# Multi-User CDH Problems and the Concrete Security of NAXOS and HMQV

Eike Kiltz[1], Jiaxin Pan[2], Doreen Riepel[1,3], and Magnus Ringerud[2]

[1] Ruhr-Universität Bochum, Bochum, Germany
{eike.kiltz,doreen.riepel}@rub.de
[2] NTNU – Norwegian University of Science and Technology, Trondheim, Norway
{jiaxin.pan,magnus.ringerud}@ntnu.no
[3] University of California San Diego, San Diego, USA

**Abstract.** We introduce CorrGapCDH, the Gap Computational Diffie-Hellman problem in the multi-user setting with Corruptions. In the random oracle model, our assumption *tightly* implies the security of the authenticated key exchange protocols NAXOS in the eCK model and (a simplified version of) X3DH without ephemeral key reveal. We prove hardness of CorrGapCDH in the generic group model, with *optimal bounds* matching the one of the discrete logarithm problem.

We also introduce CorrCRGapCDH, a stronger Challenge-Response variant of our assumption. Unlike standard GapCDH, CorrCRGapCDH implies the security of the popular AKE protocol HMQV in the eCK model, *tightly* and *without rewinding*. Again, we prove hardness of CorrCRGapCDH in the generic group model, with (almost) optimal bounds.

Our new results allow implementations of NAXOS, X3DH, and HMQV without having to adapt the group sizes to account for the tightness loss of previous reductions. As a side result of independent interest, we also obtain modular and simple security proofs from standard GapCDH with tightness loss, improving previously known bounds.

**Keywords:** Authenticated key exchange, HMQV, NAXOS, X3DH, generic hardness.

## 1 Introduction

Authenticated key exchange (AKE) is a fundamental cryptographic protocol where two users agree on a joint session key. In a simple and efficient blueprint of Diffie-Hellman protocols, Alice (holding long-term key $g^a$) sends a random ephemeral key $g^x$ to Bob; Bob (holding long-term key $g^b$) sends a random ephemeral key $g^y$ to Alice. After receiving their input, both users derive the joint session key $K$ from the four Diffie-Hellman values $g^{ab}, g^{ay}, g^{xy}, g^{bx}$. The practically relevant protocols HMQV [Kra05], NAXOS [LLM07], and X3DH$^-$ [CCG+19] (a simplification of Extended Triple Diffie-Hellman X3DH [MP16]) fall into this class of Diffie-Hellman protocols, see Figure 1. They are all two message protocols with implicit authentication, namely, only the designated users can share the same key and together with a MAC they can confirm their session keys and authenticate each other explicitly.

We highlight that HMQV is the well-known "provably secure" variant of MQV [MQV95, LMQ+03] which is included in the IEEE P1363 standard for key exchange [P1300]. X3DH$^-$ is essentially the Extended Triple Diffie-Hellman (X3DH) key exchange protocol without involving any signature and ignoring the server. The original X3DH protocol is used for the initial key exchange in Signal, where the receiver publishes (signed) prekeys on a server which can be retrieved (asynchronously) by the sender. The NAXOS protocol is X3DH$^-$ combined with the "NAXOS hashing trick" which is marked with a dashed box in Figure 1.

AKE Security Model. Adversaries against AKE protocols can control all messages transferred among involved users, and they can also reveal some of the shared session keys and the long-term secret keys of honest users. These capabilities are captured by security models such as [BR94, CK01, LLM07]. The goal of an adversary is to distinguish a non-revealed session key from a random key of the same length. We use the extended Canetti-Krawczyk (eCK) model [BR94, CK01, LLM07] in a game-based formulation of [JKRS21] that allows adversaries to register dishonest users, corrupt long-term secret keys of the $N \geq 2$ honest users, reveal ephemeral states and session keys of the $S$ sessions. The adversary is allowed to make $T$ test queries based on the same random bit $b$. It captures weak forward secrecy (which is the

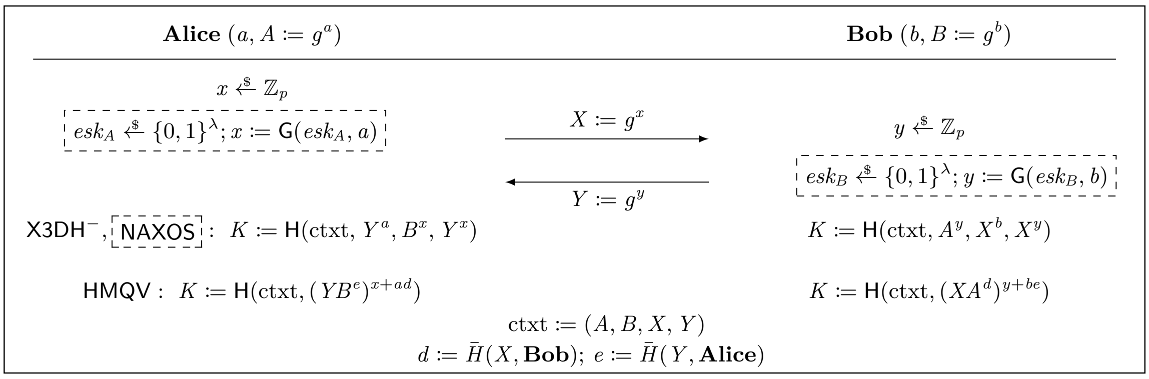
**Alice** $(a, A := g^a)$
**Bob** $(b, B := g^b)$
$x \xleftarrow{\$} \mathbb{Z}_p$
$esk_A \xleftarrow{\$} \{0,1\}^\lambda; x := \mathsf{G}(esk_A, a)$
$X := g^x$
$y \xleftarrow{\$} \mathbb{Z}_p$
$esk_B \xleftarrow{\$} \{0,1\}^\lambda; y := \mathsf{G}(esk_B, b)$
$Y := g^y$
X3DH$^-$, NAXOS : $K := \mathsf{H}(\text{ctxt}, Y^a, B^x, Y^x)$
$K := \mathsf{H}(\text{ctxt}, A^y, X^b, X^y)$
HMQV : $K := \mathsf{H}(\text{ctxt}, (YB^e)^{x+ad})$
$K := \mathsf{H}(\text{ctxt}, (XA^d)^{y+be})$
$\text{ctxt} := (A, B, X, Y)$
$d := \bar{H}(X, \textbf{Bob}); e := \bar{H}(Y, \textbf{Alice})$

**Fig. 1.** Overview of different AKE protocols, HMQV, X3DH$^-$, and NAXOS. NAXOS computes exponents $x$ and $y$ as shown in the dashed box. We make a small twist to HMQV that includes the context ctxt in computing the session key $K$. This twist is to avoid the trivial winning of an adversary in the eCK model (see Section 6) and is also applied in the analysis of [BCLS15].

strongest forward secrecy a two-pass implicit AKE protocol can achieve [Kra05]) and security against key-compromise impersonation (KCI) attacks and reflection attacks. We stress that our model is using a single challenge bit and hence allows for tight composition of the AKE with symmetric primitives [CCG$^+$19].

Tightness. The security of AKE protocols is usually established by a security reduction. More precisely, for any adversary $\mathcal{A}$ against an AKE protocol with success probability $\varepsilon^{\mathsf{AKE}}$, there exists an adversary $\mathcal{B}$ with roughly the same running time that breaks the underlying assumption with probability $\varepsilon^{\mathsf{Ass}} = \varepsilon^{\mathsf{AKE}}/\ell$. The security loss $\ell$ plays an important role in choosing the system parameters. If $\ell$ is large, one has to increase the size of the underlying group $\mathbb{G}$ to account for the security loss. Optimally, $\ell$ is a small constant in which case we call the reduction *tight*.

Security proofs for AKE protocols are rather complex and the resulting bounds are highly non-tight [Kra05,LLM07,CCG$^+$19,Ust08,SE16,PW11]. A reduction $\mathcal{B}$ usually makes several case distinctions and, by guessing the behavior of an adversary in each case, $\mathcal{B}$ embeds a problem instance into either the protocol transcripts or the users' public keys. In the end, this guessing strategy ends up with a large security loss. Most of the AKE protocols lose a linear (or even quadratic) factor in the number of users $N$, the number of sessions $S$, and the number of test sessions $T$. Even worse, HMQV and its variants (such as [Ust08, SE16, PW11]) additionally require the Forking Lemma [PS00] to rewind the adversary and bound its success probability, which ends up with an even larger security loss. X3DH$^-$ is a noteworthy exception because it loses only a linear factor in $N$ [CCG$^+$19]. This linear loss in $N$ is shown to be optimal for a large class of Diffie-Hellman protocols [CCG$^+$19], including our simple blueprint of Diffie-Hellman protocols.

## 1.1 Our Contributions

In this paper, we simplify the difficulty of proving AKE protocols by introducing new variants of the Computational Diffie-Hellman (CDH) problem in the multi-user setting:

– We introduce $n$-CorrGapCDH, the Gap Computational Diffie-Hellman problem in an $n$-user setting with Corruptions. The hardness of $(N + S)$-CorrGapCDH *tightly* implies the security of NAXOS and X3DH$^-$.
– We introduce $(n, Q_{\mathrm{CH}})$-CorrCRGapCDH, a stronger Challenge-Response variant of $n$-CorrGapCDH. The hardness of $(N + S, Q_{\mathrm{RO}})$-CorrCRGapCDH *tightly* implies the security of HMQV without rewinding.

Recall that in the eCK model the variables $N$, $S$, $T$, and $Q_{\mathrm{RO}}$ correspond to the number of users, sessions, test queries, and random oracle queries, respectively. For NAXOS and HMQV, we prove security with state corruptions. For X3DH$^-$, state corruption is not allowed, since it will lead to a trivial attack.

We prove our new assumptions based on the Gap Diffie-Hellman (GapCDH) assumption [OP01, ABR01] via non-tight reductions. Combined with these non-tight reductions, we give simple, intuitive and modular security proofs of X3DH$^-$, NAXOS and HMQV. For NAXOS and HMQV, we obtain tighter

|  | wFS | St | Security tightly implied by | Security loss wrt. GapCDH Old | New |
|---|---|---|---|---|---|
| NAXOS | $\surd$ | $\surd$ | $(N+S)$-CorrGapCDH | $T(N+S)^2$ | $(N+S)^2$ |
| X3DH$^-$ | $\surd$ | – | $(N+S,N)$-CorrAGapCDH | $N$ | $N$ |
| HMQV | $\surd$ | $\surd$ | $(N+S,Q_{\mathrm{RO}})$-CorrCRGapCDH | $Q_{\mathrm{RO}}\,T(N+S)^2$ | $Q_{\mathrm{RO}}(N+S)^2$ |

**Fig. 2.** Security of the AKE protocols NAXOS, X3DH$^-$, and HMQV in the eCK model. St stands for state reveal attacks and wFS stands for weak forward secrecy. The "Security tightly implied by" column names the *new multi-user problem* which tightly implies the AKE's security. The last two columns contain old and new security loss for the AKE protocols relative to the *standard* GapCDH *problem*, ignoring constants. HMQV additionally incorporates the $\sqrt{\varepsilon^{\mathsf{GapCDH}}}$ loss due to the Forking Lemma.

security bounds, and for X3DH$^-$ we match the optimal bound from [CCG+19]. Our results in the random oracle model are summarized in Figure 2.[4]

The main novelty of our new multi-user CDH assumptions lies in their practical applicability. We show the quantitative hardness of CorrGapCDH in the Generic Group Model (GGM) [Sho97, Mau05], which is optimal and matches the one of the discrete logarithm problem. We also prove the hardness of CorrCRGapCDH in the GGM and it is (almost) optimal. Our new results in the GGM support the implementation of NAXOS, X3DH$^-$, and HMQV without increasing the group sizes to compensate the security loss of the previous reductions. Our results in the generic group model are summarized in Figure 3 on page 5.

## 1.2 Multi-User CDH with Corruptions

Let $\mathsf{par} = (p, g, \mathbb{G})$ be system parameters that describe a group $\mathbb{G}$ of prime order $p = |\mathbb{G}|$ and a generator $g$ of $\mathbb{G}$. Given $g^{a_1}, g^{a_2}$, the standard GapCDH problem (over par) requires to compute the Diffie-Hellman key $g^{a_1 a_2}$ [OP01, ABR01]. Here Gap stands for the presence of a (decisional) Gap Oracle which on input $(X = g^x, Y = g^y, Z = g^z)$ returns 1 iff $xy = z \bmod p$. We now describe our new assumptions in more details. Formal definitions will be given in Section 3.

MULTI-USER GapCDH WITH CORRUPTIONS. For $n \geq 2$, the $n$-user GapCDH problem with Corruptions ($n$-CorrGapCDH) is a natural generalization of GapCDH to the $n$-user setting. The adversary is given the $n$-tuple $(g^{a_1}, \dots, g^{a_n})$ and is allowed to corrupt any user $i$ to obtain its secret $a_i$. In order to win, it must output any of the $n(n-1)$ possible Diffie-Hellman keys $g^{a_i a_j}$ for two non-corrupted users $i \neq j$. Even though the two assumptions are asymptotically equivalent, they are quantitatively different: Due to the corruptions, one can only prove the non-tight bound $\varepsilon^{\mathsf{CorrGapCDH}} \leq O(n^2) \cdot \varepsilon^{\mathsf{GapCDH}}$.

For $n_1 \leq n$, we also consider an Asymmetric version of this assumption called $(n, n_1)$-CorrAGapCDH. It is asymmetric in the sense that $n_1$ splits the set of users $[n]$ in two disjoint sets $[n_1]$ and $[n_1 + 1, n]$, where only the first $n_1$ users can be corrupted. The adversary has to output any of the Diffie-Hellman keys $g^{a_i a_j}$ for two non-corrupted users $i \in [n_1]$ and $j \in [n_1 + 1, n]$. Note that CorrGapCDH tightly implies CorrAGapCDH. However, the fact that the challenge set is split asymmetrically allows us to give a tighter relation to GapCDH. In particular, we prove that $\varepsilon^{\mathsf{CorrAGapCDH}} \leq O(n_1) \cdot \varepsilon^{\mathsf{GapCDH}}$.

MULTI-USER CHALLENGE-RESPONSE GapCDH WITH CORRUPTIONS. The $(n, Q_{\mathrm{CH}})$-CorrCRGapCDH problem is a generalization of $n$-CorrGapCDH, where the adversary is additionally given $Q_{\mathrm{CH}}$ many challenge-response pairs $(R_k, h_k)$, for adaptively chosen $R_k \in \mathbb{G}$. To win, the adversary must output any of the $n(n-1)Q_{\mathrm{CH}}$ possible Diffie-Hellman Challenge-Response keys $g^{a_i a_j h_k} \cdot R_k^{a_j}$ for two non-corrupted users $i \neq j$.

Another interpretation of the CorrCRGapCDH problem stems from canonical (three-round) identification schemes (a.k.a. $\Sigma$ protocols) with a designated Verifier, where the Prover (holding secret key $a_j$) sends commitment $R_k$, the Verifier (holding secret key $a_i$) responds with a random challenge $h_k$, and finally the Prover sends the response $C = g^{a_i a_j h_k} \cdot R_k^{a_j}$. In this setting, the CorrCRGapCDH problem can

[4] Our new and previously known bounds for HMQV in Figure 2 are stated in the eCK model disallowing reflection attacks. The reason is that for reflection attacks, one additionally requires the hardness of Square Diffie-Hellman (i.e., compute $g^{a^2}$ from $g^a$) which is non-tightly equivalent to CDH. We remark that our generic group bounds from Figure 3 can be shown in the full eCK model allowing reflection attacks.

be seen as an $n$-user version with corruptions of Parallel IMPersonification against Key-Only Attack (PIMP-KOA) [KMP16].

The interpretation in the context of identification schemes gives a hint that the $(n, Q_{\mathrm{CH}})$-CorrCRGapCDH problem is again of qualitatively different nature than GapCDH and $n$-CorrGapCDH. Using techniques from [KMP16], one can prove that GapCDH and $(n, Q_{\mathrm{CH}})$-CorrCRGapCDH are asymptotically equivalent. However, since the proof involves the Forking Lemma [PS00], the resulting bound $\varepsilon^{\mathsf{CorrCRGapCDH}} \leq Q_{\mathrm{CH}} n^2 \cdot \sqrt{\varepsilon^{\mathsf{GapCDH}}}$ is highly non-tight.

Generic Hardness. In the generic group model (GGM) [Sho97], the running time of an adversary is captured by the number of queries to a group operation oracle. Ignoring constants, the advantages of an adversary making $Q_{\mathrm{OP}}$ group operations to a generic group of order $p$ are upper bounded by

$$\varepsilon^{\mathsf{CorrCRGapCDH}} \leq \frac{(Q_{\mathrm{OP}} + n)^2}{p} + \frac{n^2 Q_{\mathrm{CH}}}{p} \tag{1}$$

$$\varepsilon^{\mathsf{CorrGapCDH}} \leq \frac{(Q_{\mathrm{OP}} + n)^2}{p}. \tag{2}$$

We note that $\varepsilon^{\mathsf{CorrGapCDH}}$ is the same as the generic hardness of the standard discrete logarithm (DL) problem in [Sho97]. The generic hardness of CorrAGapCDH follows from that of CorrGapCDH.

## 1.3 Concrete Security of AKE Protocols

We will now state the concrete security bounds of the AKE protocols in the eCK model which depend on the number of users $N \geq 2$, the total number of sessions $S \geq 0$, the total number of test queries $T \geq 0$, and the number of random oracle queries $Q_{\mathrm{RO}}$.

Concrete Bounds from GapCDH. We summarize the previously known and our security loss of NAXOS, X3DH$^-$, and HMQV relative to GapCDH in Figure 2 on page 3. For HMQV [Kra05], we could not identify a concrete security bound in the literature so we had to estimate it from [Kra05,BCLS15] and the one of CMQV [Ust08]. The original bounds of NAXOS and HMQV are proven in a model that allows only a single test query. The bounds from Figure 2 are derived using a hybrid argument inducing a multiplicative factor of $T$, the number of test queries.

We stress that the multiplicative factor $T$ seems to be unavoidable using the original proof strategies of NAXOS [LLM07] and HMQV [Kra05]. Even using the random self reducibility of CDH, these strategies still need to guess $T$ possible test sessions out of $S$ many sessions in total, resulting in an exponential loss of $\binom{S}{T}$. Thus, the best way is to apply a hybrid argument and replace the keys one by one for each test query, which results in the security loss $T$. Our new assumptions resolve this issue and allow us to get rid of the factor $T$. In particular, we can replace the session keys of all $T$ test sessions at once as the reduction can embed challenge instances in all sessions and then adaptively choose which instance to solve, while allowing corruptions from adversaries.

We believe that improving the bound by the factor $T$ is relevant in practice. When combining session keys with a symmetric primitive, security should still hold for many sessions, thus $T$ can be about $2^{30}$, e.g. in modern messaging applications.

Concrete Bounds in the GGM. The main novelty of our multi-user CDH problems is that they allow us to give *optimal* security bounds for NAXOS, X3DH$^-$, and HMQV in the GGM. Our bounds in the eCK security model depend on the number of honest users $N$, sessions $S$, test sessions $T$, random oracle queries $Q_{\mathrm{RO}}$, and generic group operations $Q_{\mathrm{OP}}$ made by the adversary. Since $N$, $S$, and $T$ correspond to "online queries", we will merge them into one single value $t_{\mathrm{ON}} = N + S + T$, the time adversary $\mathcal{A}$ spends on online queries. Similarly, $t_{\mathrm{OFF}} = Q_{\mathrm{RO}} + Q_{\mathrm{OP}}$ counts the time that adversary $\mathcal{A}$ spends on "offline queries". (The reason is that offline queries are considerably less expensive than online queries, see below.) Figure 3 summarizes the security bounds in the GGM expressed as functions in $t_{\mathrm{ON}}, t_{\mathrm{OFF}}$.

We now explain the bounds for NAXOS in more detail. According to Figure 2, its security is tightly implied by $(N + S)$-CorrGapCDH. This means that in practice one can just pick a group $\mathbb{G}$ where the $(N+S)$-CorrGapCDH problem is hard (say, with 128-bit security) and implementing NAXOS in $\mathbb{G}$ directly gives us the same level of security (namely, 128-bit security) without increasing the group size. Applying (2) and using that $Q_{\mathrm{OP}} \geq (N + S)$, the quantitative hardness of NAXOS in the GGM is $(Q_{\mathrm{OP}}+N+S)^2/p = t_{\mathrm{OFF}}^2/p$. This is *optimal* in the sense that it matches the generic bounds on the best attack on NAXOS

| | Old GGM Bounds | | New GGM Bounds | |
|---|---|---|---|---|
| | $\varepsilon^{\mathsf{AKE}}(t_{\mathrm{OFF}}, t_{\mathrm{ON}})$ | **Bit security** | $\varepsilon^{\mathsf{AKE}}(t_{\mathrm{OFF}}, t_{\mathrm{ON}})$ | **Bit security** |
| NAXOS | $\frac{t_{\mathrm{ON}}^3 t_{\mathrm{OFF}}^2}{p}$ | 32 | $\frac{t_{\mathrm{OFF}}^2}{p}$ | 128 |
| X3DH$^-$ | $\frac{t_{\mathrm{ON}} t_{\mathrm{OFF}}^2}{p}$ | 96 | $\frac{t_{\mathrm{OFF}}^2}{p}$ | 128 |
| HMQV | $\frac{t_{\mathrm{ON}}^3 t_{\mathrm{OFF}}^2}{\sqrt{p}}$ | 0 | $\frac{t_{\mathrm{OFF}}^2 + t_{\mathrm{ON}}^2 t_{\mathrm{OFF}}}{p}$ | 128 |

**Fig. 3.** Security bounds in the GGM, where $t_{\mathrm{OFF}} = Q_{\mathrm{OP}} + Q_{\mathrm{RO}}$ counts the number of offline queries and $t_{\mathrm{ON}} = N + S + T$ counts the number of online queries. The "Bit security" columns refer to the bit security supported by the respective bounds over generic groups of order $p \approx 2^{256}$ and assuming $t_{\mathrm{ON}} \approx 2^{32}$ and $t_{\mathrm{OFF}} \lesssim 2^{128}$.

