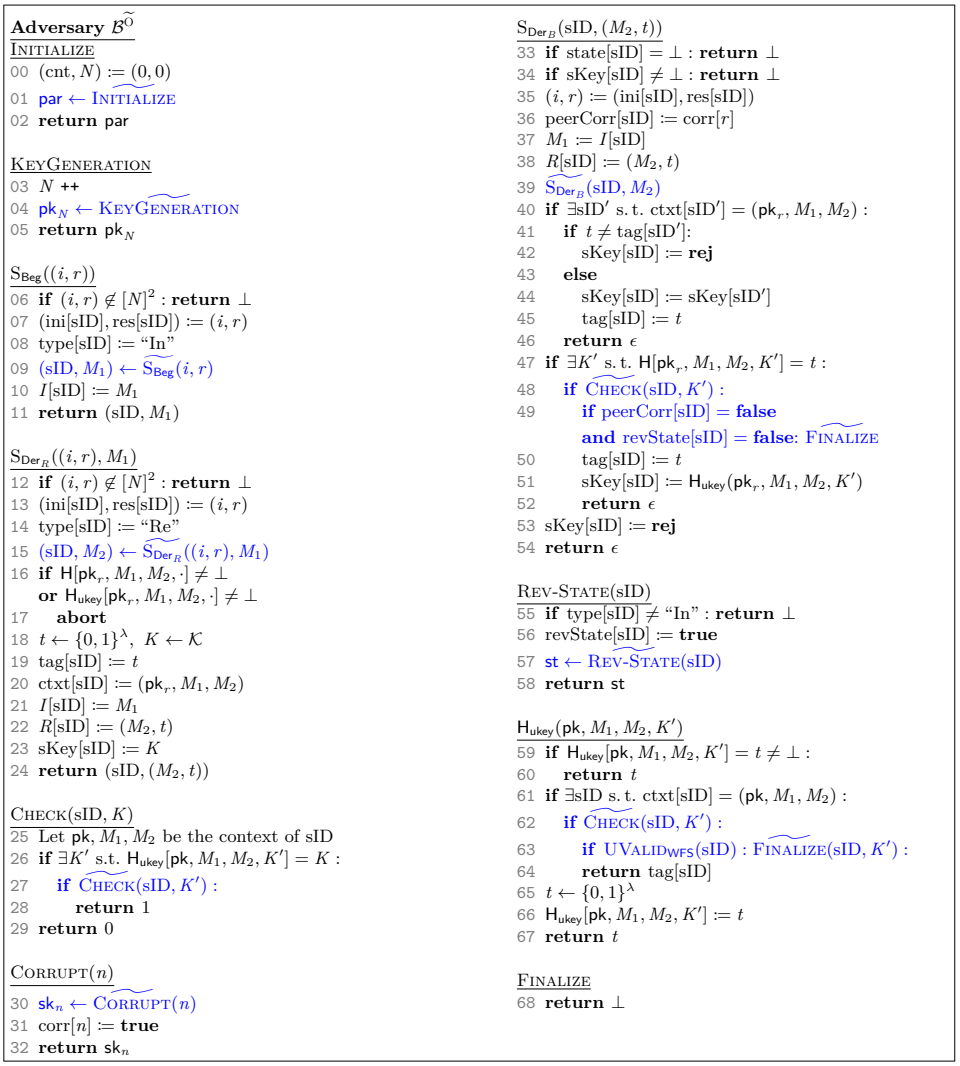

Fig. 18: Adversary $\mathcal{B}$ for the proof of Theorem 1. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\text{Initialize}, \mathrm{S}_{\mathsf{Beg}}, \mathrm{S}_{\mathsf{Der}_R}, \mathrm{S}_{\mathsf{Der}_B}, \text{Rev-State}, \text{Corrupt}, \text{Check}, \mathsf{H}, \mathsf{H}_{\mathsf{ukey}}, \text{Finalize}\}$, where $\mathsf{H}_{\mathsf{ukey}}$ is simulated as H.