(which computes one DL and breaks the scheme). From previously known reductions [LLM07], one can only obtain the weaker GGM bound $T(N+S)^2(Q_{\mathrm{OP}}+N+S)^2/p = t_{\mathrm{ON}}^3 t_{\mathrm{OFF}}^2/p$. As for a concrete comparison, we compute the bit security offered by NAXOS when implemented over prime-order elliptic curves with $\log(p) = 256$. According to [CKMS16], a scheme offers a security level of $\kappa$ bits if $\varepsilon/(t_{\mathrm{ON}}+t_{\mathrm{OFF}}) \leq 2^{-\kappa}$ for all adversaries running in time $t_{\mathrm{ON}} + t_{\mathrm{OFF}}$ where $1 \leq t_{\mathrm{ON}} + t_{\mathrm{OFF}} \leq 2^{\kappa}$. A simple computation shows that our new bounds offer $\kappa = 128$ bits security as long as $t_{\mathrm{ON}} + t_{\mathrm{OFF}} \leq 2^{128}$. Using the bound from previously known proofs, one obtains a provable security guarantee of $128 - 3\log_2(t_{\mathrm{ON}})$ bits. Using the conservative $t_{\mathrm{ON}} = 2^{32}$ [CCG$^+$19], this makes only 32 bits. Since $(N+S, N)$-CorrAGapCDH implies $(N+S)$-CorrGapCDH, the computations for X3DH$^-$ are similar. The old GGM bound is obtained from the bound in [CCG$^+$19] which has a security loss linear in $N$.

The same computation shows that the quantitative hardness of HMQV in the GGM is $(Q_{\mathrm{OP}}+N+S)^2/p + (N+S)^2(Q_{\mathrm{RO}}+1)/p = (t_{\mathrm{OFF}}^2 + t_{\mathrm{ON}}^2 t_{\mathrm{OFF}})/p$. Hence HMQV over prime-order elliptic curves of size $\log(p) = 256$ offers a security of 128 bits as long as $t_{\mathrm{ON}} \leq 2^{64}$. In contrast, from previously known proofs one can only obtain $t_{\mathrm{ON}}^3 t_{\mathrm{OFF}}^2/\sqrt{p}$ which means that we are left with $-96$ bits of security (meaning zero). If, to guarantee 128 bits of security, group sizes were chosen according to this bound, they would be quite large, and the scheme correspondingly slow.

## 1.4 Discussion and Prior Work

We showed that for HMQV, X3DH$^-$, and NAXOS one can pay the price of stronger cryptographic assumptions for the benefit of getting tighter bounds. One might argue that our new assumptions partly "abstract away" the looseness of prior proofs and moreover come very close to a tautology of the AKE's security. While there is certainly some truth to the first statement, we would like to stress that our AKE security proofs are still rather complex and non-trivially relate the AKE experiment involving multiple oracles to the much simpler multi-user CDH experiment. Our new assumptions are purely algebraic and do not involve any hash function. Hence, they precisely characterize the "algebraic complexity" of the AKEs' security which certainly improves our understanding of their security. As a matter of fact, as a side result our approach also led to improved security reductions from the standard GapCDH assumption. Furthermore, our new generic bounds are the only known formal argument supporting the security of HMQV in 256-bit groups, c.f. Figure 3.

Another point of criticism might be that our new assumptions are non-falsifiable. We remark that the full Gap oracle (i.e., oracle DDH in Figure 4) is the only reason why our new assumptions (such as CorrGapCDH) are non-falsifiable. Previous (non-tight) proofs for HMQV and NAXOS also relied on the non-falsifiable GapCDH, whereas X3DH$^-$ was proved from the weaker and falsifiable Strong CDH assumption, where the first input of the DDH oracle is fixed. For simplicity we decided to analyze all protocols with respect to a gap assumption. But we would like to stress that for NAXOS and X3DH$^-$ we actually do not need the full power of the gap oracle in our proofs (see our comment in the beginning to Section 5). This way we can prove the security of NAXOS and X3DH$^-$ from falsifiable assumptions. Proving HMQV with respect to a falsifiable assumption remains an interesting open problem.

We analyzed the tightness of *existing* AKE protocols of practical relevance. The works [KMP16, BD20, FPS20] took a similar approach in the context of the Schnorr (blind) signature scheme. For example, [KMP16] proved that UF-CMA security of Schnorr signatures in the multi-user setting is tightly implied by the interactive $Q_{\mathrm{RO}}$-IDLOG assumption which in turn has optimal bounds in the GGM. In a different line of work, *new* AKE protocols with a tight security reduction from standard assumptions were created from scratch, for example [BHJ+15, CCG+19, JKRS21]. All those schemes are considerably less efficient than $\mathsf{NAXOS}$, $\mathsf{X3DH}^-$, and $\mathsf{HMQV}$.

<u>Open Problems.</u> We note that there are several variants of $\mathsf{HMQV}$ and $\mathsf{NAXOS}$, such as [Ust08, PW11, Ust09, YZ13]. We are optimistic that our analysis will carry over in a straightforward manner but leave the concrete analysis as an open problem. While we only use our assumptions to analyze two-message DH-based AKE protocols in this paper, we believe that our framework can be extended to analyze the Noise framework [Per17, DRS20] in combination of suitable symmetric primitives. Another interesting open problem is to improve the generic bound for $\mathsf{HMQV}$ to $t_{\text{OFF}}^2/p$, or to show an attack matching our slightly worse bound from Figure 3.

## 2 Preliminaries

<u>Notation.</u> For integers $N, M \in \mathbb{N}^+$, we define $[N, M] := \{N, N+1, \dots, M\}$ (which is the empty set for $M < N$) and $[N] := [1, N]$. For an adversary $\mathcal{A}$, we write $a \leftarrow \mathcal{A}(b)$ as the output of $\mathcal{A}$ on input $b$. To express $\mathcal{A}$'s random tape $\rho$ explicitly, we write $a := \mathcal{A}(b; \rho)$. In this case, $\mathcal{A}$'s execution is deterministic. The notation $[\![B]\!]$, where $B$ is a boolean statement, refers to a bit that is 1 if the statement is true and 0 otherwise.

<u>Games.</u> We use code-based games in this paper, following [BR06]. In every game, Boolean values are all initialized to false, numerical values to 0, sets to $\varnothing$, strings to undefined $\perp$. For the empty string, we use a special symbol $\epsilon$. A procedure terminates once it has returned an output.

<u>Idealized Models.</u> In the Generic Group Model (GGM) [Sho97, Mau05], group operations in group $\mathbb{G}$ can only be computed via an oracle Op (Op stands for operation) provided by the GGM, and adversaries only receive unique handles for the corresponding group elements. The GGM internally identifies elements in $\mathbb{G}$ with elements in $\mathbb{Z}_p$, since $(\mathbb{G}, \cdot)$ of order $p$ is isomorphic to $(\mathbb{Z}_p, +)$. Moreover, the GGM maintains an internal list that keeps track of all elements that have been issued. In this paper, our GGM proofs follow the work of Kiltz et al. [KMP16] which essentially uses the Maurer model [Mau05]. In the Random Oracle Model (ROM) [BR93], a hash function is modeled as a perfectly random function. That is, an adversary is only given access to the hash functions via an oracle $\mathsf{H}$ which (consistently) outputs uniform random elements in the hash function's range.

The running time of an adversary $\mathcal{A}$ in the GGM and ROM counts the number of calls to the Op and $\mathsf{H}$ oracles. We define such calls to the hash and group operation oracles as *offline* queries, since these operations can in practice be performed by an adversary offline, without any interaction with a server. In contrast, we define all queries that require interaction with a server as *online* queries. (For example, queries to a signing oracle in a digital signature scheme.) Adversary $\mathcal{A}$'s offline (or online) running time $t_{\text{OFF}}$ (or $t_{\text{ON}}$) is the time $\mathcal{A}$ spends on offline (or online) queries.

<u>Bit Security.</u> According to [CKMS16], a scheme has $\kappa$-bit security if $\varepsilon/(t_{\text{ON}}+t_{\text{OFF}}) \le 2^{-\kappa}$ for all adversaries that run in time $t_{\text{ON}} + t_{\text{OFF}}$ where $1 \le t_{\text{ON}} + t_{\text{OFF}} \le 2^\kappa$.

## 3 Multi-User CDH Problems

We formally define our new multi-user CDH problems $\mathsf{CorrGapCDH}$ and $\mathsf{CorrCRGapCDH}$, discuss their relation to the standard CDH problem and analyze their generic bounds.

For the rest of this section, we fix parameters $\mathsf{par} = (p, g, \mathbb{G})$ that describe a group $\mathbb{G}$ of prime order $p = |\mathbb{G}|$ and a generator $g$ of $\mathbb{G}$. For $g, A \in \mathbb{G}$, we define $\mathsf{DL}_g(A)$ as the unique $a \in \mathbb{Z}_p$ satisfying $g^a = A$.

<u>Standard $\mathsf{CDH}$.</u> We first recall the standard $\mathsf{CDH}$ problem which is to compute $g^{a_1 a_2}$ given $g^{a_1}$ and $g^{a_2}$ for randomly chosen $a_1, a_2 \xleftarrow{\$} \mathbb{Z}_p$. A popular variant for proving security of encryption and key exchange protocols is the Gap CDH $\mathsf{GapCDH}$ [OP01, ABR01] problem. In $\mathsf{GapCDH}$, the adversary can make queries to a gap oracle $\textsc{Ddh}(A, Y, Z)$ returning the Boolean value $[\![Y^{\mathsf{DL}_g(A)} = Z]\!]$.

<u>Multi-User $\mathsf{GapCDH}$.</u> We now consider natural generalizations of $\mathsf{GapCDH}$ to a setting with $n \ge 2$ users where the adversary is given the $n$-tuple $(g^{a_1}, \dots, g^{a_n})$ and in order to win, it must output any of the $n(n-1)$ possible CDH tuples in the winning set $\mathsf{Win} = \{g^{a_i a_j} \mid i \ne j\}$. Formally, to $n \ge 2$ and $Q_{\text{DDH}} \ge 0$, we associate game $\mathsf{GapCDH}_{n, Q_{\text{DDH}}}$ of Figure 4 and define the advantage function of $\mathcal{A}$ as

$\mathrm{Adv}^{\mathsf{GapCDH}}_{n,Q_{\mathrm{DDH}}}(\mathcal{A}) := \Pr[\mathsf{GapCDH}^{\mathcal{A}}_{n,Q_{\mathrm{DDH}}} \Rightarrow 1]$. We let $n$-$\mathsf{GapCDH}$ be the problem with parameters $n \geq 2$ such that $\mathsf{GapCDH} = 2$-$\mathsf{GapCDH}$. (To simplify notation we ignore the value $Q_{\mathrm{DDH}}$ when naming assumptions.) By a standard re-randomization argument [NR97] over the users, one can show that $n$-$\mathsf{GapCDH}$ is tightly equivalent to $\mathsf{GapCDH} = 2$-$\mathsf{GapCDH}$ .

```
GAME G
00 for $i \in [n]$
01     $a_i \xleftarrow{\$} \mathbb{Z}_p$; $A_i := g^{a_i}$
02 $C \leftarrow \mathcal{A}^{\mathrm{O}}(A_1, \cdots, A_n)$
03 return $[\![C \in \mathsf{Win}]\!]$

DDH$(X_\ell, Y_\ell, Z_\ell)$                 // $\ell$-th query ($\ell \in [Q_{\mathrm{DDH}}]$)
04 return $[\![Z_\ell = Y_\ell^{\mathsf{DL}_g(X_\ell)}]\!]$

CH$(R_k \in \mathbb{G})$                       // $k$-th query ($k \in [Q_{\mathrm{CH}}]$)
05 return $h_k \xleftarrow{\$} \mathbb{Z}_p$

CORR$_{n'}(i \in [n'])$
06 $\mathcal{L}_A := \mathcal{L}_A \cup \{i\}$
07 return $a_i$
```

$$\mathsf{Win} = \begin{cases} \{(A_i^{a_j} \mid (i,j) \in [n]^2 \wedge (i \neq j)\} & : \mathsf{G} = \mathsf{GapCDH}_{n,Q_{\mathrm{DDH}}} \\ \{(A_i^{a_j} \mid (i,j) \in ([n] \setminus \mathcal{L}_A)^2 \wedge (i \neq j)\} & : \mathsf{G} = \mathsf{CorrGapCDH}_{n,Q_{\mathrm{DDH}}} \\ \{(A_i^{a_j} \mid (i,j) \in ([n_1] \setminus \mathcal{L}_A) \times [n_1+1, n]\} & : \mathsf{G} = \mathsf{CorrAGapCDH}_{n,n_1,Q_{\mathrm{DDH}}} \\ \{(A_i^{h_k} \cdot R_k)^{a_j} \mid (i,j,k) \in ([n] \setminus \mathcal{L}_A)^2 \times [Q_{\mathrm{CH}}] \wedge (i \neq j)\} & : \mathsf{G} = \mathsf{CorrCRGapCDH}_{n,Q_{\mathrm{CH}},Q_{\mathrm{DDH}}} \end{cases}$$

$$\mathrm{O} = \begin{cases} \textsc{Ddh}(\cdot,\cdot,\cdot) & : \mathsf{G} = \mathsf{GapCDH}_{n,Q_{\mathrm{DDH}}} \\ \textsc{Ddh}(\cdot,\cdot,\cdot), \textsc{Corr}_n(\cdot) & : \mathsf{G} = \mathsf{CorrGapCDH}_{n,Q_{\mathrm{DDH}}} \\ \textsc{Ddh}(\cdot,\cdot,\cdot), \textsc{Corr}_{n_1}(\cdot) & : \mathsf{G} = \mathsf{CorrAGapCDH}_{n,n_1,Q_{\mathrm{DDH}}} \\ \textsc{Ddh}(\cdot,\cdot,\cdot), \textsc{Corr}_n(\cdot), \textsc{Ch}(\cdot) & : \mathsf{G} = \mathsf{CorrCRGapCDH}_{n,Q_{\mathrm{CH}},Q_{\mathrm{DDH}}} \end{cases}$$

**Fig. 4.** Game $\mathsf{G} \in \{\mathsf{GapCDH}_{n,Q_{\mathrm{DDH}}}, \mathsf{CorrGapCDH}_{n,Q_{\mathrm{DDH}}}, \mathsf{CorrAGapCDH}_{n,n_1,Q_{\mathrm{DDH}}}, \mathsf{CorrCRGapCDH}_{n,Q_{\mathrm{CH}},Q_{\mathrm{DDH}}}\}$ for defining our Multi-User CDH problems.

Multi-User GapCDH With Corruption. We now generalize the $n$-$\mathsf{GapCDH}$ problem to allow for user corruptions. Corruptions are modeled by oracle $\textsc{Corr}_n(i \in [n])$ which returns $a_i$, the discrete logarithm of $A_i = g^{a_i}$. To win, the adversary must output one of the Diffie-Hellman keys $g^{a_i a_j}$ for two distinct, non-corrupted users $i$ and $j$. More formally, to $n \geq 2$, and $Q_{\mathrm{DDH}} \geq 0$, we associate game $\mathsf{CorrGapCDH}_{n,Q_{\mathrm{DDH}}}$ of Figure 4 and define the advantage function of $\mathcal{A}$ as $\mathrm{Adv}^{\mathsf{CorrGapCDH}}_{n,Q_{\mathrm{DDH}}}(\mathcal{A}) := \Pr[\mathsf{CorrGapCDH}^{\mathcal{A}}_{n,Q_{\mathrm{DDH}}} \Rightarrow 1]$. We let $n$-$\mathsf{CorrGapCDH}$ be the problem with parameters $n \geq 2$ and $Q_{\mathrm{DDH}}$. We note that due to the corruption oracle a re-randomization argument as for the case without corruptions can no longer be applied and therefore we can not prove tight equivalence between $\mathsf{GapCDH}$ and $n$-$\mathsf{CorrGapCDH}$.

Multi-User Asymmetric GapCDH With Corruption. This problem is like $n$-$\mathsf{CorrGapCDH}$, where the corruption oracle $\textsc{Corr}_{n_1}(i \in [n_1])$ is restricted to users $i \in [n_1]$, where parameter $0 \leq n_1 \leq n$ splits interval $[n]$ in $[n_1]$ and $[n_1+1, n]$. To win, the adversary has to return one of the $\leq n_1(n-n_1)$ asymmetric Diffie-Hellman values $A_i^{a_j}$ for non-corrupted users $i \in [n_1]$ and $j \in [n_1+1, n]$. More formally, to $n \geq 2$, $0 \leq n_1 \leq n$, and $Q_{\mathrm{DDH}} \geq 0$, we associate game $\mathsf{CorrAGapCDH}_{n,n_1,Q_{\mathrm{DDH}}}$ of Figure 4 and define the advantage function of $\mathcal{A}$ as $\mathrm{Adv}^{\mathsf{CorrAGapCDH}}_{n,n_1,Q_{\mathrm{DDH}}}(\mathcal{A}) := \Pr[\mathsf{CorrAGapCDH}^{\mathcal{A}}_{n,n_1,Q_{\mathrm{DDH}}} \Rightarrow 1]$. We let $(n, n_1)$-$\mathsf{CorrAGapCDH}$ be the problem with parameters $n \geq 2$ and $0 \leq n_1 \leq n$.

Multi-User Challenge-Response GapCDH With Corruption. Our final problem is a generalization of the $n$-$\mathsf{CorrGapCDH}$ problem. The adversary is given access to a challenge oracle $\textsc{Ch}(R_k \in \mathbb{G})$ $(k \in [Q_{\mathrm{CH}}])$ which returns a response $h_k \xleftarrow{\$} \mathbb{Z}_p$. In the winning condition, the adversary is required to output any of the at most $n(n-1)Q_{\mathrm{CH}}$ elements of the winning set $\mathsf{Win} = \{(A_i^{h_k} \cdot R_k)^{a_j} \mid i \neq j \text{ uncorrupted}\}$. Furthermore, we will give the adversary access to the full gap oracle $\textsc{Ddh}$. More formally, to integers $n \geq 2$, $Q_{\mathrm{CH}} \geq 0$, and $Q_{\mathrm{DDH}} \geq 0$, we associate game $\mathsf{CorrCRGapCDH}_{n,Q_{\mathrm{CH}},Q_{\mathrm{DDH}}}$ of Figure 4 and define the advantage function $\mathrm{Adv}^{\mathsf{CorrCRGapCDH}}_{n,Q_{\mathrm{CH}},Q_{\mathrm{DDH}}}(\mathcal{A}) := \Pr[\mathsf{CorrCRGapCDH}^{\mathcal{A}}_{n,Q_{\mathrm{CH}},Q_{\mathrm{DDH}}} \Rightarrow 1]$. We let $(n, Q_{\mathrm{CH}})$-$\mathsf{CorrCRGapCDH}$ be the problem with parameters $n \geq 2$ and $Q_{\mathrm{CH}}$.

Relations. Figure 5 summarizes the relations between the multi-user CDH problems. We only state the important ones for our analysis here, all other formal statement and proofs are postponed to Appendix A.

$$\begin{tikzcd}
(n, n_1)\text{-}\mathsf{CorrAGapCDH} \arrow[d, shift left, "n_1\ (\text{L. } 3)"'] \arrow[dr, shift right] & n\text{-}\mathsf{GapCDH} \arrow[dl, shift left] & & \\
\boxed{\mathsf{GapCDH}} \arrow[u, shift left] \arrow[ur, shift left] \arrow[r, shift left, "n^2\ (\text{L. } 2)"] \arrow[d, shift right, "\text{rew. } (\text{L. } 6)"'] & n\text{-}\mathsf{CorrGapCDH} \arrow[l, shift left] \arrow[u] & & \\
(2,1)\text{-}\mathsf{CorrCRGapCDH} \arrow[u, shift right] \arrow[r, shift left, "n^2\ (\text{L. } 7)"] & (n,1)\text{-}\mathsf{CorrCRGapCDH} \arrow[l, shift left] \arrow[u, "(\text{L. } 1)"] \arrow[r, shift left, "Q_{\textsc{Ch}}\ (\text{L. } 8)"] & (n,\, Q_{\textsc{Ch}})\text{-}\mathsf{CorrCRGapCDH} \arrow[l, shift left] & \mathsf{true} \arrow[l, "\text{GGM (Th. } 2)"']
\end{tikzcd}$$

**Fig. 5.** Standard model relations between the standard problem GapCDH (CDH with full gap oracle) and our new problems $n$-GapCDH, $n$-CorrGapCDH, and $(n,\, Q_{\textsc{Ch}})$-CorrCRGapCDH. Red arrows denote non-tight implications with tightness loss as indicated; Green arrows denote tight implications; The black arrow denotes an unconditional statement in the GGM. Formal statements and proofs (unless trivial) are referenced.

**Theorem 1** (GapCDH $\xrightarrow{\textbf{non-tightly}}$ $(n,\, Q_{\textsc{Ch}})$-CorrCRGapCDH). *For any adversary $\mathcal{A}$ against $(n,\, Q_{\textsc{Ch}})$-CorrCRGapCDH, there exist an adversary $\mathcal{B}$ against GapCDH such that*

$$\mathrm{Adv}^{\mathsf{CorrCRGapCDH}}_{n,\, Q_{\textsc{Ch}},\, Q_{\textsc{Ddh}}}(\mathcal{A}) \leq Q_{\textsc{Ch}} \cdot n^2 \left( \sqrt{\mathrm{Adv}^{\mathsf{GapCDH}}_{Q_{\textsc{Ddh}}}(\mathcal{B})} + \frac{1}{p} \right), \text{ and } \mathbf{T}(\mathcal{B}) \approx 2\mathbf{T}(\mathcal{A}), \tag{3}$$

*where $\mathbf{T}(\mathcal{A})$ and $\mathbf{T}(\mathcal{B})$ are the running times of adversaries $\mathcal{A}$ and $\mathcal{B}$, respectively.*

The proof of Theorem 1 and Lemmas 6 to 8 referred to in Figure 5 can be found in Appendix A.1. Proofs of the following lemmas can be found in Appendix A.2.

**Lemma 1** ($(n,1)$-CorrCRGapCDH $\rightarrow$ $n$-CorrGapCDH). *For any adversary $\mathcal{A}$ against $n$-CorrGapCDH, there exists an adversary $\mathcal{B}$ against $(n,1)$-CorrCRGapCDH with*

$$\mathrm{Adv}^{\mathsf{CorrGapCDH}}_{n,\, Q_{\textsc{Ddh}}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{CorrCRGapCDH}}_{n,1,\, Q_{\textsc{Ddh}}}(\mathcal{B}).$$

**Lemma 2** (GapCDH $\xrightarrow{n^2}$ $n$-CorrGapCDH). *For any adversary $\mathcal{A}$ against $n$-CorrGapCDH, there exists an adversary $\mathcal{B}$ against GapCDH with*

$$\mathrm{Adv}^{\mathsf{CorrGapCDH}}_{n,\, Q_{\textsc{Ddh}}}(\mathcal{A}) \leq n^2 \cdot \mathrm{Adv}^{\mathsf{GapCDH}}_{Q_{\textsc{Ddh}}}(\mathcal{B}).$$

**Lemma 3** (GapCDH $\xrightarrow{n_1}$ $(n, n_1)$-CorrAGapCDH). *For any adversary $\mathcal{A}$ against $(n, n_1)$-CorrAGapCDH, there exists an adversary $\mathcal{B}$ against GapCDH with*

$$\mathrm{Adv}^{\mathsf{CorrAGapCDH}}_{n, n_1,\, Q_{\textsc{Ddh}}}(\mathcal{A}) \leq n_1 \cdot \mathrm{Adv}^{\mathsf{GapCDH}}_{Q_{\textsc{Ddh}}}(\mathcal{B}).$$

**Theorem 2 (Generic Hardness of CorrCRGapCDH).** *For an adversary $\mathcal{A}$ against $(n,\, Q_{\textsc{Ch}})$-CorrCR-GapCDH in the GGM that makes at most $Q_{\textsc{Op}}$ queries to the group oracle* Op, *$n'$ queries to the corruption oracle* Corr, *$Q_{\textsc{Ddh}}$ queries to the gap oracle* Ddh, *and $Q_{\textsc{Ch}}$ queries to the challenge oracle* Ch, *$\mathcal{A}$'s advantage is*

$$\mathrm{Adv}^{\mathsf{CorrCRGapCDH}}_{n,\, Q_{\textsc{Ch}},\, Q_{\textsc{Ddh}},\mathrm{GGM}}(\mathcal{A}) \leq \frac{(Q_{\textsc{Op}} + n + 1)^2}{p} + \frac{2Q_{\textsc{Ddh}}}{p} + \frac{(n - n')^2 Q_{\textsc{Ch}}}{2p} + \frac{Q_{\textsc{Ch}}(n - n')}{p}\ .$$

We analyze the hardness of $(n,\, Q_{\textsc{Ch}})$-CorrCRGapCDH in the generic group model (GGM) [Sho97, Mau05]. In particular, our GGM proofs follow the work of Kiltz et al. [KMP16] which essentially uses the Maurer model [Mau05]. Theorem 2 presents the hardness of $(n,\, Q_{\textsc{Ch}})$-CorrCRGapCDH in the GGM. Before proving it, we recall a useful lemma.

**Lemma 4 (Schwartz–Zippel Lemma).** *Let $f(x_1, .., x_n)$ be a non-zero multivariate polynomial of degree $d \geq 0$ over a field $\mathbb{F}$. Let $S$ be a finite subset of $\mathbb{F}$. Let $\alpha_1, \ldots, \alpha_n$ be chosen uniformly at random from $S$. Then*

$$\Pr[f(\alpha_1, \ldots, \alpha_n) = 0] \leq \frac{d}{|S|}.$$

At this point we also want to mention the non-triviality of applying Schwartz-Zippel in our proof which was raised in the analysis of the one-more discrete logarithm assumption [BFP21]. In particular, we cannot apply Schwartz-Zippel at the end of the proof since queries to Corr leak information about the input (i.e., the challenge elements $A_1, \dots, A_n$ which will be represented by polynomials $x_1, \dots, x_n$) and even allow the adversary to produce a collision.[5] We account for this in our proof by removing corrupted exponents from the challenge and transforming all polynomials accordingly. We also want to note that our Corr oracle is more restricted then the DLog oracle in [BFP21] which can be queried on any group element.

*Proof (of Theorem 2).* We construct a simulator $\mathcal{B}$ who interacts and plays a $\mathsf{CorrCRGapCDH}_{n,Q_{\text{CH}},Q_{\text{DDH}}}$ game with $\mathcal{A}$ in the GGM. Group operation, corruption and Ddh oracle queries are simulated as in Figure 6.

```
$\mathcal{B}$                                              //simulating in the GGM
$\mathcal{L}_E := \{(x_0 := 1, P_{x_0} := 1)\}$            //set of polynomials
for $i \in [n]$
    $\alpha_i \xleftarrow{\$} \mathbb{Z}_p$; $\mathcal{L}_E := \mathcal{L}_E \cup \{(x_i, P_{x_i} := i+1)\}$
$\vec{x} := (x_1, \dots, x_n)$
$\vec{\alpha} := (\alpha_1, \dots, \alpha_n)$
cnt $:= n + 1$                                             //size of $\mathcal{L}_E$
$C \leftarrow \mathcal{A}^{\mathrm{O}}(P_{x_0}, \dots, P_{x_n})$
if $C \notin [\text{cnt}]$
    return 0
fetch $(z^*(\vec{x}), C) \in \mathcal{L}_E$
if $\exists (f(\vec{x}), P), (g(\vec{x}), P') \in \mathcal{L}_E$
    and $f(\vec{x}) \neq g(\vec{x})$ and $f(\vec{\alpha}) = g(\vec{\alpha})$
    $\text{BAD}_{\mathbb{G}} := 1$; Abort
if $z^*(\vec{\alpha}) = (\alpha_{i^*} h_k + r_k(\vec{\alpha}))\alpha_{j^*}$
    if $(i^*, j^*, k) \in ([n] \setminus \mathcal{L}_A)^2 \times [Q_{\text{CH}}]$ and $i^* \neq j^*$
        return 1
return 0

Ddh$(P_i, P_j, P_k)$
if $(P_i, P_j, P_k) \notin [\text{cnt}]^3$
    return $\perp$
fetch $(a(\vec{x}), P_i), (b(\vec{x}), P_j), (c(\vec{x}), P_k) \in \mathcal{L}_E$
if $c(\vec{x}) = a(\vec{x}) \cdot b(\vec{x})$
    return 1
if $c(\vec{\alpha}) = a(\vec{\alpha}) \cdot b(\vec{\alpha})$
    $\text{BAD}_{\text{DDH}} := 1$; Abort
return 0
```

```
Corr$(i)$
if $i \notin [n]$
    return $\perp$
$\mathcal{L}_A := \mathcal{L}_A \cup \{i\}$
if $\exists (f(\vec{x}), P), (g(\vec{x}), P') \in \mathcal{L}_E$
    and $f(\vec{x}) \neq g(\vec{x})$ and $f(\vec{\alpha}) = g(\vec{\alpha})$
    $\text{BAD}_{\mathbb{G}} := 1$; Abort
let $\vec{x}'$ be the same as $\vec{x}$ but with $x_i = \alpha_i$
$\forall (f(\vec{x}), P_{f(\vec{x})}) \in \mathcal{L}_E$
    replace $(f(\vec{x}), P_{f(\vec{x})})$ with $(f(\vec{x}'), P_{f(\vec{x}')})$
$\vec{x} := (x_i)_{i \in [n] \setminus \mathcal{L}_A}$
$\vec{\alpha} := (\alpha_i)_{i \in [n] \setminus \mathcal{L}_A}$
return $\alpha_i$

Ch$(R_k)$                                 //$k$-th query $(k \in [Q_{\text{CH}}])$
if $\nexists (r_k(\vec{x}), R_k) \in \mathcal{L}_E$
    return $\perp$
$h_k \xleftarrow{\$} \mathbb{Z}_p$
return $h_k$

Op$(P, P')$
if $(P, P') \notin [\text{cnt}]^2$
    return $\perp$
fetch $(a(\vec{x}), P), (b(\vec{x}), P') \in \mathcal{L}_E$
$z(\vec{x}) := a(\vec{x}) + b(\vec{x})$
if $\exists (z(\vec{x}), P_{z(\vec{x})}) \in \mathcal{L}_E$
    return $P_{z(\vec{x})}$
cnt ++
$P_{z(\vec{x})} :=$ cnt
$\mathcal{L}_E := \mathcal{L}_E \cup \{(z(\vec{x}), P_{z(\vec{x})})\}$
return $P_{z(\vec{x})}$
```

**Fig. 6.** $\mathcal{B}$ simulates $\mathsf{CorrCRGapCDH}_{n,Q_{\text{CH}},Q_{\text{DDH}}}$ in the Generic Group Model (GGM) and interacts with $\mathcal{A}$. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\text{Ddh}, \text{Corr}, \text{Ch}, \text{Op}\}$.

Our overall idea is to simulate the $\mathsf{CorrCRGapCDH}_{n,Q_{\text{CH}},Q_{\text{DDH}}}$ game in a symbolic way using degree-1 polynomials. More precisely, during the simulation our simulator keeps an internal list $\mathcal{L}_E$ with entries of the form $(z(\vec{x}), P_{z(\vec{x})})$ where $z$ is a degree-1 polynomial and $P_{z(\vec{x})} \in \mathbb{N}$ identifies which entry it is. After $\mathcal{A}$ outputs a forgery, our simulator replaces the un-corrupted variables $(x_i)_{i \in [n] \setminus \mathcal{L}_A}$ with $\alpha_i \xleftarrow{\$} \mathbb{Z}_p$.

Now we note that the simulator perfectly simulates the $\mathsf{CorrCRGapCDH}_{n,Q_{\text{CH}},Q_{\text{DDH}}}$ in the GGM if both $\text{BAD}_{\text{DDH}}$ and $\text{BAD}_{\mathbb{G}}$ are equal to 0. Note that we check for collisions and thus for event $\text{BAD}_{\mathbb{G}}$ on every query to Corr. This is to address the issue described above. If $\text{BAD}_{\mathbb{G}}$ has not happened, we transform all polynomials in list $\mathcal{L}_E$ by evaluating them on $x_i = \alpha_i$ and removing $x_i$ and $\alpha_i$ from the respective vectors.

[5] The idea is to first query Corr on any index $i$ and receive the exponent $\alpha_i$. Then the adversary produces the *constant* polynomial $\alpha_i$. It should receive the handle for challenge $x_i$, but instead it receives a new handle (since $f(\vec{x}) \neq \alpha_i$) and has successfully produced a collision.

To bound the probability that one of the bad events happens, we use Lemma 4:

For each DDH query, $\Pr_{\vec{\alpha}}[c(\vec{x}) \neq a(\vec{x}) \cdot b(\vec{x})$ and $c(\vec{\alpha}) = a(\vec{\alpha}) \cdot b(\vec{\alpha})] \leq 2/p$, since $c(\vec{x}) - a(\vec{x}) \cdot b(\vec{x})$ is a non-zero polynomial of degree two. By the union bound, $\Pr[\mathrm{BAD}_{\mathrm{DDH}}] \leq 2Q_{\mathrm{DDH}}/p$, where $Q_{\mathrm{DDH}}$ is $\mathcal{A}$'s maximum number of DDH queries.

If event $\mathrm{BAD}_{\mathbb{G}}$ happens, there are two distinct degree-1 polynomials $z_i(\vec{x})$ and $z_j(\vec{x})$ in $\mathcal{L}_E$ that collide on input $\vec{\alpha} \xleftarrow{\$} \mathbb{Z}_p^n$. We want to bound that $\mathrm{BAD}_{\mathbb{G}}$ happens any time during the game, that is, either when CORR is queried or at the end of the simulation. Let $n'$ be the number of queries to CORR. For this we define $Q_{\mathrm{OP}} := q_0 + \ldots q_{n'}$, where $q_j$ is the number of $Q_{\mathrm{OP}}$ queries between the $j$-th and $(j+1)$-th CORR query and $q_{n'}$ is the number of $Q_{\mathrm{OP}}$ queries from the last CORR query to the end of the game. Let $\mathrm{EVENT}_j$ be the event that $\mathrm{BAD}_{\mathbb{G}}$ is set to 1 on the $j$-th query to CORR (and $\mathrm{BAD}_{\mathbb{G}}$ has not been set to 1 during a previous query). Note that $\mathrm{EVENT}_{n'}$ corresponds to $\mathrm{BAD}_{\mathbb{G}}$ happening at the end of the game. We have

$$\Pr[\mathrm{EVENT}_0] \leq \binom{q_0 + n + 1}{2} \cdot \frac{1}{p} \leq \frac{(q_0 + n + 1)^2}{2p} \qquad \text{and}$$

$$\forall j \in [n'] : \ \Pr[\mathrm{EVENT}_j] \leq \left( \binom{q_j}{2} + q_j \cdot \left( \sum_{k \in [0, j-1]} q_k + n + 1 \right) \right) \cdot \frac{1}{p} \leq \frac{q_j (\sum_{k \in [0,j]} q_k + n + 1)}{p} ,$$

where the $1/p$ factor comes from Lemma 4, and the fact that all our polynomials have degree one. For $\mathrm{EVENT}_j$ and $j > 0$, either two new polyonmials may collide on input $\vec{\alpha}$ or a new polynomial and one from a previous sequence.
Now we can bound $\mathrm{BAD}_{\mathbb{G}}$ by the sum of the individual probabilities and we additionally use that $\forall j$ we have $\sum_{k \in [0,j]} q_k \leq Q_{\mathrm{OP}}$. In particular, we get

$$\begin{aligned}
\Pr[\mathrm{BAD}_{\mathbb{G}}] &\leq \sum_{j \in [0,n']} \Pr[\mathrm{EVENT}_j] \\
&\leq \frac{(q_0 + n + 1)(Q_{\mathrm{OP}} + n + 1) + \sum_{j \in [n']} q_j (Q_{\mathrm{OP}} + n + 1)}{p} \\
&= \frac{(Q_{\mathrm{OP}} + n + 1)^2}{p} .
\end{aligned}$$

Recall that $n' := |\mathcal{L}_A|$ is the size of $\mathcal{L}_A$, i.e., the number of queries to CORR. The advantage function of $\mathcal{A}$ in the GGM can be bounded as

$$\begin{aligned}
\mathrm{Adv}^{\mathsf{CorrCRGapCDH}}_{n, Q_{\mathrm{CH}}, Q_{\mathrm{DDH}}, \mathrm{GGM}}(\mathcal{A}) \leq\ & \Pr[\mathrm{BAD}_{\mathbb{G}}] + \Pr[\mathrm{BAD}_{\mathrm{DDH}}] \\
& + \Pr_{\vec{\alpha}}[\exists (i^*, j^* \neq i^*, k) \in ([n] \setminus \mathcal{L}_A)^2 \times [Q_{\mathrm{CH}}] \colon z^*(\vec{\alpha}) = (\alpha_{i^*} h_k + r_k(\vec{\alpha}))\alpha_{j^*}] \\
\leq\ & \frac{(Q_{\mathrm{OP}} + n + 1)^2}{p} + \frac{2Q_{\mathrm{DDH}}}{p} + \frac{(n - n')^2 Q_{\mathrm{CH}}}{2p} + \frac{(n - n') Q_{\mathrm{CH}}}{p}.
\end{aligned}$$

To bound the third probability statement above, we use the following general inequality for events $A$ and $B$:

$$\Pr[A] = \Pr[A \mid B] \cdot \Pr[B] + \Pr[A \wedge \neg B] \cdot \Pr[\neg B] \leq \Pr[A \mid B] + \Pr[\neg B].$$

This allows us to split the statement into two terms, for which we can apply Lemma 4 to both and get

$$\begin{aligned}
& \Pr_{\vec{\alpha}}[\exists (i^*, j^* \neq i^*, k) \in ([n] \setminus \mathcal{L}_A)^2 \times [Q_{\mathrm{CH}}] \colon z^*(\vec{\alpha}) = (\alpha_{i^*} h_k + r_k(\vec{\alpha}))\alpha_{j^*}] \\
\leq & \Pr_{\vec{\alpha}}[\exists (i^*, j^*, k) \colon z^*(\vec{\alpha}) = (\alpha_{i^*} h_k + r_k(\vec{\alpha}))\alpha_{j^*} \mid \alpha_{i^*} h_k + r_k(\vec{\alpha}) \neq 0] \\
& + \Pr_{\vec{\alpha}}[\exists (i^*, k) \colon \alpha_{i^*} h_k + r_k(\vec{\alpha}) = 0] \\
\leq & \binom{n - n'}{2} \cdot \binom{Q_{\mathrm{CH}}}{1} \cdot \frac{1}{p} + \binom{n - n'}{1} \cdot \binom{Q_{\mathrm{CH}}}{1} \cdot \frac{1}{p} \\
= & \frac{(n - n')^2 Q_{\mathrm{CH}}}{2p} + \frac{(n - n') Q_{\mathrm{CH}}}{p}.
\end{aligned}$$

The following corollary is obtained by applying Lemma 1 to Theorem 2.

**Corollary 1 (Generic Hardness of $\mathsf{CorrGapCDH}$).** *For an adversary $\mathcal{A}$ against $n$-$\mathsf{CorrGapCDH}$ in the GGM that makes at most $Q_{\text{OP}}$ queries to the group oracle* Op*, $n'$ queries to the corruption oracle* Corr*, and $Q_{\text{DDH}}$ queries to the gap oracle* Ddh*, $\mathcal{A}$'s advantage is*

$$\mathrm{Adv}^{\mathsf{CorrGapCDH}}_{n,Q_{\text{DDH}},\text{GGM}}(\mathcal{A}) \le \frac{(Q_{\text{OP}}+n+1)^2}{p} + \frac{2Q_{\text{DDH}}}{p} + \frac{(n-n')^2}{2p} + \frac{n-n'}{p} \ .$$

## 4 Two-Message Authenticated Key Exchange

A two-message key exchange protocol $\mathsf{AKE} = (\mathsf{Gen}_{\mathsf{AKE}}, \mathsf{Init}_{\mathrm{I}}, \mathsf{Init}_{\mathsf{R}}, \mathsf{Der}_{\mathsf{R}}, \mathsf{Der}_{\mathrm{I}})$ consists of five algorithms which are executed interactively by two parties as shown in Figure 7. We denote the party which initiates the session by $\mathsf{P}_i$ and the party which responds to the session by $\mathsf{P}_r$. The key generation algorithm $\mathsf{Gen}_{\mathsf{AKE}}$ outputs a key pair $(\mathsf{pk}, \mathsf{sk})$ for one party. The initialization algorithms $\mathsf{Init}_{\mathrm{I}}$ and $\mathsf{Init}_{\mathsf{R}}$ input the long-term secret key of the party running the algorithm and the corresponding peer's long-term public key and output a message $I$ or $R$ and a state $\mathrm{st}_{\mathrm{I}}$ or $\mathrm{st}_{\mathrm{R}}$. The derivation algorithms $\mathsf{Der}_{\mathrm{I}}$ and $\mathsf{Der}_{\mathsf{R}}$ take as input the corresponding long-term secret key, the peer's public key, a message $I$ or $R$ and the state. It computes a session key $K$. Note that the terms initiator and responder are used to identify the parties, but the notation does not enforce an order of execution. In particular, the protocols we are looking at here allow that messages can be sent simultaneously and both parties may store a state.