We get

$$\left|\Pr[\mathsf{OW\text{-}UAKE\text{-}G}^{\mathcal{A}}_{\mathsf{WFS}} \Rightarrow 1] - \Pr[\mathsf{G}^{\mathcal{A}}_0 \Rightarrow 1]\right| \leq \frac{2S^2}{2^{\mu^e}} + \frac{N^2+S^2}{2^{\mu^L}} + \frac{q}{2^{v-2}} \ .$$

Game G_1. Now, we raise flag BAD^e in line 55 (Fig. 19) if a session sID and a session key K are issued to the check oracle Check with a valid attack. That means, if there exists a (partially) matching session sID′ to session sID and if sID is an initiator type "In" session and its state is not revealed, or if sID is a responder type "Re" session and the state of sID′ is not revealed, then we abort. To upper bound the probability that BAD^e is raised, we design an adversary $\mathcal{B}$ (Fig. 20) against the $\mathsf{OW\text{-}PCVA\text{-}CR}$ of the underlying KEM^e. We have to track the new ephemeral public keys since they are related to initiator type sessions, so we introduce a new counter m that we store in the new variable kgcnt. Then, the adversary simulate every ephemeral outputs properly and if in the check oracle Check it issued a session sID and a session key K such that there exists a (partially) matching session, the

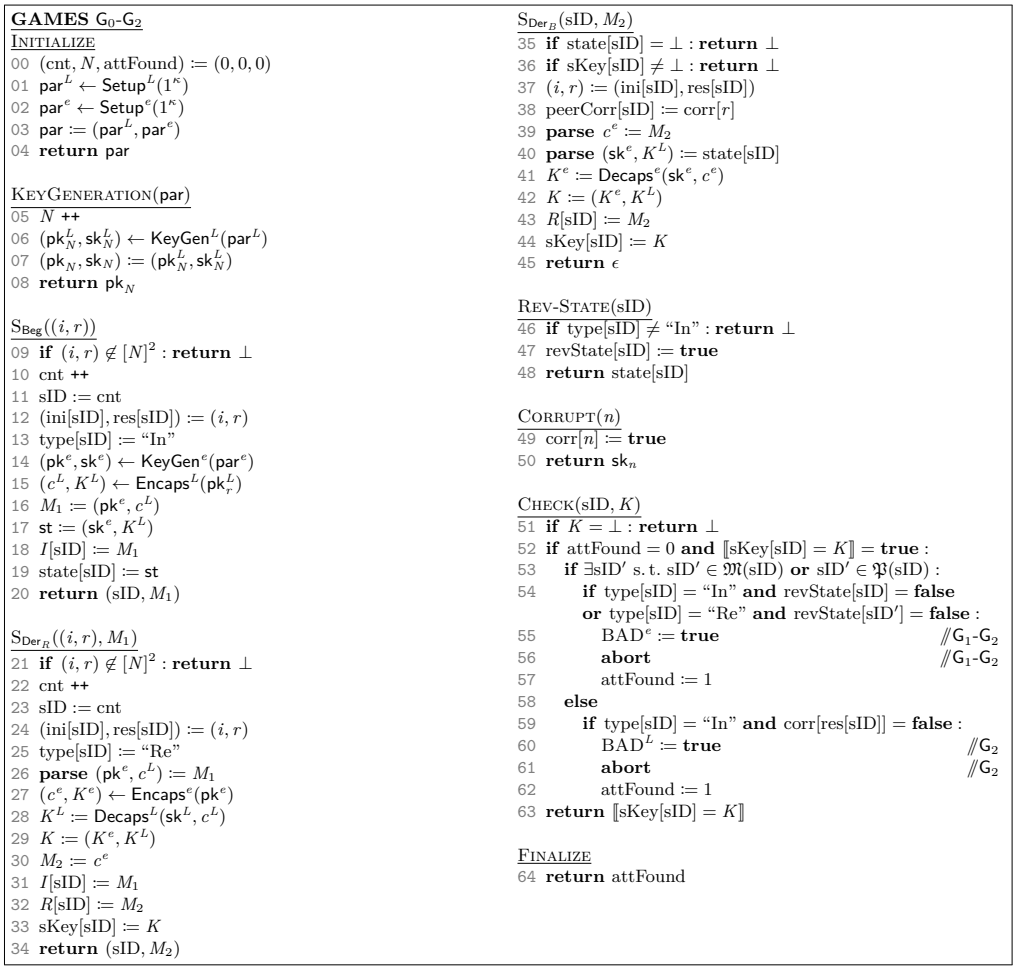

Fig. 19: Games G_0-G_2 for the proof of Theorem 3. Adversary $\mathcal{A}$ has access to oracles $\mathrm{O} \coloneqq \{\text{INITIALIZE}, \mathrm{S}_{\mathsf{Beg}}, \mathrm{S}_{\mathrm{Der}_R}, \mathrm{S}_{\mathrm{Der}_B}, \text{REV-STATE}, \text{CORRUPT}, \text{CHECK}, \text{FINALIZE}\}$.

state is not revealed and the underlying ephemeral check oracle $\widetilde{\text{CHECK}}$ output true (which is exactly the bad event BAD^e), then it won since it found a valid attack.