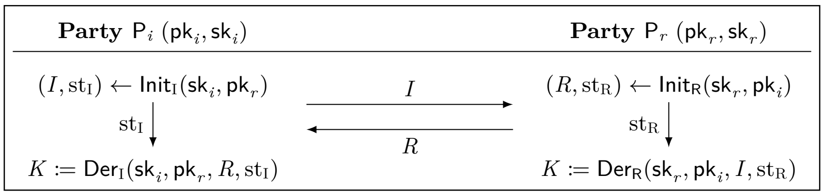
**Party** $\mathsf{P}_i$ $(\mathsf{pk}_i, \mathsf{sk}_i)$
**Party** $\mathsf{P}_r$ $(\mathsf{pk}_r, \mathsf{sk}_r)$
$(I, \mathrm{st}_{\mathrm{I}}) \leftarrow \mathsf{Init}_{\mathrm{I}}(\mathsf{sk}_i, \mathsf{pk}_r)$
$I$
$(R, \mathrm{st}_{\mathrm{R}}) \leftarrow \mathsf{Init}_{\mathsf{R}}(\mathsf{sk}_r, \mathsf{pk}_i)$
$\mathrm{st}_{\mathrm{I}}$
$R$
$\mathrm{st}_{\mathrm{R}}$
$K := \mathsf{Der}_{\mathrm{I}}(\mathsf{sk}_i, \mathsf{pk}_r, R, \mathrm{st}_{\mathrm{I}})$
$K := \mathsf{Der}_{\mathsf{R}}(\mathsf{sk}_r, \mathsf{pk}_i, I, \mathrm{st}_{\mathrm{R}})$

**Fig. 7.** Running a key exchange protocol between two parties.

We give a security game written in pseudocode in the style of [JKRS21]. We define two models for *implicitly authenticated* protocols achieving weak forward secrecy, where one is without and one is with state reveals. The latter models the same security as the eCK model [LLM07], extended by multiple test queries with respect to the same random bit $b$. The games $\mathsf{IND\text{-}wFS}$ and $\mathsf{IND\text{-}wFS\text{-}St}$ are given in Figures 8 and 9.

Execution Environment. We consider $N$ parties $\mathsf{P}_1, ..., \mathsf{P}_N$ with long-term key pairs $(\mathsf{pk}_n, \mathsf{sk}_n)$, $n \in [N]$. Each session between two parties has a unique identification number sID and variables which are defined relative to sID:

- init[sID] $\in [N]$ denotes the initiator of the session.
- resp[sID] $\in [N]$ denotes the responder of the session.
- type[sID] $\in$ {"In", "Re"} denotes the session's view, i. e. whether the initiator or the responder computes the session key.
- $I$[sID] denotes the message that was computed by the initiator.
- $R$[sID] denotes the message that was computed by the responder.
- state[sID] denotes the (secret) state information, i. e. ephemeral secret keys.
- sKey[sID] denotes the session key.

To establish a session between two parties, the adversary is given access to oracles $\textsc{Session}_{\mathrm{I}}$ and $\textsc{Session}_{\mathsf{R}}$, where the first one starts a session of type "In" and the second one of type "Re". In order to complete the session, the oracle $\textsc{Der}_{\mathrm{I}}$ or $\textsc{Der}_{\mathsf{R}}$ has to be queried. At any time, the adversary can register an *adversarially controlled* party by providing a long-term public key via the oracle RegisterLTK. The adversary does not need to know the corresponding secret key, but the party will be corrupted by definition. Note that oracles $\textsc{Session}_{\mathrm{I}}$ and $\textsc{Session}_{\mathsf{R}}$ cannot take an adversarially controlled party as owner.

```
**GAMES** IND-wFS and $\boxed{\text{IND-wFS-St}}$
$\mathrm{cnt_P} := N$
for $n \in [N]$
    $(\mathsf{pk}_n, \mathsf{sk}_n) \leftarrow \mathsf{Gen}_{\mathsf{AKE}}$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pk}_1, \cdots, \mathsf{pk}_N)$
for $\mathrm{sID}^* \in \mathcal{S}$
    if $\textsc{Fresh}(\mathrm{sID}^*) =$ false
        return $b$                          //session not fresh
    if $\textsc{Valid}(\mathrm{sID}^*) =$ false
        return $b$                          //no valid attack
return $[\![b = b']\!]$

$\textsc{Session}_{\mathrm{R}}((i,r) \in [\mathrm{cnt_P}] \times [N])$
$\mathrm{cnt_S}$ ++
$\mathrm{sID} := \mathrm{cnt_S}$
$(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i, r)$
$\mathrm{type}[\mathrm{sID}] :=$ "Re"
$(R, \mathrm{st}) \leftarrow \mathsf{Init}_{\mathrm{R}}(\mathsf{sk}_r, \mathsf{pk}_i)$
$(R[\mathrm{sID}], \mathrm{state}[\mathrm{sID}]) := (R, \mathrm{st})$
return $(\mathrm{sID}, R)$

$\textsc{Der}_{\mathrm{R}}(\mathrm{sID} \in [\mathrm{cnt_S}], I)$
if $\mathrm{sKey}[\mathrm{sID}] \neq \perp$ or $\mathrm{type}[\mathrm{sID}] \neq$ "Re"
    return $\perp$                          //no re-use
$(i, r) := (\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}])$
$\mathrm{st} := \mathrm{state}[\mathrm{sID}]$
$K := \mathsf{Der}_{\mathrm{R}}(\mathsf{sk}_r, \mathsf{pk}_i, I, \mathrm{st})$
$(I[\mathrm{sID}], \mathrm{sKey}[\mathrm{sID}]) := (I, K)$
return $\varepsilon$

$\boxed{\textsc{Rev-State}(\mathrm{sID})}$
$\mathrm{revState}[\mathrm{sID}] :=$ true
return $\mathrm{state}[\mathrm{sID}]$

$\textsc{Session}_{\mathrm{I}}((i,r) \in [N] \times [\mathrm{cnt_P}])$
$\mathrm{cnt_S}$ ++
$\mathrm{sID} := \mathrm{cnt_S}$
$(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i, r)$
$\mathrm{type}[\mathrm{sID}] :=$ "In"
$(I, \mathrm{st}) \leftarrow \mathsf{Init}_{\mathrm{I}}(\mathsf{sk}_i, \mathsf{pk}_r)$
$(I[\mathrm{sID}], \mathrm{state}[\mathrm{sID}]) := (I, \mathrm{st})$
return $(\mathrm{sID}, I)$

$\textsc{Der}_{\mathrm{I}}(\mathrm{sID} \in [\mathrm{cnt_S}], R)$
if $\mathrm{sKey}[\mathrm{sID}] \neq \perp$ or $\mathrm{type}[\mathrm{sID}] \neq$ "In"
    return $\perp$                          //no re-use
$(i, r) := (\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}])$
$\mathrm{st} := \mathrm{state}[\mathrm{sID}]$
$K := \mathsf{Der}_{\mathrm{I}}(\mathsf{sk}_i, \mathsf{pk}_r, R, \mathrm{st})$
$(R[\mathrm{sID}], \mathrm{sKey}[\mathrm{sID}]) := (R, K)$
return $\varepsilon$

$\textsc{Reveal}(\mathrm{sID})$
$\mathrm{revealed}[\mathrm{sID}] :=$ true
return $\mathrm{sKey}[\mathrm{sID}]$

$\textsc{Corrupt}(n \in [N])$
$\mathrm{corrupted}[n] :=$ true
return $\mathsf{sk}_n$

$\textsc{RegisterLTK}(\mathsf{pk})$
$\mathrm{cnt_P}$++
$\mathsf{pk}_{\mathrm{cnt_P}} := \mathsf{pk}$
$\mathrm{corrupted}[\mathrm{cnt_P}] :=$ true
return $\mathrm{cnt_P}$

$\textsc{Test}(\mathrm{sID})$
if $\mathrm{sID} \in \mathcal{S}$ return $\perp$          //already tested
if $\mathrm{sKey}[\mathrm{sID}] = \perp$ return $\perp$
$\mathcal{S} := \mathcal{S} \cup \{\mathrm{sID}\}$
$K_0^* := \mathrm{sKey}[\mathrm{sID}]$
$K_1^* \xleftarrow{\$} \mathcal{K}$
return $K_b^*$
```

**Fig. 8.** Games IND-wFS and IND-wFS-St for AKE. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\textsc{Session}_{\mathrm{I}}, \textsc{Session}_{\mathrm{R}}, \textsc{Der}_{\mathrm{I}}, \textsc{Der}_{\mathrm{R}}, \textsc{Reveal}, \textsc{Corrupt}, \textsc{RegisterLTK}, \textsc{Test}\}$. In game IND-wFS-St, $\mathcal{A}$ has additionally access to oracle Rev-State. Helper procedures Fresh and Valid are defined in Figure 9. If there exists any test session which is not both fresh and valid, the game will return the random bit $b$.

Furthermore, the adversary has access to oracles Corrupt and Reveal to obtain secret information. In game IND-wFS-St, the adversary has additional access to Rev-State. We use the following boolean values to keep track of which queries the adversary made:

– $\mathrm{corrupted}[n]$ denotes whether the long-term secret key of party $\mathsf{P}_n$ was given to the adversary.
– $\mathrm{revealed}[\mathrm{sID}]$ denotes whether the session key was given to the adversary.
– $\mathrm{revState}[\mathrm{sID}]$ denotes whether the state information of that session was given to the adversary.

The adversary can forward messages between sessions or modify them. By that, we can define the relationship between two sessions:

– **Matching Session**: Two sessions sID and sID′ *match* if the same parties are involved ($\mathrm{init}[\mathrm{sID}] = \mathrm{init}[\mathrm{sID}']$ and $\mathrm{resp}[\mathrm{sID}] = \mathrm{resp}[\mathrm{sID}']$), the messages sent and received are the same ($I[\mathrm{sID}] = I[\mathrm{sID}']$ and $R[\mathrm{sID}] = R[\mathrm{sID}']$) and they are of different types ($\mathrm{type}[\mathrm{sID}] \neq \mathrm{type}[\mathrm{sID}']$).

As we look at implicitly authenticated protocols that consist only of group elements, they are not vulnerable to no-match attacks described in [LS17].

Finally, the adversary is given access to oracle Test, which can be queried multiple times and which will return either the session key of the specified session or a uniformly random key. We use one bit $b$ for all test queries. We store test sessions in a set $\mathcal{S}$. In general, the adversary can disclose the complete interaction between two parties by querying the long-term secret keys, the state information and the

```
$\underline{\textsc{Fresh}(\text{sID}^*)}$
$(i^*, r^*) := (\text{init}[\text{sID}^*], \text{resp}[\text{sID}^*])$
$\mathfrak{M}(\text{sID}^*) := \{\text{sID} \mid (\text{init}[\text{sID}], \text{resp}[\text{sID}]) = (i^*, r^*) \ \wedge\ (I[\text{sID}], R[\text{sID}]) =$
        $(I[\text{sID}^*], R[\text{sID}^*]) \ \wedge\ \text{type}[\text{sID}] \neq \text{type}[\text{sID}^*]\}$        $/\!\!/$matching sessions
if revealed$[\text{sID}^*]$ or $(\exists \text{sID} \in \mathfrak{M}(\text{sID}^*) : \text{revealed}[\text{sID}] = \textbf{true})$
    return false        $/\!\!/ \mathcal{A}$ trivially learned the test session's key
if $\exists \text{sID} \in \mathfrak{M}(\text{sID}^*)$ s. t. $\text{sID} \in \mathcal{S}$
    return false        $/\!\!/ \mathcal{A}$ also tested a matching session
return true

$\underline{\textsc{Valid}(\text{sID}^*)}$
$(i^*, r^*) := (\text{init}[\text{sID}^*], \text{resp}[\text{sID}^*])$
$\mathfrak{M}(\text{sID}^*) := \{\text{sID} \mid (\text{init}[\text{sID}], \text{resp}[\text{sID}]) = (i^*, r^*) \ \wedge\ (I[\text{sID}], R[\text{sID}]) =$
        $(I[\text{sID}^*], R[\text{sID}^*]) \ \wedge\ \text{type}[\text{sID}] \neq \text{type}[\text{sID}^*]\}$        $/\!\!/$matching sessions
for attack $\in$ Table 2 $\boxed{\text{Table 1}}$
    if attack = true return true
return false
```

**Fig. 9.** Helper procedures Fresh and Valid for games IND-wFS and IND-wFS-St defined in Figure 8. Procedure Fresh checks if the adversary performed some trivial attack. In procedure Valid, each attack is evaluated by the set of variables shown in Table 1 (IND-wFS-St, excluding trivial attacks) or Table 2 (IND-wFS) and checks if an allowed attack was performed. If the values of the variables are set as in the corresponding row, the attack was performed, i. e. attack = **true**, and thus the session is valid.

session key. However, for each test session, we require that the adversary does not issue queries such that the session key can be trivially computed. We define the properties of freshness and validity which all test sessions have to satisfy:

– **Freshness**: A (test) session is called *fresh* if the session key was not revealed. Furthermore, if there exists a matching session, we require that this session's key is not revealed and that this session is not also a test session.
– **Validity**: A (test) session is called *valid* if it is fresh and the adversary performed any attack which is defined in the security model. We capture this with attack tables (cf. Tables 1 and 2).

Attack Tables. We define validity of different attack strategies. All attacks are defined using variables to indicate which queries the adversary may (not) make. We consider three dimensions:

– whether the test session is on the initiator's (type[sID$^*$] =“In”) or the responder's side (type[sID$^*$] =“Re”),
– all combinations of long-term secret key and state reveals (corrupted and revState variables),
– whether the adversary acted passively (matching session) or actively (no matching session).

This way, we capture all kind of combinations which are possible. From the 16 attacks in total, four are trivial wins for the adversary and thus they are excluded:

– Attack (9.)+(10.): no implicitly authenticated key exchange can achieve full forward security, so that we cannot reveal the long-term keys of both parties when there is no matching session.
– Attack (14.)+(15.): an adversary cannot reveal the long-term secret key of the test session's peer when there is no matching session, otherwise it can simply impersonate the party.

Instead of black-listing these trivial attacks, our model captures what the adversary is allowed to do. Hence, all non-trivial attacks are covered in our model, in particular capturing *weak forward secrecy* (wFS), *key compromise impersonation* (KCI) and *maximal exposure* (MEX) attacks. In more detail, wFS covers passive adversaries that are allowed to corrupt both parties' long-term keys after the session is completed (1.+2.). KCI covers adversaries that will try to impersonate an honest party to a corrupted party (13., 16.). MEX covers adversaries that have revealed any pair of long-term secret key and state, except for both the long-term key and state of one party (5.-8., 11.+12.).

An attack is performed if the variables are set to the corresponding values in the table. Table 1 is used for the IND-wFS-St security game, excluding trivial attacks highlighted in blue. For completeness, we add the trivial attack in row (0b.), where an adversary may query all secret information of a session. When not considering states, most of the attacks are redundant. This way, we obtain the *distilled* table for the IND-wFS security game given in Table 2.

However, if the protocol does not use appropriate randomness, it should not be considered secure. Thus, if the adversary is able to create more than one matching session to a test session, he may also

|  | $\mathcal{A}$ gets (Initiator, Responder) | corrupted[$i^*$] | corrupted[$r^*$] | type[sID$^*$] | revState[sID$^*$] | $\exists$sID $\in \mathfrak{M}$(sID$^*$) : revState[sID] | $\lvert\mathfrak{M}$(sID$^*$)$\rvert$ |
|---|---|---|---|---|---|---|---|
| 0a. | **multiple matching sessions** | – | – | – | – | – | $> 1$ |
| *0b. | **trivial attack** | – | – | – | – | – | – |
| 1.+2. | **(long-term, long-term)** | – | – | – | **F** | **F** | 1 |
| 3.+4. | **(state, state)** | **F** | **F** | – | – | – | 1 |
| 5. | **(long-term, state)** | – | **F** | "In" | **F** | – | 1 |
| 6. | **(long-term, state)** | – | **F** | "Re" | – | **F** | 1 |
| 7. | **(state, long-term)** | **F** | – | "In" | – | **F** | 1 |
| 8. | **(state, long-term)** | **F** | – | "Re" | **F** | – | 1 |
| *9.+10. | **(long-term, long-term)** | – | – | – | **F** | n/a | 0 |
| 11.+12. | **(state, state)** | **F** | **F** | – | – | n/a | 0 |
| 13. | **(long-term, state)** | – | **F** | "In" | **F** | n/a | 0 |
| *14. | **(long-term, state)** | – | **F** | "Re" | – | n/a | 0 |
| *15. | **(state, long-term)** | **F** | – | "In" | – | n/a | 0 |
| 16. | **(state, long-term)** | **F** | – | "Re" | **F** | n/a | 0 |

**Table 1.** Table of possible attacks for adversaries against implicitly authenticated two-message protocols with ephemeral state reveals. Trivial attacks are highlighted in blue color (and additionally marked with an asterisk *) and thus are NOT valid in our security definition. 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" and "n/a" indicates that there is no state which can be revealed as no matching session exists.

run a trivial attack. We model this in row (0a.) of Tables 1 and 2.

*How to read the tables.* As an example, we choose row (1.+2.) of Table 1. Then, if the test session is an initiating session, the state was not revealed (revState[sID$^*$] = **false**) and there is a matching session ($|\mathfrak{M}(\text{sID}^*)| = 1$) whose state was also not revealed, this row will evaluate to true. In this scenario, the adversary is allowed to query both long-term secret keys.

For all test sessions, at least one attack has to evaluate to true. If not, the game will return a random bit. The adversary wins if he does not make a trivial attack and distinguishes the session keys from uniformly random keys which he obtains through queries to the TEST oracle.

When proving the security of a protocol, the success probability for each attack strategy listed in the corresponding table will have to be analyzed, thus showing that independently of which queries the adversary makes, he cannot distinguish the session key from a uniformly random key.

In the protocols we look at, the state is defined as the ephemeral secret key (e.g., the exponent of a group element) and thus equivalent with the randomness which is used to compute the first message. Thus IND-wFS-St is exactly the same level of security as captured by the eCK model, extended by multiple test queries to the same random bit $b$.

**Definition 1 (Key Indistinguishability of AKE).** *We define games* IND-wFS *and* IND-wFS-St *as in Figures 8 and 9. The advantage of an adversary $\mathcal{A}$ against* AKE *in these games is defined as*

$$\mathrm{Adv}_{\mathsf{AKE}}^{\mathsf{IND\text{-}wFS}}(\mathcal{A}) := \left|2\Pr[\mathsf{IND\text{-}wFS}^{\mathcal{A}} \Rightarrow 1] - 1\right| \quad \textit{and}$$
$$\mathrm{Adv}_{\mathsf{AKE}}^{\mathsf{IND\text{-}wFS\text{-}St}}(\mathcal{A}) := \left|2\Pr[\mathsf{IND\text{-}wFS\text{-}St}^{\mathcal{A}} \Rightarrow 1] - 1\right| \quad .$$

## 5 Protocols X3DH$^-$ and NAXOS

In this section, we want to analyze the X3DH$^-$ and NAXOS protocols (see Figure 1 in the introduction). The protocols are defined relative to fixed parameters $(p, g, \mathbb{G})$ that describe a group $\mathbb{G}$ of prime order $p = |\mathbb{G}|$ and a generator $g$ of $\mathbb{G}$. $\mathsf{G}$ and $\mathsf{H}$ are hash functions with $\mathsf{G} : \{0,1\}^\lambda \times \mathbb{Z}_p \to \mathbb{Z}_p$ and $\mathsf{H} : \mathbb{G}^7 \to \{0,1\}^\lambda$, where $\lambda \geq \log(p)$.

|  | $\mathcal{A}$ gets (Initiator, Responder) | corrupted$[i^*]$ | corrupted$[r^*]$ | type[sID$^*$] | $\lvert\mathfrak{M}(\mathrm{sID}^*)\rvert$ |
|---|---|---|---|---|---|
| 0a. | **multiple matching sessions** | – | – | – | $> 1$ |
| 1.+2. | **(long-term, long-term)** | – | – | – | 1 |
| 13. | **(long-term, –)** | – | **F** | "In" | 0 |
| 16. | **(–, long-term)** | **F** | – | "Re" | 0 |

**Table 2.** Distilled table of attacks for adversaries against implicitly authenticated two-message protocols without ephemeral state reveals. 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 and **F** means "false".

We note that the original proof by Cohn-Gordon et al. [CCG$^+$19] for $\mathsf{X3DH}^-$ is based on the strong Diffie Hellman Assumption, where the first input of the DDH oracle is fixed. Our proof strategy does not allow for that as we handle multiple attacks at a time and avoid guessing. However, we want to stress that we do not require the full power of the gap oracle, but could restrict ourselves to queries to DDH, where the first value is one of the input elements of the corresponding multi-user CDH problem. The same applies to the proof of NAXOS.

Also note that $\mathsf{X3DH}^-$ is insecure under ephemeral key reveals, so we prove security in a weaker model as done in the original proof by [CCG$^+$19].

**Theorem 3** **($(N+S,\ N)$-CorrAGapCDH + $S$-GapCDH $\xrightarrow{\textbf{tight, ROM}}$ $\mathsf{X3DH}^-$ IND-wFS).** *For any IND-wFS adversary $\mathcal{A}$ against $\mathsf{X3DH}^-$ with $N$ parties that establishes at most $S$ sessions and issues at most $T$ queries to the* TEST *oracle and at most $Q_\mathsf{H}$ queries to the random oracle $\mathsf{H}$, there exist an adversary $\mathcal{B}$ against $(N+S,\ N)$-CorrAGapCDH and an adversary $\mathcal{C}$ against $S$-GapCDH with running times $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{C})$ such that*

$$\mathrm{Adv}^{\mathsf{IND\text{-}wFS}}_{\mathsf{X3DH}^-}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{CorrAGapCDH}}_{N+S,\ N,\, 3Q_\mathsf{H}}(\mathcal{B}) + \mathrm{Adv}^{\mathsf{GapCDH}}_{S,\ Q_\mathsf{H}}(\mathcal{C}) + \frac{(N+S)^2}{p} \quad .$$

The proof is given in Appendix B.

**Theorem 4** **($(N+S)$-CorrGapCDH $\xrightarrow{\textbf{tight, ROM}}$ NAXOS IND-wFS-St).** *For any IND-wFS-St adversary $\mathcal{A}$ against NAXOS with $N$ parties that establishes at most $S$ sessions and issues at most $T$ queries to the* TEST *oracle, at most $Q_\mathsf{G}$ queries to random oracle $\mathsf{G}$ and at most $Q_\mathsf{H}$ queries to random oracle $\mathsf{H}$, there exists an adversary $\mathcal{B}$ against $(N+S)$-CorrGapCDH with running time $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ such that*

$$\mathrm{Adv}^{\mathsf{IND\text{-}wFS\text{-}St}}_{\mathsf{NAXOS}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{CorrGapCDH}}_{N+S,\, 3Q_\mathsf{H}}(\mathcal{B}) + \frac{(N+S)^2}{p} + \frac{S^2}{p} + \frac{2Q_\mathsf{G}S}{p} \quad .$$

*Proof.* Let $\mathcal{A}$ be an adversary against IND-wFS-St security of NAXOS, where $N$ is the number of parties, $S$ is the maximum number of sessions that $\mathcal{A}$ establishes and $T$ is the maximum number of test sessions. Consider the sequence of games in Figure 10.

<u>GAME $G_0$</u>. This is the original IND-wFS-St game. In this game, we implicitly assume that all long-term keys, all messages output by SESSION$_\mathsf{I}$ and SESSION$_\mathsf{R}$, and all ephemeral secret keys are different. If such a collision happens, the game will abort. Using the birthday paradox, the probability for that can be upper bounded by $(N+S)^2/(2p)$ for $N$ long-term key pairs and at most $S$ messages, where exponents are chosen uniformly at random from $\mathbb{Z}_p$, and $S^2/(2p)$ for ephemeral secret keys $esk$, which are chosen uniformly at random from $\{0,1\}^\lambda$ and $\lambda \geq \log(p)$. This rules out attack (0a.), as there will be no two sessions having the same transcript. We get

$$\Pr[\mathsf{IND\text{-}wFS\text{-}St}^\mathcal{A} \Rightarrow 1] \leq \Pr[G_0^\mathcal{A} \Rightarrow 1] + \frac{(N+S)^2}{2p} + \frac{S^2}{2p} \quad . \tag{4}$$

```
GAMES $G_0$, $G_1$, $\boxed{G_2}$
00 $\text{cnt}_P := N$
01 for $n \in [N]$
02     $a_n \xleftarrow{\$} \mathbb{Z}_p$; $A_n := g^{a_n}$
03     $(\mathsf{pk}_n, \mathsf{sk}_n) := (A_n, a_n)$
04 $b \xleftarrow{\$} \{0,1\}$
05 $b' \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pk}_1, \cdots, \mathsf{pk}_N)$
06 for $\text{sID}^* \in \mathcal{S}$
07     if $\text{FRESH}(\text{sID}^*) = \textbf{false}$ return $b$
08     if $\text{VALID}(\text{sID}^*) = \textbf{false}$ return $b$
09 return $[\![b = b']\!]$

$\text{SESSION}_\text{R}((i, r) \in [\text{cnt}_P] \times [N])$
10 $\text{cnt}_S$ ++
11 $\text{sID} := \text{cnt}_S$
12 $(\text{init}[\text{sID}], \text{resp}[\text{sID}]) := (i, r)$
13 $\text{type}[\text{sID}] := \text{“Re”}$
14 $esk_r \xleftarrow{\$} \{0,1\}^\lambda$
15 $y := \mathsf{G}(esk_r, a_r)$; $Y := g^y$
16 $(R[\text{sID}], \text{state}[\text{sID}]) := (Y, esk_r)$
17 return $(\text{sID}, Y)$

$\text{DER}_\text{R}(\text{sID} \in [\text{cnt}_S], X)$
18 if $\text{sKey}[\text{sID}] \neq \bot$ or $\text{type}[\text{sID}] \neq \text{“Re”}$
19     return $\bot$
20 $(i, r) := (\text{init}[\text{sID}], \text{resp}[\text{sID}])$
21 $(Y, esk_r) := (R[\text{sID}], \text{state}[\text{sID}])$
22 $y := \mathsf{G}(esk_r, a_r)$
23 $\text{ctxt} := (A_i, A_r, X, Y)$
24 $K := \mathsf{H}(\text{ctxt}, A_i^y, X^{a_r}, X^y)$
25 $(I[\text{sID}], \text{sKey}[\text{sID}]) := (X, K)$
26 return $\varepsilon$

$\text{TEST}(\text{sID})$
27 if $\text{sID} \in \mathcal{S}$ return $\bot$
28 if $\text{sKey}[\text{sID}] = \bot$ return $\bot$
29 $\mathcal{S} := \mathcal{S} \cup \{\text{sID}\}$
30 $K_0^* := \text{sKey}[\text{sID}]$
31 $\boxed{K_0^* \xleftarrow{\$} \mathcal{K}}$
32 $K_1^* \xleftarrow{\$} \mathcal{K}$
33 return $K_b^*$

$\text{SESSION}_\text{I}((i, r) \in [N] \times [\text{cnt}_P])$
34 $\text{cnt}_S$ ++
35 $\text{sID} := \text{cnt}_S$
36 $(\text{init}[\text{sID}], \text{resp}[\text{sID}]) := (i, r)$
37 $\text{type}[\text{sID}] := \text{“In”}$
38 $esk_i \xleftarrow{\$} \{0,1\}^\lambda$
39 $x := \mathsf{G}(esk_i, a_i)$; $X := g^x$
40 $(I[\text{sID}], \text{state}[\text{sID}]) := (X, esk_i)$
41 return $(\text{sID}, X)$

$\text{DER}_\text{I}(\text{sID} \in [\text{cnt}_S], Y)$
42 if $\text{sKey}[\text{sID}] \neq \bot$ or $\text{type}[\text{sID}] \neq \text{“In”}$
43     return $\bot$
44 $(i, r) := (\text{init}[\text{sID}], \text{resp}[\text{sID}])$
45 $(X, esk_i) := (I[\text{sID}], \text{state}[\text{sID}])$
46 $x := \mathsf{G}(esk_i, a_i)$
47 $\text{ctxt} := (A_i, A_r, X, Y)$
48 $K := \mathsf{H}(\text{ctxt}, Y^{a_i}, A_r^x, Y^x)$
49 $(R[\text{sID}], \text{sKey}[\text{sID}]) := (Y, K)$
50 return $\varepsilon$

$\mathsf{G}(esk, a)$
51 if $\mathsf{G}[esk, a] = z$
52     return $z$
53 elseif $\exists \text{sID}$ s. t. $esk = \text{st}[\text{sID}]$
   and $\text{revState}[\text{sID}] = \textbf{false}$
54     $\text{BAD}_{\text{STATE}} := \textbf{true}$
55     abort
56 else
57     $z \xleftarrow{\$} \mathbb{Z}_p$
58     $\mathsf{G}[esk, a] := z$
59     return $z$

$\mathsf{H}(A_i, A_r, X, Y, Z_1, Z_2, Z_3)$
60 if $\mathsf{H}[A_i, A_r, X, Y, Z_1, Z_2, Z_3] = K$
61     return $K$
62 else
63     $K \xleftarrow{\$} \mathcal{K}$
64     $\mathsf{H}[A_i, A_r, X, Y, Z_1, Z_2, Z_3] := K$
65     return $K$
```

**Fig. 10.** Games $G_0$-$G_2$ for the proof of Theorem 4. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\text{SESSION}_\text{I}, \text{SESSION}_\text{R}, \text{DER}_\text{I}, \text{DER}_\text{R}, \text{REV-STATE}, \text{REVEAL}, \text{CORRUPT}, \text{REGISTERLTK}, \text{TEST}, \mathsf{G}, \mathsf{H}\}$, where REGISTERLTK, CORRUPT, REV-STATE and REVEAL are defined as in the original IND-wFS-St game (Fig. 8). $G_0$ implicitly assumes that no long-term keys or messages generated by the experiment collide.

GAME $G_1$. In game $G_1$, we define event $\text{BAD}_{\text{STATE}}$ which occurs if the adversary makes a query to random oracle $\mathsf{G}$ on a string $esk \in \{0,1\}^\lambda$ which was used in any session, but was not revealed to the adversary yet (line 53). This will become important in the next game hop since we need to be able to reprogram $\mathsf{G}$ in case there is a REV-STATE query and CORRUPT query for the party involved. If $\text{BAD}_{\text{STATE}}$ happens, the game aborts. The probability for this event to happen can be upper bounded by the number of oracle queries and the number of sessions:

$$\left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_0^{\mathcal{A}} \Rightarrow 1]\right| \leq \Pr[\text{BAD}_{\text{STATE}}] \leq \frac{Q_\mathsf{G} \cdot S}{p} \ .$$

GAME $G_2$. In game $G_2$, the challenge oracle TEST always outputs a uniformly random key, independent from the bit $b$ (line 31). We use that

$$\begin{aligned}\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| &= \frac{1}{2}\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b = 0] + \Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b = 1]\right. \\ &\quad \left. - \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b = 0] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b = 1]\right| \\ &= \frac{1}{2}\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b = 0] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b = 0]\right| \ , \end{aligned} \tag{5}$$

where the last equation holds because $\Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b = 1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b = 1]$.

Due to the exclusion of collisions, a particular (test) session cannot be recreated, i.e., the adversary cannot create two sessions sID, sID′ of the same type that compute the same session key. Thus, the adversary must query the random oracle $\mathsf{H}$ on the correct input to distinguish a session key from a random key. We construct adversary $\mathcal{B}$ against $(N + S)$-$\mathsf{CorrGapCDH}$ in Figure 11 to interpolate between the two games. We now describe adversary $\mathcal{B}$ in detail.

$\mathcal{B}$ gets as input $(N + S)$ group elements and has access to oracles Corr and Ddh. The first $N$ group elements $(A_1, ..., A_N)$ are used as public keys for the parties $\mathsf{P}_1, ..., \mathsf{P}_N$ (line 02). The remaining group elements $(B_1, ..., B_S)$ will be used as outputs for Session$_\mathrm{I}$ and Session$_\mathrm{R}$. This means that whenever $\mathcal{A}$ initiates a session sID, $\mathcal{B}$ increments the session counter and chooses the secret random string *esk*. Instead of evaluating $\mathsf{G}$, it outputs the group element $B_{\mathrm{sID}}$ (lines 22, 14). Note that as long as *esk* is unknown to $\mathcal{A}$, this is a perfect simulation.

To identify queries to the random oracle with correct Diffie-Hellman tuples, $\mathcal{B}$ uses a flag $f$ which is added as additional entry in the list of queries to $\mathsf{H}$. This helps to reduce the number of Ddh queries in oracles Der$_\mathrm{I}$ or Der$_\mathrm{R}$. In particular, whenever $\mathcal{A}$ calls one of the two oracles, $\mathcal{B}$ first checks the list of queries to $\mathsf{H}$ (lines 70, 53) and if there is an entry with $f = 1$, it outputs the corresponding session key. If this is not the case, it checks if there is an entry with unknown Diffie-Hellman tuples (lines 72, 55). This is to keep session keys of matching sessions consistent. If there is no such entry, $\mathcal{B}$ chooses a session key uniformly at random (lines 75, 58) and adds an entry with unknown Diffie-Hellman tuples to the list. If $\mathcal{A}$ issues a query to $\mathsf{H}$ which has not been asked before, $\mathcal{B}$ checks if the Diffie-Hellman tuples are correct using the Ddh oracle (line 36). In this case, it sets the flag $f$ to 1. Furthermore, if there is an entry with unknown values, it updates the entry (line 39) and outputs the corresponding key. Otherwise, $f$ is set to 0. $\mathcal{B}$ chooses a key uniformly at random (line 43), adds an entry with $f$ to the list and outputs the key.

We now describe how we patch random oracle $\mathsf{G}$. As soon as the adversary has queried both Rev-State and Corrupt for the owner of the session (i.e., the initiator in a session of type "In" or the responder in a session of type "Re"), then it can query $\mathsf{G}$ on the respective inputs. Thus, we fix the output value of $\mathsf{G}$ at exactly that time, i.e., on a corrupt query (after a state reveal query) as well as on a state reveal query (after a corrupt query).

That is, whenever $\mathcal{A}$ calls Rev-State on sID, $\mathcal{B}$ checks if the owner of the session is corrupted (Fig. 11, lines 27, 30). If this is the case, we have to patch the random oracle $\mathsf{G}$ by querying the Corr oracle on $B_{\mathrm{sID}}$ which is the message output by this session (lines 28, 31). Note that the corresponding input has not been queried to $\mathsf{G}$ before because then event $\mathrm{BAD}_{\mathrm{State}}$ would have occurred.

Further, whenever $\mathcal{A}$ corrupts a party $\mathsf{P}_n$, $\mathcal{B}$ queries the Corr oracle on $n$ (line 81). We then have to patch $\mathsf{G}$ for all sessions where $\mathsf{P}_n$ is the owner and the state of that session was revealed (line 83). Note that the corresponding input has not been queried to $\mathsf{G}$ before because then $\mathcal{B}$ would have already aborted.

If $\mathcal{A}$ makes a query to $\mathsf{G}$, where the input $a$ equals the secret key of any user which was not corrupted before (line 92), i.e., $g^a = A_n$ for some $n \in [N]$, then $\mathcal{B}$ is able to compute a solution for the $\mathsf{CorrGapCDH}$ problem. It just looks for some $A_{n'}$ such that $n'$ was not queried to Corrupt or $B_{\mathrm{sID}}$ such that $b_{\mathrm{sID}}$ has not been revealed via a Corr query. Then it can output $C = (A_{n'})^a$ or $C = (B_{\mathrm{sID}})^a$ as valid solution. Note that such an $A_{n'}$ or $B_{\mathrm{sID}}$ must exist. Note also that in this case, the adversary $\mathcal{A}$ can trivially compute the session key for a valid test session.

We now show that if $\mathcal{A}$ queries to the random oracle on the correct input for at least one test session, $\mathcal{B}$ is able to output a solution $C \in \mathsf{Win}$ to the $\mathsf{CorrGapCDH}$ problem. Let $\mathrm{sID}^* \in \mathcal{S}$ be any test session and $\mathsf{H}[A_{i^*}, A_{r^*}, X^*, Y^*, Z_1^*, Z_2^*, Z_3^*, 1] = \mathrm{sKey}[\mathrm{sID}^*]$ be the corresponding entry in the list of hash queries. $\mathcal{B}$ has to find this query in the list and depending on which reveal queries $\mathcal{A}$ has made (i.e., which attack was performed), $\mathcal{B}$ returns either $Z_1^*$, $Z_2^*$ or $Z_3^*$ as described below. Therefore, we will now argue that for each possible attack listed in Table 1, there will be a correct solution for $\mathsf{CorrGapCDH}$.

```
$\mathcal{B}^{\textsc{Corr},\textsc{Ddh}}(A_1, ..., A_N, B_1, ..., B_S)$
cnt$_\mathrm{P}$ := $N$
for $n \in [N]$
    $(\mathsf{pk}_n, \mathsf{sk}_n) := (A_n, \perp)$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pk}_1, \cdots, \mathsf{pk}_N)$
for sID$^* \in \mathcal{S}$
    if $\textsc{Fresh}(\mathrm{sID}^*)$ = false  return $b$
    if $\textsc{Valid}(\mathrm{sID}^*)$ = false  return $b$
return $C \in \mathsf{Win}$ (see text)

$\textsc{Session}_\mathrm{I}((i, r) \in [N] \times [\mathrm{cnt_P}])$
cnt$_\mathrm{S}$ ++
sID := cnt$_\mathrm{S}$
(init[sID], resp[sID]) := $(i, r)$
type[sID] := "In"
$esk_i \xleftarrow{\$} \{0,1\}^\lambda$
$X := B_{\mathrm{sID}}$
$(I[\mathrm{sID}], \mathrm{state}[\mathrm{sID}]) := (X, esk_i)$
return (sID, $X$)

$\textsc{Session}_\mathrm{R}((i, r) \in [\mathrm{cnt_P}] \times [N])$
cnt$_\mathrm{S}$ ++
sID := cnt$_\mathrm{S}$
(init[sID], resp[sID]) := $(i, r)$
type[sID] := "Re"
$esk_r \xleftarrow{\$} \{0,1\}^\lambda$
$Y := B_{\mathrm{sID}}$
$(R[\mathrm{sID}], \mathrm{state}[\mathrm{sID}]) := (Y, esk_r)$
return (sID, $Y$)

$\textsc{Rev-State}(\mathrm{sID})$
revState[sID] := true
$(i, r)$ := (init[sID], resp[sID])
if type[sID] = "In" and corrupted[$i$]
    $b_{\mathrm{sID}} := \textsc{Corr}(N + \mathrm{sID})$
    $\mathsf{G}[\mathrm{state}[\mathrm{sID}], a_i] := b_{\mathrm{sID}}$
elseif type[sID] = "Re" and corrupted[$r$]
    $b_{\mathrm{sID}} := \textsc{Corr}(N + \mathrm{sID})$
    $\mathsf{G}[\mathrm{state}[\mathrm{sID}], a_r] := b_{\mathrm{sID}}$
return state[sID]

$\mathsf{H}(A_i, A_r, X, Y, Z_1, Z_2, Z_3)$
if $\mathsf{H}[A_i, A_r, X, Y, Z_1, Z_2, Z_3, \cdot\,] = K$
    return $K$
if $\textsc{Ddh}(A_i, Y, Z_1) = 1$
 and $\textsc{Ddh}(A_r, X, Z_2) = 1$
 and $\textsc{Ddh}(X, Y, Z_3) = 1$
    $f := 1$
    if $\mathsf{H}[A_i, A_r, X, Y, \perp, \perp, \perp, \perp] = K$
        replace $(\perp, \perp, \perp, \perp)$ with $(Z_1, Z_2, Z_3, f)$
        return $K$
else
    $f := 0$
$K \xleftarrow{\$} \mathcal{K}$
$\mathsf{H}[A_i, A_r, X, Y, Z_1, Z_2, Z_3, f] := K$
return $K$

$\textsc{Der}_\mathrm{I}(\mathrm{sID} \in [\mathrm{cnt_S}], Y)$
if sKey[sID] $\neq \perp$ or type[sID] $\neq$ "In"
    return $\perp$
$(i, r)$ := (init[sID], resp[sID])
$X := I[\mathrm{sID}]$
ctxt := $(A_i, A_r, X, Y)$
if $\exists$sID$'$ s. t. (type[sID$'$], $R$[sID$'$]) = ("Re", $Y$)
    $\mathcal{P} := \mathcal{P} \cup \{\mathrm{sID}\}$
if $\exists Z_1, Z_2, Z_3$ s. t. $\mathsf{H}[\mathrm{ctxt}, Z_1, Z_2, Z_3, 1] = K$
    sKey[sID] := $K$
elseif $\mathsf{H}[\mathrm{ctxt}, \perp, \perp, \perp, \perp] = K$
    sKey[sID] := $K$
else
    $K \xleftarrow{\$} \mathcal{K}$
    $\mathsf{H}[\mathrm{ctxt}, \perp, \perp, \perp, \perp] := K$
    sKey[sID] := $K$
($R$[sID], sKey[sID]) := $(Y, K)$
return $\varepsilon$

$\textsc{Der}_\mathrm{R}(\mathrm{sID} \in [\mathrm{cnt_S}], X)$
if sKey[sID] $\neq \perp$ or type[sID] $\neq$ "Re"
    return $\perp$
$(i, r)$ := (init[sID], resp[sID])
$Y := R[\mathrm{sID}]$
ctxt := $(A_i, A_r, X, Y)$
if $\exists$sID$'$ s. t. (type[sID$'$], $I$[sID$'$]) = ("In", $X$)
    $\mathcal{P} := \mathcal{P} \cup \{\mathrm{sID}\}$
if $\exists Z_1, Z_2, Z_3$ s. t. $\mathsf{H}[\mathrm{ctxt}, Z_1, Z_2, Z_3, 1] = K$
    sKey[sID] := $K$
elseif $\mathsf{H}[\mathrm{ctxt}, \perp, \perp, \perp, \perp] = K$
    sKey[sID] := $K$
else
    $K \xleftarrow{\$} \mathcal{K}$
    $\mathsf{H}[\mathrm{ctxt}, \perp, \perp, \perp, \perp] := K$
    sKey[sID] := $K$
($I$[sID], sKey[sID]) := $(X, K)$
return $\varepsilon$

$\textsc{Corrupt}(n \in [N])$
corrupted[$n$] := true
$a_n \leftarrow \textsc{Corr}(n)$
$\mathsf{sk}_n := a_n$
$\forall$sID with ((init[sID], type[sID]) = ($n$, "In")
  or (resp[sID], type[sID]) = ($n$, "Re"))
  and revState[sID]
    $b_{\mathrm{sID}} \leftarrow \textsc{Corr}(N + \mathrm{sID})$
    $\mathsf{G}[\mathrm{state}[\mathrm{sID}], a_n] := b_{\mathrm{sID}}$
return $\mathsf{sk}_n$

$\mathsf{G}(esk, a)$
if $\mathsf{G}[esk, a] = z$
    return $z$
elseif $\exists$sID s. t. $esk$ = st[sID]
  and revState[sID] = false
    BAD$_{\textsc{State}}$ := true
    abort
elseif $\exists n \in [N]$ s. t. $A_n = g^a$
  and corrupted[$n$] = false
    abort and return $C \in \mathsf{Win}$ (see text)
else
    $z \xleftarrow{\$} \mathbb{Z}_p$
    $\mathsf{G}[esk, a] := z$
    return $z$
```

**Fig. 11.** Adversary $\mathcal{B}$ against $(N + S)$-$\mathsf{CorrGapCDH}$ for the proof of Theorem 4. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\textsc{Session}_\mathrm{I}, \textsc{Session}_\mathrm{R}, \textsc{Der}_\mathrm{I}, \textsc{Der}_\mathrm{R}, \textsc{Rev-State}, \textsc{Reveal}, \textsc{Corrupt}, \textsc{RegisterLTK}, \textsc{Test}, \mathsf{G}, \mathsf{H}\}$, where $\textsc{RegisterLTK}$, $\textsc{Reveal}$ and $\textsc{Test}$ are defined as in game $G_2$ of Figure 10. Lines written in blue color highlight how $\mathcal{B}$ simulates $G_1$ and $G_2$, respectively.