We get

$$\left|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1]\right| \leq \Pr[\text{BAD}^e] \leq \text{Adv}^{\mathsf{OW\text{-}PCVA\text{-}CR}}_{\mathsf{KEM}^e}(\mathcal{B}) \ .$$

GAME G_2. This game is similar to the previous one. We raise flag BAD^L in line 60 (Fig. 19) if a session sID and a session key K are issued to the check oracle CHECK with a valid attack. That means, if there exists a (partially) matching session sID' to an initiator type "In" session sID and its responder is not corrupted, then we abort. To upper bound the probability that BAD^L is raised, we design an adversary $\mathcal{C}$ (Fig. 21) against the $\mathsf{OW\text{-}PCVA\text{-}CR}$ of the underlying KEM^L. As the previous game, the adversary simulate every long-term outputs properly and if in the check oracle CHECK it issued a session sID and a session key K such that there exists a (partially) matching session, the session sID is an initiator type session, the responder is not corrupted and the underlying long-term check oracle $\widetilde{\text{CHECK}}$ output true (which is exactly the bad event BAD^L), then it won since it found a valid attack.

We get

$$\left|\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| \leq \Pr[\text{BAD}^L] \leq \text{Adv}^{\mathsf{OW\text{-}PCVA\text{-}CR}}_{\mathsf{KEM}^L}(\mathcal{C}) \ .$$

Fig. 20: Adversary $\mathcal{B}$ against OW-PCVA-CR for KEM^e for the proof of Theorem 3. It has access to oracles $\widetilde{O} := \{$INITIALIZE, KEYGENERATION, ENC, DEC, CORR, REV, CHECK, FINALIZE$\}$.

Finally, it is easy to see that attFound cannot be ever set to 1. Then,

$$\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] = 0 \ .$$

Combining all the probabilities, we finally obtain the bound stated in Theorem 3. □

C The $\mathsf{KEM}_{\mathsf{LWE}}$

We introduce the $\mathsf{KEM}_{\mathsf{LWE}}$ described in [PWZ23a]. The $\mathsf{KEM}_{\mathsf{LWE}}$ is a tightly OW-ChCCA secure under the LWE assumption key encapsulation mechanism. The scheme is described in Figure 22. It utilizes algorithms SampleD and Round:

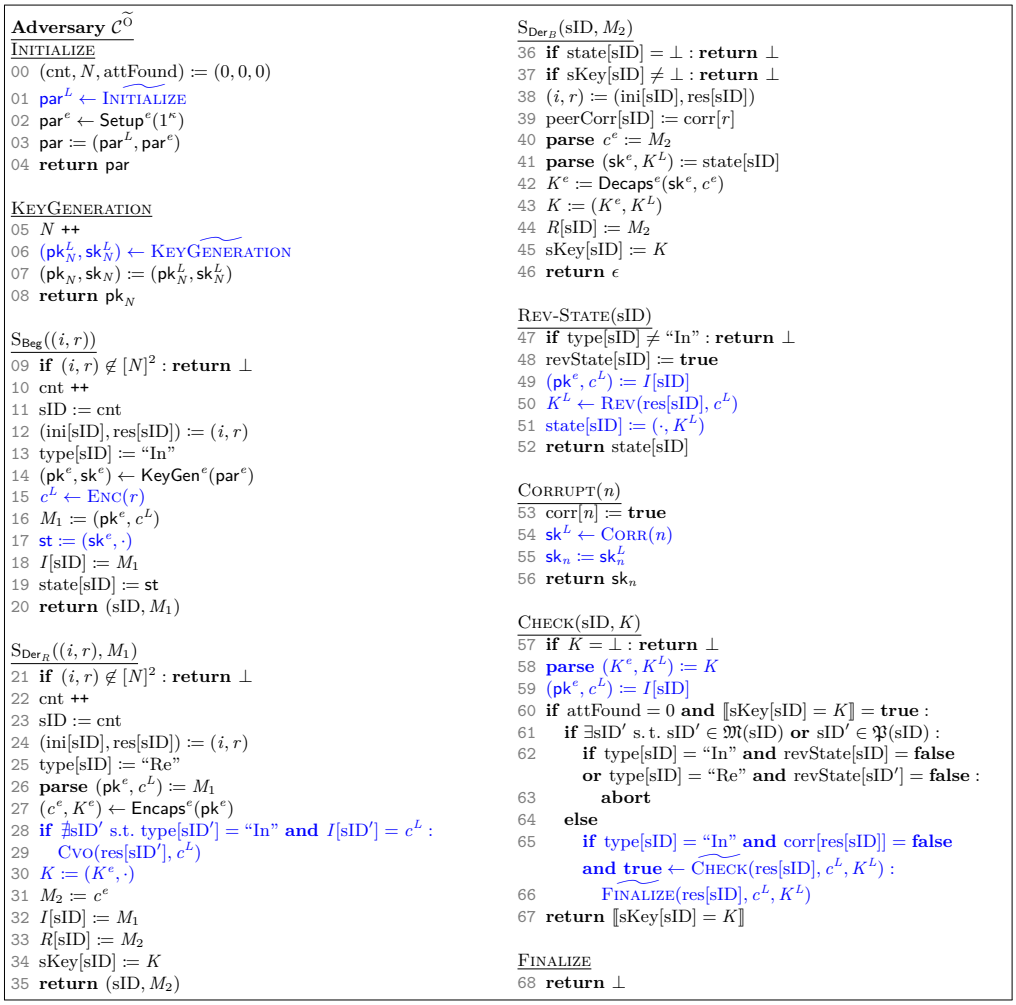

Fig. 21: Adversary $\mathcal{C}$ against OW-PCVA-CR for KEM^L for the proof of Theorem 3. It has access to oracles $\widetilde{\mathrm{O}} := \{$INITIALIZE, KEYGENERATION, ENC, DEC, CORR, REV, CHECK, FINALIZE$\}$.

- $\mathbf{e} \leftarrow \mathsf{SampleD}(m, \alpha'; \rho)$: Sample Gaussian $\mathbf{e} \leftarrow D^m_{\mathbb{Z},\alpha'}$ using random coins $\rho \in \{0,1\}^\kappa$.
- $\mathbf{h} \leftarrow \mathsf{Round}(\mathbf{t})$: Do component-wise rounding of $\mathbf{t} \in \mathbb{Z}_q^\kappa$ to get $\mathbf{h} \in \{0,1\}^\kappa$, i.e. for all $i \in [\kappa]$, $\mathbf{h}_i = 0$ is closer to 0 than to $\lfloor q/2 \rceil$, and $\mathbf{h}_i = 1$ otherwise.

The scheme is parameterized by matrix dimensions $n, m, k \in \mathbb{N}$, a modulus $q \in \mathbb{N}$, and (Gaussian) widths $\alpha, \alpha', \gamma, \eta > 0$.

$\mathsf{KEM}_{\mathsf{LWE}}$ exhibits deterministic ciphertext derivation. The strategy follows a classical double encryption approach where one of the secret keys is always known to the reduction. In this way, the reduction does not have to rely on any non-tight guessing arguments. Accordingly, the ciphertext is split into two parts, each potentially recoverable using a corresponding secret key. The use of LWE is employed to modify the setup parameters, all other modifications account only for statistically small changes in the success probability of the attacker. Overall yields a tight proof. This strategy, although feasible for providing a corruption oracle, becomes more intricate when simulating a decapsulation oracle. The challenge is to ensure that the simulated decapsulation does not reveal crucial information. This is addressed by deterministically deriving ciphertext parts from the encapsulated key and implementing deterministic functions for consistency checks. Another challenge arises from the reveal oracle, which requires ensuring consistency in ciphertexts once revealed. This