Attack (1.)+(2.). There is a matching session $\mathrm{sID}'$ and $\mathcal{A}$ has queried both long-term secret keys $a_{i^*}$ and $a_{r^*}$. $\mathcal{A}$ is not allowed to query the state of those sessions. W.l.o.g. assume the test session is of type "Re". Then, messages $X^*$ and $Y^*$ are chosen by the reduction $\mathcal{B}$ as $B_{\mathrm{sID}'}$ and $B_{\mathrm{sID}^*}$. Thus, in order to distinguish the session key, $\mathcal{A}$ has to compute $Z_3^* = \mathsf{DH}(X^*, Y^*) = \mathsf{DH}(B_{\mathrm{sID}'}, B_{\mathrm{sID}^*})$.

Attack (3.)+(4.). There is a matching session $\mathrm{sID}'$ and $\mathcal{A}$ has queried both states $esk_{i^*}$ and $esk_{r^*}$. $\mathcal{A}$ is not allowed to query the long-term secret keys of both parties. Again, we assume that the test session is of type "Re" (w.l.o.g). The states do not reveal any information about the exponents of $X^*$ and $Y^*$ (i.e., $B_{\mathrm{sID}'}$ and $B_{\mathrm{sID}^*}$), as $\mathcal{A}$ has not made a query to $\mathsf{G}$ specifying the correct long-term secret key. Also note that $\mathcal{B}$ never queried the Corr oracle to reveal the exponents of $B_{\mathrm{sID}'}$ and $B_{\mathrm{sID}^*}$ or $A_{i^*}$ and $A_{r^*}$. Thus, in order to distinguish the session key, $\mathcal{A}$ has to compute all of the Diffie-Hellman tuples $Z_1^* = \mathsf{DH}(A_{i^*}, B_{\mathrm{sID}^*})$, $Z_2^* = \mathsf{DH}(A_{r^*}, B_{\mathrm{sID}'})$ and $Z_3^* = \mathsf{DH}(B_{\mathrm{sID}^*}, B_{\mathrm{sID}'})$.

Attack (5.)+(6.). There is a matching session $\mathrm{sID}'$ and $\mathcal{A}$ has queried the initiator's long-term secret key $a_{i^*}$ and the responder's state $esk_{r^*}$, but neither the responder's long-term secret key $a_{r^*}$ nor the initiator's state $esk_{i^*}$. Again, assume the test session is of type "Re" (w.l.o.g.). Message $X^*$ is chosen as $B_{\mathrm{sID}'}$. In order to distinguish the session key, $\mathcal{A}$ has to compute $Z_2^* = \mathsf{DH}(A_{r^*}, X^*) = \mathsf{DH}(A_{r^*}, B_{\mathrm{sID}'})$.

Attack (7.)+(8.). This is the same as the case before, only that the adversary queried the other party's long-term key or state. Message $Y^*$ is chosen as $B_{\mathrm{sID}^*}$ and in order to distinguish the session key, $\mathcal{A}$ has to compute $Z_1^* = \mathsf{DH}(A_{i^*}, Y^*) = \mathsf{DH}(A_{i^*}, B_{\mathrm{sID}^*})$.

Attack (11.). The test session is of type "In" and there is no matching session. $\mathcal{A}$ has queried the initiator's state $esk_{i^*}$. Message $X^*$ is chosen as $B_{\mathrm{sID}^*}$, whereby $Y^*$ is chosen by $\mathcal{A}$. The state does not reveal any information about the exponent of $X^*$ ($B_{\mathrm{sID}^*}$) as $\mathcal{A}$ has not made a query to $\mathsf{G}$ on $(esk_{i^*}, a_{i^*})$. In order to distinguish the session key, $\mathcal{A}$ has to compute $Z_2^* = \mathsf{DH}(A_{r^*}, B_{\mathrm{sID}^*})$.

Attack (12.). The test session is of type "Re" and there is no matching session. $\mathcal{A}$ has queried the responder's state $esk_{r^*}$. Message $Y^*$ is chosen as $B_{\mathrm{sID}^*}$, whereby $X^*$ is chosen by $\mathcal{A}$. The state does not reveal any information about the exponent of $Y^*$ ($B_{\mathrm{sID}^*}$) as $\mathcal{A}$ has not made a query to $\mathsf{G}$ on $(esk_{r^*}, a_{r^*})$. In order to distinguish the session key, $\mathcal{A}$ has to compute $Z_1^* = \mathsf{DH}(A_{i^*}, B_{\mathrm{sID}^*})$ .

Attack (13.). The test session is of type "In" and there is no matching session. $\mathcal{A}$ has queried the initiator's long-term secret keys $a_{i^*}$. Message $X^*$ is chosen by the reduction $\mathcal{B}$ as $B_{\mathrm{sID}^*}$, whereby $Y^*$ is chosen by $\mathcal{A}$. In order to distinguish the session key, $\mathcal{A}$ has to compute $Z_2^* = \mathsf{DH}(A_{r^*}, X^*) = \mathsf{DH}(A_{r^*}, B_{\mathrm{sID}^*})$.

Attack (16.). The test session is of type "Re" and there is no matching session. $\mathcal{A}$ has queried the responder's long-term secret keys $a_{r^*}$. Message $Y^*$ is chosen by the reduction $\mathcal{B}$ as $B_{\mathrm{sID}^*}$, whereby $X^*$ is chosen by $\mathcal{A}$. In order to distinguish the session key, $\mathcal{A}$ has to compute $Z_1^* = \mathsf{DH}(A_{i^*}, Y^*) = \mathsf{DH}(A_{i^*}, B_{\mathrm{sID}^*})$.

The number of queries to the Ddh oracle is upper bounded by $3 \cdot Q_{\mathsf{H}}$. Thus,

$$\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b = 0] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b = 0]\right| \leq \mathrm{Adv}_{N+S,\,3Q_{\mathsf{H}}}^{\mathsf{CorrGapCDH}}(\mathcal{B}) \ .$$

Finally, the output of the Test oracle in $G_2$ is independent of the bit $b$, so we have

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2} \ .$$

Collecting the probabilities yields the bound stated in Theorem 4.

## 6 Protocol HMQV

The HMQV protocol was first presented in [Kra05]. Compared to the original protocol, we include the context into the hash of the session key (see Figure 1 in the introduction). The protocol is defined relative to fixed parameters $(p, g, \mathbb{G})$ that describe a group $\mathbb{G}$ of prime order $p = |\mathbb{G}|$ and a generator $g$ of $\mathbb{G}$. $\mathsf{G}$ and $\mathsf{H}$ are hash functions with $\mathsf{G} : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$ and $\mathsf{H} : \mathbb{G}^5 \to \{0,1\}^\lambda$, where $\lambda \geq \log(p)$.

One reason to include the context into the hash is the definition of matching sessions. The original proof is in the CK model which defines matching sessions solely based on the involved parties and transcripts. The eCK model additionally includes the session's type (initiator or responder). Now consider an active adversary that initiates two sessions of the same type. In the first query, it starts a session between parties $A$ and $B$ and receives message $X$. In the second query, it starts a session between $B$ and $A$ and receives message $Y$. Now it completes both sessions with the other message respectively. Both sessions will compute the same key, but will not be matching sessions (as they are both of type "In"), thus the adversary can trivially win. This issue also affects other role-symmetric protocols, as already noted by Cremers in [Cre09]. We can avoid it by including the context inside the hash, as done in the analysis of [BCLS15] and also in various variants of the protocol, e.g. [Ust08, YZ13, ZZ15].[6]

We give a tight reduction under $\mathsf{CorrCRGapCDH}$. However, we cannot show security against reflection attacks in general, which is why we require $i^* \neq r^*$ for all test sessions, indicated by the asterisk in $\mathsf{IND\text{-}wFS\text{-}St}^*$. Note that the original proof of $\mathsf{HMQV}$ needs the KEA assumption for the case that $i^* = r^*$ and $X \neq Y$ and the squared $\mathsf{CDH}$ assumption[7] for $i^* = r^*$ and $X = Y$, which is implied by the standard $\mathsf{CDH}$ assumption non-tightly.[8]

**Theorem 5** **($(N+S,\ Q_\mathsf{G}+2Q_\mathsf{H}+1)$-$\mathsf{CorrCRGapCDH} \xrightarrow{\text{tight, ROM}} \mathsf{HMQV}$ $\mathsf{IND\text{-}wFS\text{-}St}$).** *For any $\mathsf{IND\text{-}wFS\text{-}St}^*$ adversary $\mathcal{A}$ against $\mathsf{HMQV}$ with $N$ parties that establishes at most $S$ sessions and issues at most $T$ queries to the* Test *oracle and $Q_\mathsf{G}$ queries to random oracle $\mathsf{G}$ and $Q_\mathsf{H}$ queries to random oracle $\mathsf{H}$, there exists an adversary $\mathcal{B}$ against $(N+S,\ Q_\mathsf{G}+2Q_\mathsf{H}+1)$-$\mathsf{CorrCRGapCDH}$ with running time $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ such that*

$$\mathrm{Adv}_{\mathsf{HMQV}}^{\mathsf{IND\text{-}wFS\text{-}St}^*}(\mathcal{A}) \leq \mathrm{Adv}_{N+S,\ Q_\mathsf{G}+2Q_\mathsf{H}+1,\ Q_\mathsf{H}}^{\mathsf{CorrCRGapCDH}}(\mathcal{B}) + \frac{(N+S)^2}{p} \quad .$$

*Proof.* Let $\mathcal{A}$ be an adversary against $\mathsf{IND\text{-}wFS\text{-}St}^*$ security of $\mathsf{HMQV}$, where $N$ is the number of parties, $S$ is the maximum number of sessions that $\mathcal{A}$ establishes and $T$ is the maximum number of test sessions. Consider the sequence of games in Figure 12.

<u>Game $G_0$</u>. This is the original $\mathsf{IND\text{-}wFS\text{-}St}^*$ game. Similar to Equation (4), we implicitly assume that all long-term keys and all messages output by Session$_\mathrm{I}$ and Session$_\mathrm{R}$ are different. If such a collision happens, the game will abort. Using the birthday paradox, the probability for that can be upper bounded by $(N+S)^2/(2p)$ as there are $N$ long-term key pairs and at most $S$ messages, where exponents are chosen uniformly at random from $\mathbb{Z}_p$. This rules out attack (0a.), as there will be no two sessions having the same transcript. We get

$$\Pr[\mathsf{IND\text{-}wFS\text{-}St}^{*\mathcal{A}} \Rightarrow 1] = \Pr[G_0^\mathcal{A} \Rightarrow 1] + \frac{(N+S)^2}{2p} \quad .$$

<u>Game $G_1$</u>. In game $G_1$, the challenge oracle Test always outputs a uniformly random key, independent from the bit $b$ (line 56). To show the difference between $G_1$ and $G_0$, we can use that

$$\left|\Pr[G_1^\mathcal{A} \Rightarrow 1] - \Pr[G_0^\mathcal{A} \Rightarrow 1]\right| = \frac{1}{2}\left|\Pr[G_1^\mathcal{A} \Rightarrow 1 \mid b=0] - \Pr[G_0^\mathcal{A} \Rightarrow 1 \mid b=0]\right| \quad ,$$

as $\Pr[G_1^\mathcal{A} \Rightarrow 1 \mid b=1] = \Pr[G_0^\mathcal{A} \Rightarrow 1 \mid b=1]$.

Due to the exclusion of collisions, a particular (test) session cannot be recreated, i.e., the adversary cannot create two sessions sID, sID$'$ of the same type that compute the same session key. Thus, the only way to distinguish $G_1$ from $G_0$ is to query the random oracle on the correct input. We construct adversary $\mathcal{B}$ against $(N+S,\ Q_\mathsf{G}+2Q_\mathsf{H}+1)$-$\mathsf{CorrCRGapCDH}$ in Figure 13 to interpolate between the two games. We now describe adversary $\mathcal{B}$ in detail.

$\mathcal{B}$ gets as input $(N+S)$ group elements and has access to oracles Corr, Ch and Ddh. The first $N$ group elements $(A_1, ..., A_N)$ are used as public keys for the parties $\mathsf{P}_1, ..., \mathsf{P}_N$ (line 02). The remaining

[6] Even when dropping the session's type from the definition of matching sessions (similar to the original CK model), giving a tight proof for the original version of HMQV seems non-trivial since patching the random oracle $\mathsf{H}$ requires more care. In particular, it is always necessary to check if the input corresponds to any session for which the adversary can potentially compute the key, but the reduction itself cannot. In order to handle these queries in a naive way, the reduction needs to query the Ddh oracle once for each session, leading to $O(Q_\mathsf{H} \cdot S)$ queries.

[7] On input $g^x$, the squared $\mathsf{CDH}$ problem requires to compute $g^{x^2}$.

[8] We could also show security of $\mathsf{HMQV}$ including reflection attacks under a variant of $\mathsf{CorrCRGapCDH}$ that does not restrict the winning condition on $i \neq j$ and which can be reduced non-tightly to squared $\mathsf{GapCDH}$.

```
GAMES $G_0$, $\boxed{G_1}$
00 $\mathrm{cnt_P} := N$
01 for $n \in [N]$
02    $a_n \xleftarrow{\$} \mathbb{Z}_p$; $A_n := g^{a_n}$
03    $(\mathsf{pk}_n, \mathsf{sk}_n) := (A_n, a_n)$
04 $b \xleftarrow{\$} \{0,1\}$
05 $b' \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pk}_1, \cdots, \mathsf{pk}_N)$
06 for $\mathrm{sID}^* \in \mathcal{S}$
07    if $\textsc{Fresh}(\mathrm{sID}^*) =$ false
08       return $b$
09    if $\textsc{Valid}(\mathrm{sID}^*) =$ false
10       return $b$
11 return $[\![b = b']\!]$

$\textsc{Session}_\mathrm{R}((i,r) \in [\mathrm{cnt_P}] \times [N])$
12 $\mathrm{cnt_S}$ ++
13 $\mathrm{sID} := \mathrm{cnt_S}$
14 $(\mathrm{init[sID]}, \mathrm{resp[sID]}) := (i, r)$
15 $\mathrm{type[sID]} :=$ "Re"
16 $y \xleftarrow{\$} \mathbb{Z}_p$; $Y := g^y$
17 $(R[\mathrm{sID}], \mathrm{state[sID]}) := (Y, y)$
18 return $(\mathrm{sID}, Y)$

$\textsc{Der}_\mathrm{R}(\mathrm{sID} \in [\mathrm{cnt_S}], X)$
19 if $\mathrm{sKey[sID]} \neq \perp$ or $\mathrm{type[sID]} \neq$ "Re"
20    return $\perp$
21 $(i, r) := (\mathrm{init[sID]}, \mathrm{resp[sID]})$
22 $(Y, y) := (R[\mathrm{sID}], \mathrm{state[sID]})$
23 $d := \mathsf{G}(X, ID_r)$
24 $e := \mathsf{G}(Y, ID_i)$
25 $\sigma := (XA_i^d)^{y + ea_r}$
26 $K := \mathsf{H}((A_i, A_r, X, Y), \sigma)$
27 $(I[\mathrm{sID}], \mathrm{sKey[sID]}) := (X, K)$
28 return $\varepsilon$

$\mathsf{G}(Z, ID)$
29 if $\mathsf{G}[Z, ID] = h$
30    return $h$
31 else
32    $h \xleftarrow{\$} \mathbb{Z}_p$
33    $\mathsf{G}[Z, ID] := h$
34    return $h$

$\textsc{Session}_\mathrm{I}((i,r) \in [N] \times [\mathrm{cnt_P}])$
35 $\mathrm{cnt_S}$ ++
36 $\mathrm{sID} := \mathrm{cnt_S}$
37 $(\mathrm{init[sID]}, \mathrm{resp[sID]}) := (i, r)$
38 $\mathrm{type[sID]} :=$ "In"
39 $x \xleftarrow{\$} \mathbb{Z}_p$; $X := g^x$
40 $(I[\mathrm{sID}], \mathrm{state[sID]}) := (X, x)$
41 return $(\mathrm{sID}, X)$

$\textsc{Der}_\mathrm{I}(\mathrm{sID} \in [\mathrm{cnt_S}], Y)$
42 if $\mathrm{sKey[sID]} \neq \perp$ or $\mathrm{type[sID]} \neq$ "In"
43    return $\perp$
44 $(i, r) := (\mathrm{init[sID]}, \mathrm{resp[sID]})$
45 $(X, x) := (I[\mathrm{sID}], \mathrm{state[sID]})$
46 $d := \mathsf{G}(X, ID_r)$
47 $e := \mathsf{G}(Y, ID_i)$
48 $\sigma := (YA_r^e)^{x + da_i}$
49 $K := \mathsf{H}((A_i, A_r, X, Y), \sigma)$
50 $(R[\mathrm{sID}], \mathrm{sKey[sID]}) := (Y, K)$
51 return $\varepsilon$

$\textsc{Test}(\mathrm{sID})$
52 if $\mathrm{sID} \in \mathcal{S}$ return $\perp$
53 if $\mathrm{sKey[sID]} = \perp$ return $\perp$
54 $\mathcal{S} := \mathcal{S} \cup \{\mathrm{sID}\}$
55 $K_0^* := \mathrm{sKey[sID]}$
$\boxed{56\ K_0^* \xleftarrow{\$} \mathcal{K}}$
57 $K_1^* \xleftarrow{\$} \mathcal{K}$
58 return $K_b^*$

$\mathsf{H}(A_i, A_r, X, Y, \sigma)$
59 if $\mathsf{H}[A_i, A_r, X, Y, \sigma] = K$
60    return $K$
61 else
62    $K \xleftarrow{\$} \mathcal{K}$
63    $\mathsf{H}[A_i, A_r, X, Y, \sigma] := K$
64    return $K$
```

**Fig. 12.** Games $G_0$-$G_1$ for the proof of Theorem 5. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\textsc{Session}_\mathrm{I}, \textsc{Session}_\mathrm{R}, \textsc{Der}_\mathrm{I}, \textsc{Der}_\mathrm{R}$, $\textsc{Rev-State}, \textsc{Reveal}, \textsc{Corrupt}, \textsc{RegisterLTK}, \textsc{Test}, \mathsf{G}, \mathsf{H}\}$, where $\textsc{RegisterLTK}, \textsc{Corrupt}, \textsc{Rev-State}$ and $\textsc{Reveal}$ are defined as in the original $\mathsf{IND\text{-}wFS\text{-}St}^*$ game (see Figure 8). $G_0$ implicitly assumes that no long-term keys or messages generated by the experiment collide.

group elements $(B_1, ..., B_S)$ will be used as messages output by $\textsc{Session}_\mathrm{I}$ and $\textsc{Session}_\mathrm{R}$. This means that whenever $\mathcal{A}$ initiates a session sID, $\mathcal{B}$ increments the session counter and outputs the group element $B_{\mathrm{sID}}$ (lines 15, 45). To identify queries to the random oracle with $\sigma$, $\mathcal{B}$ uses a flag $f$ which is added as additional entry in the list of queries to $\mathsf{H}$. This helps to reduce the number of $\textsc{Ddh}$ queries in oracles $\textsc{Der}_\mathrm{I}$ or $\textsc{Der}_\mathrm{R}$. Thus, whenever $\mathcal{A}$ calls one of the two oracles, $\mathcal{B}$ first checks the list of random oracle queries if there has already been a query on the correct $\sigma$ indicated by $f = 1$ (lines 23, 53) and outputs the corresponding session key. If this is not the case, it checks if there is an entry with unknown $\sigma$ (lines 25, 55) to keep session keys of matching sessions consistent. If there is no such entry, $\mathcal{B}$ chooses a session key uniformly at random (lines 28, 58) and adds an entry with unknown $\sigma$ to the list. If $\mathcal{A}$ issues a random oracle query later, $\mathcal{B}$ checks if $\sigma$ is correct using the $\textsc{Ddh}$ oracle (line 71). In this case, it sets the flag $f$ to 1. Furthermore, if there is an entry with unknown $\sigma$ (line 73), it updates the entry with the correct value and outputs the corresponding key. Otherwise, $f$ is set to 0. $\mathcal{B}$ chooses a key uniformly at random (line 78), adds an entry with $f$ to the list and outputs the key.

```
$\mathcal{B}^{\textsc{Corr},\textsc{Ch},\textsc{Ddh}}(A_1, ..., A_N, B_1, ..., B_S)$
00 $\mathrm{cnt_P} := N$
01 for $n \in [N]$
02     $(\mathsf{pk}_n, \mathsf{sk}_n) := (A_n, \bot)$
03 $b \xleftarrow{\$} \{0,1\}$
04 $b' \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pk}_1, \cdots, \mathsf{pk}_N)$
05 for $\mathrm{sID}^* \in \mathcal{S}$
06     if $\textsc{Fresh}(\mathrm{sID}^*) = \mathbf{false}$
07         return 0
08     if $\textsc{Valid}(\mathrm{sID}^*) = \mathbf{false}$
09         return 0
10 return $C \in \mathsf{Win}$ (see text)

$\textsc{Session}_\mathrm{R}((i, r) \in [\mathrm{cnt_P}] \times [N])$
11 $\mathrm{cnt_S}$ ++
12 $\mathrm{sID} := \mathrm{cnt_S}$
13 $(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i, r)$
14 $\mathrm{type}[\mathrm{sID}] :=$ "Re"
15 $Y := B_\mathrm{sID}$
16 $(R[\mathrm{sID}], \mathrm{state}[\mathrm{sID}]) := (Y, \bot)$
17 return $(\mathrm{sID}, Y)$

$\textsc{Der}_\mathrm{R}(\mathrm{sID} \in [\mathrm{cnt_S}], X)$
18 if $\mathrm{sKey}[\mathrm{sID}] \neq \bot$ or $\mathrm{type}[\mathrm{sID}] \neq$ "Re"
19     return $\bot$
20 $(i, r) := (\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}])$
21 $Y := R[\mathrm{sID}]$
22 $\mathrm{ctxt} := (A_i, A_r, X, Y)$
23 if $\exists \sigma$ s. t. $\mathsf{H}[\mathrm{ctxt}, \sigma, 1] = K$
24     $\mathrm{sKey}[\mathrm{sID}] = K$
25 elseif $\mathsf{H}[\mathrm{ctxt}, \bot, \bot] = K$
26     $\mathrm{sKey}[\mathrm{sID}] = K$
27 else
28     $K \xleftarrow{\$} \mathcal{K}$
29     $\mathsf{H}[\mathrm{ctxt}, \bot, \bot] = K$
30     $\mathrm{sKey}[\mathrm{sID}] = K$
31 $(I[\mathrm{sID}], \mathrm{sKey}[\mathrm{sID}]) := (X, K)$
32 return $\varepsilon$

$\textsc{Corrupt}(n \in [N])$
33 $\mathrm{corrupted}[n] := \mathbf{true}$
34 $a_n \leftarrow \textsc{Corr}(n)$
35 $\mathsf{sk}_n := a_n$
36 return $\mathsf{sk}_n$

$\textsc{Rev-State}(\mathrm{sID})$
37 $\mathrm{revState}[\mathrm{sID}] := \mathbf{true}$
38 $b_\mathrm{sID} \leftarrow \textsc{Corr}(N + \mathrm{sID})$
39 $\mathrm{state}[\mathrm{sID}] := b_\mathrm{sID}$
40 return $\mathrm{state}[\mathrm{sID}]$

$\textsc{Session}_\mathrm{I}((i, r) \in [N] \times [\mathrm{cnt_P}])$
41 $\mathrm{cnt_S}$ ++
42 $\mathrm{sID} := \mathrm{cnt_S}$
43 $(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i, r)$
44 $\mathrm{type}[\mathrm{sID}] :=$ "In"
45 $X := B_\mathrm{sID}$
46 $(I[\mathrm{sID}], \mathrm{state}[\mathrm{sID}]) := (X, \bot)$
47 return $(\mathrm{sID}, X)$

$\textsc{Der}_\mathrm{I}(\mathrm{sID} \in [\mathrm{cnt_S}], Y)$
48 if $\mathrm{sKey}[\mathrm{sID}] \neq \bot$ or $\mathrm{type}[\mathrm{sID}] \neq$ "In"
49     return $\bot$
50 $(i, r) := (\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}])$
51 $(X, x) := (I[\mathrm{sID}], \mathrm{state}[\mathrm{sID}])$
52 $\mathrm{ctxt} := (A_i, A_r, X, Y)$
53 if $\exists \sigma$ s. t. $\mathsf{H}[\mathrm{ctxt}, \sigma, 1] = K$
54     $\mathrm{sKey}[\mathrm{sID}] = K$
55 elseif $\mathsf{H}[\mathrm{ctxt}, \bot, \bot] = K$
56     $\mathrm{sKey}[\mathrm{sID}] = K$
57 else
58     $K \xleftarrow{\$} \mathcal{K}$
59     $\mathsf{H}[\mathrm{ctxt}, \bot, \bot] = K$
60     $\mathrm{sKey}[\mathrm{sID}] = K$
61 $(R[\mathrm{sID}], \mathrm{sKey}[\mathrm{sID}]) := (Y, K)$
62 return $\varepsilon$

$\mathsf{G}(Z, ID)$
63 if $\mathsf{G}[Z, ID] = h$
64     return $h$
65 else
66     $h \leftarrow \textsc{Ch}(Z)$
67     $\mathsf{G}[Z, ID] := h$
68     return $h$

$\mathsf{H}(A_i, A_r, X, Y, \sigma)$
69 if $\mathsf{H}[A_i, A_r, X, Y, \sigma, \cdot\,] = K$
70     return $K$
71 if $(A_i, A_r) \in \{A_1, ..., A_N\}$
   and $\textsc{Ddh}(A_i^{\mathsf{G}(X, ID_r)} X, A_r^{\mathsf{G}(Y, ID_i)} Y, \sigma) = 1$
72     $f := 1$
73     if $\mathsf{H}[A_i, A_r, X, Y, \bot, \bot] = K$
74         replace $(\bot, \bot)$ with $(\sigma, f)$
75         return $K$
76 else
77     $f := 0$
78 $K \xleftarrow{\$} \mathcal{K}$
79 $\mathsf{H}[A_i, A_r, X, Y, \sigma, f] := K$
80 return $K$
```

**Fig. 13.** Adversary $\mathcal{B}$ against $(N + S,\ Q_\mathsf{G} + 2Q_\mathsf{H} + 1)$-$\mathsf{CorrCRGapCDH}$ for the proof of Theorem 5. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\textsc{Session}_\mathrm{I}, \textsc{Session}_\mathrm{R}, \textsc{Der}_\mathrm{I}, \textsc{Der}_\mathrm{R}, \textsc{Rev-State}, \textsc{Reveal}, \textsc{Corrupt}, \textsc{RegisterLTK}, \textsc{Test}, \mathsf{G}, \mathsf{H}\}$, where $\textsc{RegisterLTK}, \textsc{Rev-State}$ and $\textsc{Reveal}$ and are defined as in Figure 12. Lines written in blue color highlight how $\mathcal{B}$ simulates $G_0$ and $G_1$, respectively.

Whenever $\mathcal{A}$ calls $\textsc{Rev-State}$ on sID, $\mathcal{B}$ queries the $\textsc{Corr}$ oracle to reveal the exponent of $B_\mathrm{sID}$ which is the message output by this session (line 38) and returns the corresponding exponent. Similarly, whenever $\mathcal{A}$ corrupts a party $\mathsf{P}_n$, $\mathcal{B}$ queries the $\textsc{Corr}$ oracle on $n$ (line 34).

Whenever $\mathcal{A}$ queries the random oracle $\mathsf{G}$ on a new pair $(Z, ID)$, $\mathcal{B}$ queries its $\textsc{Ch}$ on $Z$ (line 66) and returns the output.

We now show that if $\mathcal{A}$ queries $\mathsf{H}$ on the correct input for at least one test session, $\mathcal{B}$ is able to output a solution $C \in \mathsf{Win}$ to the $\mathsf{CorrCRGapCDH}$ problem.

Let $\mathrm{sID}^* \in \mathcal{S}$ be any test session and $(A_{i^*}, A_{r^*}, X^*, Y^*)$ be the context of this test session. Then, $\mathcal{A}$ must have queried $\sigma^* = \mathsf{DH}(A_{i^*}^d X^*, A_{r^*}^e Y^*)$ to $\mathsf{H}$, where $d = \mathsf{G}(X^*, ID_{r^*})$ and $e = \mathsf{G}(Y^*, ID_{i^*})$. Let $d$

and $e$ be the $k_1$-th and $k_2$-th output of $\mathsf{G}$, which means that $(h_{k_1}, R_{k_1}) = (d, X^*)$ and $(h_{k_2}, R_{k_2}) = (e, Y^*)$. This means that there is an (updated) entry $(A_{i^*}, A_{r^*}, X^*, Y^*, \sigma^*, 1)$ in the list of queries to $\mathsf{H}$. $\mathcal{B}$ has to find this query in the list and depending on which reveal queries $\mathcal{A}$ has made (i.e., which attack was performed), $\mathcal{B}$ outputs a solution to $\mathsf{CorrCRGapCDH}$ as described below. Therefore, we will now argue that for each possible attack listed in Table 1, there will be a valid solution.

For the following cases, there is a matching session $\mathrm{sID}'$ and we assume (w.l.o.g.) that the test session is of type "Re". Then, messages $X^*$ and $Y^*$ are chosen by the reduction $\mathcal{B}$ as $B_{\mathrm{sID}'}$ and $B_{\mathrm{sID}^*}$. In order to distinguish the session key, $\mathcal{A}$ has to compute $\sigma^* = \mathsf{DH}(A_{i^*}^d B_{\mathrm{sID}'}, A_{r^*}^e B_{\mathrm{sID}^*})$.

<u>Attack (1.)+(2.).</u> $\mathcal{A}$ has queried both long-term secret keys $a_{i^*}$ and $a_{r^*}$. $\mathcal{A}$ is not allowed to query the state of those sessions. When $\mathcal{B}$ recognizes a query on $\sigma^*$, it first computes

$$\sigma^* \cdot (A_{r^*}^e Y^*)^{-da_{i^*}} \cdot (X^*)^{-ea_{r^*}} = \mathsf{DH}(X^*, Y^*)$$

and then, in order to get a valid forgery in the $\mathsf{CorrCRGapCDH}$, it has to make another query to the $\textsc{Ch}$ oracle. Note that this is the only case where we need this. Let it be the $k^*$-th query. $\mathcal{B}$ can choose $r \in \mathbb{Z}_p$ arbitrary and query $\textsc{Ch}$ on $R_{k^*} := g^r$. It will receive $h_{k^*}$ and then computes

$$\begin{aligned}(\mathsf{DH}(X^*, Y^*))^{h_{k^*}} \cdot (Y^*)^r &= \mathsf{DH}((X^*)^{h_{k^*}} R_{k^*}, Y^*)\\ &= \mathsf{DH}((B_{\mathrm{sID}'})^{h_{k^*}} R_{k^*}, B_{\mathrm{sID}^*})\end{aligned}$$

<u>Attack (3.)+(4.).</u> $\mathcal{A}$ has queried both states $b_{\mathrm{sID}^*}$ and $b_{\mathrm{sID}'}$, but is not allowed to query the long-term secret keys of both parties. When $\mathcal{B}$ recognizes a query on $\sigma^*$, it computes

$$\begin{aligned}\left(\sigma^* \cdot (A_{i^*}^d X^*)^{-b_{\mathrm{sID}^*}}\right)^{1/e} &= \mathsf{DH}(A_{i^*}^d X^*, A_{r^*}) \quad (6)\\ &= \mathsf{DH}(A_{i^*}^{h_{k_1}} R_{k_1}, A_{r^*})\end{aligned}$$

<u>Attack (5.)+(6.).</u> $\mathcal{A}$ has queried the initiator's long-term secret key $a_{i^*}$ and the responder's state $b_{\mathrm{sID}^*}$, but neither the responder's long-term secret key $a_{r^*}$ nor the initiator's state $b_{\mathrm{sID}'}$. When $\mathcal{B}$ recognizes a query on $\sigma^*$, it computes

$$\begin{aligned}\sigma^* \cdot (A_{r^*}^e Y^*)^{-da_{i^*}} &= \mathsf{DH}(A_{r^*}^e Y^*, X^*) \quad (7)\\ &= \mathsf{DH}(A_{r^*}^{h_{k_2}} R_{k_2}, B_{\mathrm{sID}'})\end{aligned}$$

<u>Attack (7.)+(8.).</u> This is the same as the case before, only that the adversary queried the other party's long-term key or state. $\mathcal{B}$ computes

$$\begin{aligned}\sigma^* \cdot (A_{i^*}^d X^*)^{-ea_{r^*}} &= \mathsf{DH}(A_{i^*}^d X^*, Y^*) \quad (8)\\ &= \mathsf{DH}(A_{r^*}^{h_{k_1}} R_{k_1}, B_{\mathrm{sID}^*})\end{aligned}$$

For the remaining cases, there is no matching session.

<u>Attack (11.).</u> The test session is of type "In" and $X^*$ is chosen as $B_{\mathrm{sID}^*}$, whereby $Y^*$ is chosen by $\mathcal{A}$. $\mathcal{A}$ has queried the initiator's state $b_{\mathrm{sID}^*}$. When $\mathcal{B}$ recognizes a query on $\sigma^*$, it computes

$$\begin{aligned}\left(\sigma^* \cdot (A_{r^*}^e Y^*)^{-b_{\mathrm{sID}^*}}\right)^{1/d} &= \mathsf{DH}(A_{r^*}^e Y, A_{i^*}) \quad (9)\\ &= \mathsf{DH}(A_{r^*}^{h_{k_2}} R_{k_2}, A_{i^*})\end{aligned}$$

<u>Attack (12.).</u> The test session is of type "Re" and $Y^*$ is chosen as $B_{\mathrm{sID}^*}$, whereby $X^*$ is chosen by $\mathcal{A}$. $\mathcal{A}$ has queried the responder's state $b_{\mathrm{sID}^*}$. When $\mathcal{B}$ recognizes a query on $\sigma^*$, it makes the same computation as in Equation (6).

<u>Attack (13.).</u> The test session is of type "In" and $X^*$ is chosen as $B_{\mathrm{sID}^*}$, whereby $Y^*$ is chosen by $\mathcal{A}$. $\mathcal{A}$ has queried the initiator's long-term secret keys $a_{i^*}$. When $\mathcal{B}$ recognizes a query on $\sigma^*$, it makes the same computation as in Equation (7).

<u>Attack (16.).</u> The test session is of type "Re" and $Y^*$ is chosen as $B_{\mathrm{sID}^*}$, whereby $X^*$ is chosen by $\mathcal{A}$. $\mathcal{A}$ has queried the responder's long-term secret keys $a_{r^*}$. When $\mathcal{B}$ recognizes a query on $\sigma^*$, it makes the

same computation as in Equation (8).

We showed that in each of these cases, the adversary outputs a correct solution for $\mathsf{CorrCRGapCDH}$. Note that the requirement that $i^* \neq r^*$ is needed for attacks (3.)+(4.), (11.) and (12.). The number of queries to the CH oracle is upper bounded by $Q_\mathsf{G} + 2Q_\mathsf{H} + 1$ and the number of queries to the DDH oracle by $Q_\mathsf{H}$. Thus,

$$\left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b = 0] - \Pr[G_0^{\mathcal{A}} \Rightarrow 1 \mid b = 0]\right| \leq \mathrm{Adv}^{\mathsf{CorrCRGapCDH}}_{N+S,\, Q_\mathsf{G}+2Q_\mathsf{H}+1,\, Q_\mathsf{H}}(\mathcal{B}) \ .$$

Finally, the output of the TEST oracle in $G_1$ is independent of the bit $b$, so we have

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2} \ .$$

Collecting the probabilities yields the bound stated in Theorem 5.

## 7 Concrete Bounds in the Generic Group Model

### 7.1 Generic Hardness of NAXOS

When analyzing NAXOS and X3DH$^-$, we obtain the following generic bound.

**Corollary 2 (Generic Hardness of NAXOS and X3DH$^-$).** *For any adversary $\mathcal{A}$ ($\mathcal{B}$) against* NAXOS (X3DH$^-$) *in the generic group and the random oracle model running in time $\mathbf{T}(\mathcal{A})$ ($\mathbf{T}(\mathcal{B})$), we have*

$$\mathrm{Adv}^{\mathsf{IND\text{-}wFS\text{-}St}}_{\mathsf{NAXOS},\mathrm{GGM}}(\mathcal{A}) = \mathrm{Adv}^{\mathsf{IND\text{-}wFS}}_{\mathsf{X3DH}^-,\mathrm{GGM}}(\mathcal{B}) = \Theta\left(\frac{\mathbf{T}(\mathcal{A})^2}{p}\right) \ .$$

*Proof.* Let $\mathcal{A}$ be an adversary against NAXOS with $N$ parties that establishes at most $S$ sessions and issues at most $T$ queries to the TEST oracle, at most $Q_\mathsf{G}$ queries to random oracle $\mathsf{G}$, at most $Q_\mathsf{H}$ queries to random oracle $\mathsf{H}$, and at most $Q_{\mathrm{OP}}$ queries to the group oracle. Then $\mathbf{T}(\mathcal{A}) = Q_{\mathrm{OP}} + N + S + T + Q_{\mathrm{RO}}$ is the running time of adversary $\mathcal{A}$. Let $\lambda \geq \log(p)$ be the output length of $\mathsf{G}$. Combining Corollary 1 with Theorem 4 we obtain

$$\begin{aligned}\mathrm{Adv}^{\mathsf{IND\text{-}wFS\text{-}St}}_{\mathsf{NAXOS},\mathrm{GGM}}(\mathcal{A}) &\leq \frac{(Q_{\mathrm{OP}} + N + S + 1)^2}{p} + \frac{6Q_\mathsf{H}}{p} + \frac{3(N+S)^2}{p} + \frac{S^2}{p} + \frac{2Q_\mathsf{G}S}{p} \\ &= O\left(\frac{\mathbf{T}(\mathcal{A})^2}{p}\right) ,\end{aligned}$$

where we bounded the term $\frac{(N+S-n')^2}{2p} + \frac{N+S-n'}{p} + \frac{(N+S)^2}{p} \leq \frac{3(N+S)^2}{p}$.

The lower bound $\Omega(\frac{\mathbf{T}(\mathcal{A})^2}{p})$ follows by a simple discrete logarithm attack on NAXOS. The same analysis applies to X3DH$^-$ since $\mathsf{CorrGapCDH}(N+S)$ tightly implies $(N+S, S)$-$\mathsf{CorrAGapCDH}$.

The corollary with matching upper and lower bounds shows that the generic bounds on NAXOS and X3DH$^-$ are optimal.

### 7.2 Generic Hardness of HMQV

For HMQV, we split the running time of $\mathcal{A}$ into its offline running time by $\mathbf{T}_{\mathrm{OFF}}(\mathcal{A}) = Q_{\mathrm{OP}} + Q_{\mathrm{RO}}$ and its online running time by $\mathbf{T}_{\mathrm{ON}}(\mathcal{A}) = N + S + T$. It is reasonable to assume that $\mathbf{T}_{\mathrm{OFF}}(\mathcal{A}) \gg \mathbf{T}_{\mathrm{ON}}(\mathcal{A})$, i.e., the adversary spends much more time on offline queries than on online queries.