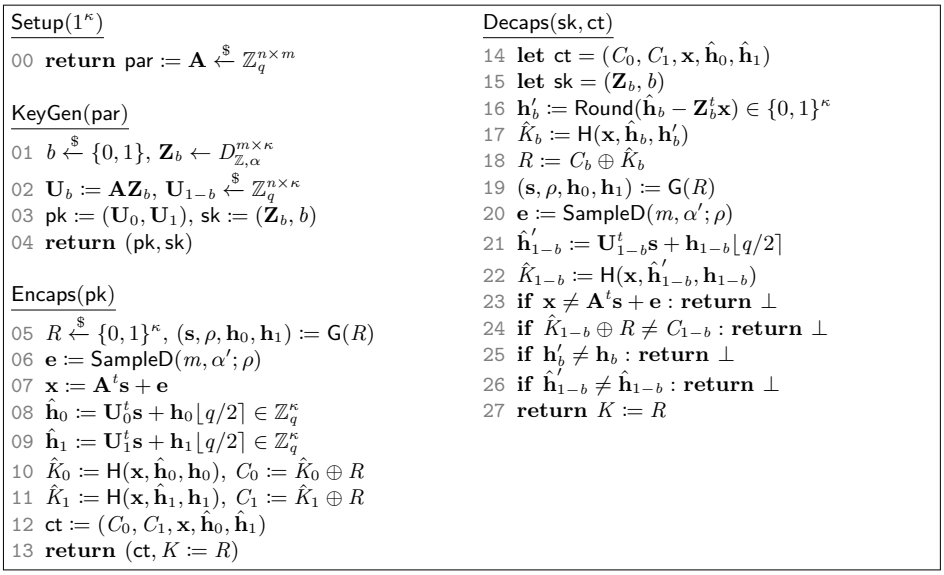

Fig. 22: The key encapsulation mechanism $\mathsf{KEM}_{\mathsf{LWE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$, where $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\kappa$ and $\mathsf{G} : \{0,1\}^* \to [-\eta,\eta]^n \times \{0,1\}^\kappa \times \{0,1\}^\kappa \times \{0,1\}^\kappa$ are random oracles.

is achieved through careful utilization of a random oracle to make inconsistent ciphertexts consistent again.

Re-Evaluating Correctness. For our purpose, it is important to analyse the correctness of the scheme in more detail. The correctness is given by the authors as $1-\epsilon$ where $\epsilon = \epsilon(\kappa)$ is a negligible function. However, for our results to hold, we need that ϵ is statistically small. Luckily, the scheme does actually guarantee that ϵ is indeed statistically small. To see this observe that in their proof of correctness, the authors need to bound two quantities in size $\|\mathbf{z}_{b,i}^t\|$ and $\|\mathbf{e}\|$, where $\mathbf{z}_{b,i}^t$ is column i of $\mathbf{Z}_b$ for $i \in [\kappa]$, to conclude that their product is bounded accordingly. As the authors already observe (Appendix B of [PWZ23a]) the bounds on these two values hold unconditionally except for statistically low probability 2^{-w+1} where $w = O(\kappa\log(\kappa))$. However, this analysis only holds for a single i. The probability that these bounds are met for all $i \in [\kappa]$ can be bounded by $(1-2^{-w+1})^\kappa \geq (1-\kappa/2^{w+1})$ by Bernouilli's inequality. Since $w = O(\kappa\log(\kappa))$, the probability $(1-\kappa/2^{w+1})$ is statistically close to uniform again.

Security. For completeness we also provide the main theorem of [PWZ23a] who prove security w.r.t. their security notion OW-ChCCA security. Since OW-PCVA-CR is a relaxation of that notion [PRZ24], we get the following lemma.

Lemma 4 (LWE to KEM [PWZ23a,PRZ24]). *For any adversary $\mathcal{A}$ against* OW-PCVA-CR *with protocol* KEM*, there exists an adversary $\mathcal{B}$ against $\mathsf{LWE}_{k,m,q,D_{\mathbb{Z},\gamma}}$ assumption, such that*

$$\mathrm{Adv}^{\mathsf{OW\text{-}PCVA\text{-}CR}}_{\mathsf{KEM}_{\mathsf{LWE}}}(\mathcal{A}) \leq 6n\mathrm{Adv}^{\mathsf{LWE}_{k,m,q,D_{\mathbb{Z},\gamma}}}(\mathcal{B}) + \mathsf{negl}(\kappa) \ ,$$

where the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.