**Corollary 3 (Generic Hardness of HMQV).** *For any adversary $\mathcal{A}$ against* HMQV *in the generic group and the random oracle model running in online time $\mathbf{T}_{\mathrm{ON}}(\mathcal{A})$ and offline time $\mathbf{T}_{\mathrm{OFF}}(\mathcal{A})$, we have*

$$\mathrm{Adv}^{\mathsf{IND\text{-}wFS\text{-}St}^*}_{\mathsf{HMQV},\mathrm{GGM}}(\mathcal{A}) = O\left(\frac{\mathbf{T}_{\mathrm{OFF}}(\mathcal{A})^2 + \mathbf{T}_{\mathrm{OFF}}(\mathcal{A}) \cdot \mathbf{T}_{\mathrm{ON}}(\mathcal{A})^2}{p}\right) \ .$$

*Proof.* Let $\mathcal{A}$ be an adversary against $\mathsf{HMQV}$ with $N$ parties that establishes at most $S$ sessions and issues at most $T$ queries to the $\textsc{Test}$ oracle, at most $Q_{\mathrm{RO}} := Q_{\mathsf{G}} + Q_{\mathsf{H}}$ queries to random oracles $\mathsf{G}$ and $\mathsf{H}$, and at most $Q_{\mathrm{OP}}$ queries to the group oracle. Then $\mathbf{T}_{\mathrm{OFF}}(\mathcal{A}) = Q_{\mathrm{OP}} + Q_{\mathrm{RO}}$ and $\mathbf{T}_{\mathrm{ON}}(\mathcal{A}) = N + S + T$ are the offline resp. online running times of adversary $\mathcal{A}$. Combining Theorem 2 with Theorem 5 and assuming $Q_{\mathrm{OP}} \geq (N+S)$, we obtain

$$\begin{aligned}\mathrm{Adv}^{\mathsf{IND\text{-}wFS\text{-}St}^*}_{\mathsf{HMQV},\mathrm{GGM}}(\mathcal{A}) &\leq \frac{(Q_{\mathrm{OP}} + N + S + 1)^2}{p} + \frac{2Q_{\mathrm{RO}}}{p} + \frac{3(N+S)^2(2Q_{\mathrm{RO}}+1)}{p} \\ &= O\left(\frac{\mathbf{T}_{\mathrm{OFF}}(\mathcal{A})^2}{p} + \frac{\mathbf{T}_{\mathrm{OFF}}(\mathcal{A}) \cdot \mathbf{T}_{\mathrm{ON}}(\mathcal{A})^2}{p}\right) ,\end{aligned}$$

where we bounded the term

$$\frac{(N+S-n')^2(2Q_{\mathrm{RO}}+1)}{2p} + \frac{(N+S-n')(2Q_{\mathrm{RO}}+1)}{p} + \frac{(N+S)^2}{p} \leq \frac{3(N+S)^2(2Q_{\mathrm{RO}}+1)}{p} .$$

For $\mathsf{HMQV}$ we have an additive term in addition to the optimal bound $\Omega(\frac{\mathbf{T}_{\mathrm{OFF}}(\mathcal{A})^2}{p})$. We claim that as long as $\mathbf{T}_{\mathrm{ON}}(\mathcal{A})$ is not too large, there is no need to increase the size of group $\mathbb{G}$.

We fix a group $\mathbb{G}$ where the DL problem has 128-bit security, meaning $p \approx 2^{256}$. Assuming $\mathbf{T}_{\mathrm{ON}}(\mathcal{A}) \leq 2^{64}$ and $\mathbf{T}_{\mathrm{OFF}}(\mathcal{A}) \leq 2^{128}$, we obtain by the corollary

$$\frac{\mathrm{Adv}^{\mathsf{IND\text{-}wFS\text{-}St}^*}_{\mathsf{HMQV},\mathrm{GGM}}(\mathcal{A})}{\mathbf{T}(\mathcal{A})} = \frac{\mathrm{Adv}^{\mathsf{IND\text{-}wFS\text{-}St}^*}_{\mathsf{HMQV},\mathrm{GGM}}(\mathcal{A})}{\mathbf{T}_{\mathrm{ON}}(\mathcal{A}) + \mathbf{T}_{\mathrm{OFF}}(\mathcal{A})} \lesssim \frac{\mathbf{T}_{\mathrm{OFF}}(\mathcal{A}) + \mathbf{T}_{\mathrm{ON}}(\mathcal{A})^2}{p} \lesssim 2^{-128} .$$

That is, $\mathsf{HMQV}$ has 128-bit security.

**Acknowledgments.** We would like to thank the anonymous reviewers for their thoughtful and constructive comments. Eike Kiltz was supported by the Deutsche Forschungsgemeinschaft (DFG, German research Foundation) as part of the Excellence Strategy of the German Federal and State Governments – EXC 2092 CASA - 390781972, and by the European Union (ERC AdG REWORC - 101054911). Jiaxin Pan was supported by the Research Council of Norway under Project No. 324235. Doreen Riepel was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

## References

ABR01. Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer, Heidelberg, April 2001.

BCLS15. Gilles Barthe, Juan Manuel Crespo, Yassine Lakhnech, and Benedikt Schmidt. Mind the gap: Modular machine-checked proofs of one-round key exchange protocols. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 689–718. Springer, Heidelberg, April 2015.

BD20. Mihir Bellare and Wei Dai. The multi-base discrete logarithm problem: Tight reductions and non-rewinding proofs for Schnorr identification and signatures. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 529–552. Springer, Heidelberg, December 2020.

BFP21. Balthazar Bauer, Georg Fuchsbauer, and Antoine Plouviez. The one-more discrete logarithm assumption in the generic group model. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 587–617. Springer, Heidelberg, December 2021.

BHJ+15. Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. Tightly-secure authenticated key exchange. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 629–658. Springer, Heidelberg, March 2015.

BP02. Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177. Springer, Heidelberg, August 2002.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

BR94. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1994.

BR06. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

CCG$^+$19. Katriel Cohn-Gordon, Cas Cremers, Kristian Gjøsteen, Håkon Jacobsen, and Tibor Jager. Highly efficient key exchange protocols with optimal tightness. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 767–797. Springer, Heidelberg, August 2019.

CK01. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, Heidelberg, May 2001.

CKMS16. Sanjit Chatterjee, Neal Koblitz, Alfred Menezes, and Palash Sarkar. Another look at tightness II: Practical issues in cryptography. Cryptology ePrint Archive, Report 2016/360, 2016. https://eprint.iacr.org/2016/360.

Cre09. Cas J.F. Cremers. Formally and practically relating the CK, CK-HMQV, and eCK security models for authenticated key exchange. Cryptology ePrint Archive, Report 2009/253, 2009. https://eprint.iacr.org/2009/253.

DRS20. Benjamin Dowling, Paul Rösler, and Jörg Schwenk. Flexible authenticated and confidential channel establishment (fACCE): Analyzing the noise protocol framework. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 341–373. Springer, Heidelberg, May 2020.

FPS20. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Heidelberg, May 2020.

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, Heidelberg, October 2021.

KMP16. Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Heidelberg, August 2016.

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, 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, Heidelberg, November 2007.

LMQ$^+$03. Laurie Law, Alfred Menezes, Minghua Qu, Jerry Solinas, and Scott Vanstone. An Efficient Protocol for Authenticated Key Agreement. *Des. Codes Cryptography*, 28(2):119–134, March 2003.

LS17. Yong Li and Sven Schäge. No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1343–1360. ACM Press, October / November 2017.

Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

MP16. Moxie Marlinspike and Trevor Perrin. The X3DH Key Agreement Protocol. 2016.

MQV95. Alfred Menezes, Minghua Qu, and Scott A. Vanstone. Some new key agreement protocols providing mutual implicit authentication. 1995.

NR97. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997.

OP01. Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer, Heidelberg, February 2001.

P1300. IEEE Standard Specifications for Public-Key Cryptography. *IEEE Std 1363-2000*, pages 1–228, 2000.

Per17. Trevor Perrin. The noise protocol framework. http://noiseprotocol.org/noise.html, 2017.

PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

PW11. Jiaxin Pan and Libin Wang. TMQV: A strongly eCK-secure Diffie-Hellman protocol without gap assumption. In Xavier Boyen and Xiaofeng Chen, editors, *ProvSec 2011*, volume 6980 of *LNCS*, pages 380–388. Springer, Heidelberg, October 2011.

SE16. Augustin P. Sarr and Philippe Elbaz-Vincent. On the security of the (F)HMQV protocol. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 16*, volume 9646 of *LNCS*, pages 207–224. Springer, Heidelberg, April 2016.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

Ust08. Berkant Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Des. Codes Cryptogr.*, 46(3):329–342, 2008.

Ust09. Berkant Ustaoglu. Comparing sessionstatereveal and ephemeralkeyreveal for Diffie-Hellman protocols. In Josef Pieprzyk and Fangguo Zhang, editors, *ProvSec 2009*, volume 5848 of *LNCS*, pages 183–197. Springer, Heidelberg, November 2009.

YZ13. Andrew Chi-Chih Yao and Yunlei Zhao. OAKE: a new family of implicitly authenticated Diffie-Hellman protocols. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 1113–1128. ACM Press, November 2013.

ZZ15. Shijun Zhao and Qianying Zhang. sHMQV: An efficient key exchange protocol for power-limited devices. Cryptology ePrint Archive, Report 2015/110, 2015. https://eprint.iacr.org/2015/110.

## A Omitted Proofs from Section 3

### A.1 Proof of Theorem 1

Theorem 1 is proved by Lemmas 6 to 8. Proofs of these lemmas are very similar to those in Section 3.2 of [KMP16], and are fairly straightforward. Hence, we only sketch them here, instead of defining them with pseudo-codes. We first recall the reset lemma from [BP02].

**Lemma 5 (Reset Lemma [BP02]).** *Let $H$ be a non-empty set. Let $\mathcal{C}$ be a randomized algorithm that on input $(I, h)$ returns a pair $(b, \sigma)$, where $b$ is a bit and $\sigma$ is called the side output. The accepting probability of $\mathcal{C}$ is defined as*

$$\mathsf{acc} := \Pr[b = 1 \mid h \xleftarrow{\$} H; (b, \sigma) \xleftarrow{\$} \mathcal{C}(I, h)]$$

*The reset algorithm $\mathcal{R}_\mathcal{C}$ associated to $\mathcal{C}$ is the randomized algorithm that takes input $I$ and proceeds as follows.*

```
Algorithm $\mathcal{R}_\mathcal{C}(I)$:
00 Pick random coins $\rho$
01 $h \xleftarrow{\$} H$
02 $(b, s) \xleftarrow{\$} \mathcal{C}(I, h; \rho)$
03 if $b = 0$ return $(\perp, \perp)$                // Abort in Phase 1
04 $h' \xleftarrow{\$} H$
05 $(b', s') \xleftarrow{\$} \mathcal{C}(I, h'; \rho)$
06 if $h \neq h'$ and $b' = 1$ return $(s, s')$
07 else return $(\perp, \perp)$
```

*Let*

$$\mathsf{frk} := \Pr[(s, s') \neq (\perp, \perp) \mid (s, s') \xleftarrow{\$} \mathcal{R}_\mathcal{C}(I)].$$

*Then*

$$\mathsf{frk} \geq \left(\mathsf{acc} - \frac{1}{|H|}\right)^2.$$

**Lemma 6** ($\mathsf{GapCDH} \xrightarrow{\textbf{rewind.}} (2,1)\text{-}\mathsf{CorrCRGapCDH}$). *For any adversary $\mathcal{A}$ against $(2,1)$-$\mathsf{CorrCRGapCDH}$, there exists an adversary $\mathcal{B}$ against $\mathsf{GapCDH}$ with $\mathbf{T}(\mathcal{B}) \approx 2\mathbf{T}(\mathcal{A})$ and*

$$\mathrm{Adv}^{\mathsf{CorrCRGapCDH}}_{2,1,Q_{\text{DDH}}}(\mathcal{A}) \leq \sqrt{\mathrm{Adv}^{\mathsf{GapCDH}}_{Q_{\text{DDH}}}(\mathcal{B})} + \frac{1}{p}.$$

*Proof.* We use the reset lemma (Lemma 5) with $H := \mathbb{Z}_p$ and $I := (A_1, A_2) := (g^{a_1}, g^{a_2})$. We first define an algorithm $\mathcal{C}((A_1, A_2), h; \rho)$ with oracle access to DDH that calls $\mathcal{A}((A_1, A_2); \rho)$. It answers $\mathcal{A}$'s single $\text{CH}(R)$ query with $h$ and $\mathcal{A}$'s DDH queries with DDH of $\mathsf{GapCDH}$ in a straightforward way. In $(2,1)$-$\mathsf{CorrCRGapCDH}$, $\mathcal{A}$ cannot ask any queries to CORR, as then $\mathcal{A}$ cannot win any more. Upon receiving $\mathcal{A}$'s forgery $C$, $\mathcal{C}$ checks if $C = (A_i^h R)^{a_j}$ with its DDH oracle, where $1 \leq i \neq j \leq 2$. If it holds, $\mathcal{C}$ returns $(1, s := (R, h, C))$; otherwise, $\mathcal{C}$ returns $(0, \perp)$. Thus, $\mathcal{C}$ returns $b = 1$ as long as $\mathcal{A}$ wins:

$$\mathsf{acc} := \mathrm{Adv}^{\mathsf{CorrCRGapCDH}}_{2,1,Q_{\text{DDH}}}(\mathcal{A}).$$

Now the reduction $\mathcal{B}$ that solves the $\mathsf{GapCDH}$ problem is constructed as follows. $\mathcal{B}$ gets the problem instance $(A_1, A_2)$ and has oracle access to $\textsc{Ddh}$ from the $\mathsf{GapCDH}$ problem. Next, it runs $(s, s') \xleftarrow{\$} \mathcal{R}_{\mathcal{C}}(I := (A_1, A_2))$ as described in Lemma 5, with $\mathcal{C}$ defined above. If $(s, s') \neq (\perp, \perp)$, then both transcripts $s = (R, h, C)$ and $s' = (R, h', C')$ are $(2,1)$-$\mathsf{CorrCRGapCDH}$ forgeries and $h \neq h'$. Wlog. we assume $C = (A_1^h R)^{a_2}$ and $C' = (A_1^{h'} R)^{a_2}$. $\mathcal{B}$ can compute $A_1^{a_2}$ as

$$A_1^{a_2} := \left(\frac{C}{C'}\right)^{(h-h')^{-1}}.$$

By construction, $\mathcal{B}$ is successful iff $\mathcal{R}_{\mathcal{C}}$ is successful. By Lemma 5, $\mathcal{B}$'s success probability is bounded by

$$\mathrm{Adv}_{Q_{\textsc{Ddh}}}^{\mathsf{GapCDH}}(\mathcal{B}) = \mathsf{frk} \geq \left(\mathrm{Adv}_{2,1,Q_{\textsc{Ddh}}}^{\mathsf{CorrCRGapCDH}}(\mathcal{A}) - \frac{1}{|\mathbb{Z}_p|}\right)^2.$$

During the simulation, $\mathcal{B}$ queries $\textsc{Ddh}$ at most $2(Q_{\textsc{Ddh}} + 1)$ times ($Q_{\textsc{Ddh}} + 1$ times for each execution of $\mathcal{C}$). The running time of $\mathcal{B}$, $\mathbf{T}(\mathcal{B})$, is that of $\mathcal{R}_{\mathcal{C}}$, namely, $2\mathbf{T}(\mathcal{A})$, plus the minor overhead of asking additional $\textsc{Ddh}$ and computing the final result $A_1^{a_2}$ from $C$ and $C'$. We write $\mathbf{T}(\mathcal{B}) \approx 2\mathbf{T}(\mathcal{A})$ to indicate that this is the dominating running time of $\mathcal{B}$.

**Lemma 7** **($(2,1)$-$\mathsf{CorrCRGapCDH} \xrightarrow{n^2} (n,1)$-$\mathsf{CorrCRGapCDH}$).** *For any adversary $\mathcal{A}$ against $(n,1)$-$\mathsf{CorrCRGapCDH}$, there exists an adversary $\mathcal{B}$ against $(2,1)$-$\mathsf{CorrCRGapCDH}$ with*

$$\mathrm{Adv}_{n,1,Q_{\textsc{Ddh}}}^{\mathsf{CorrCRGapCDH}}(\mathcal{A}) \leq n^2 \cdot \mathrm{Adv}_{2,1,Q_{\textsc{Ddh}}}^{\mathsf{CorrCRGapCDH}}(\mathcal{B}). \tag{10}$$

*Proof.* The simulator receives $B_1, B_2$ from the challenger, and chooses a pair of indexes $i^*, j^* \in [n]$ at random. For $i \in [n] \setminus \{i^*, j^*\}$, it chooses $a_i \xleftarrow{\$} \mathbb{Z}_p$ and computes $A_i := g^{a_i}$. Finally, it sets $(A_{i^*}, A_{j^*}) := (B_1, B_2)$, and calls $\mathcal{A}(A_1, \ldots, A_n)$. Any query to $\textsc{Ddh}$ is forwarded to the challengers oracle. The same is done for the single challenge call $\textsc{Ch}(R_1)$, which returns $h_1$. Calls to $\textsc{Corr}(A_i)$ for $i \neq i^*, j^*$ are answered with the corresponding discrete logarithm $a_i$. The queries $\textsc{Corr}(A_{i^*})$ and $\textsc{Corr}(A_{j^*})$ are forwarded to the challenger (note that such a query will cause us to lose the game). This simulator perfectly simulates a $\mathsf{CorrCRGapCDH}_{n,1,Q_{\textsc{Ddh}}}$ game from $\mathcal{A}$'s viewpoint. At some point, $\mathcal{A}$ will output a forgery

$$C \in \{(A_i^{h_1} \cdot R_1)^{a_j} \mid (i,j) \in ([n] \setminus \mathcal{L}_A)^2 \wedge (i \neq j)\}.$$

For the simulator to win, we require that either $(i^*, j^*) = (i,j)$ or $(i^*, j^*) = (j,i)$, meaning that we chose the correct pair of indexes from a set of $\binom{n}{2}$ pairs. This means that the probability of choosing correctly is $\frac{2}{n(n-1)} \geq \frac{1}{n^2}$, which gives us the result.

**Lemma 8** **($(n,1)$-$\mathsf{CorrCRGapCDH} \xrightarrow{Q_{\textsc{Ch}}} (n, Q_{\textsc{Ch}})$-$\mathsf{CorrCRGapCDH}$).** *For any adversary $\mathcal{A}$ against $(n, Q_{\textsc{Ch}})$-$\mathsf{CorrCRGapCDH}$, there exists an adversary $\mathcal{B}$ against $(n,1)$-$\mathsf{CorrCRGapCDH}$ with*

$$\mathrm{Adv}_{n,Q_{\textsc{Ch}},Q_{\textsc{Ddh}}}^{\mathsf{CorrCRGapCDH}}(\mathcal{A}) \leq Q_{\textsc{Ch}} \cdot \mathrm{Adv}_{n,1,Q_{\textsc{Ddh}}}^{\mathsf{CorrCRGapCDH}}(\mathcal{B}). \tag{11}$$

*Proof.* The simulation starts with $\mathcal{B}$ picking a random integer $r \xleftarrow{\$} [Q_{\textsc{Ch}}]$. Then, any query from $\mathcal{A}$ to either $\textsc{Ddh}$ or $\textsc{Corr}$ is forwarded to the appropriate oracle by. When responding to calls to $\textsc{Ch}$, $\mathcal{B}$ keeps track of how many such queries $\mathcal{A}$ has submitted. For every query $\textsc{Ch}(R_i)$ with $i \in [Q_{\textsc{Ch}}] \setminus \{r\}$, $\mathcal{B}$ returns $h_i \xleftarrow{\$} \mathbb{Z}_p$. On the $r$'th query, $\mathcal{B}$ submits a query to the challengers $\textsc{Ch}$ oracle, and forwards the response to $\mathcal{A}$. At some point $\mathcal{A}$ will submit a forgery

$$C \in \{(A_i^{h_k} \cdot R_k)^{a_j} \mid (i,j,k) \in ([n] \setminus \mathcal{L}_A)^2 \times [Q_{\textsc{Ch}}] \wedge (i \neq j)\}.$$

Then we have that $\Pr[k = r] \leq \frac{1}{Q_{\textsc{Ch}}}$, in which case $\mathcal{B}$ forwards $C$ and wins the game.

## A.2 Proofs of Other Useful Lemmas

Lemmas 1 to 3 and Lemma 9 are about the relation among $\mathsf{GapCDH}$, $\mathsf{CorrGapCDH}$ and $\mathsf{CorrAGapCDH}$. These lemmas complete Figure 5 of Section 3. We give their proofs here.

*Proof (of Lemma 1).* $\mathcal{B}$ receives a tuple $(A_1, \ldots, A_n)$ from the $\mathsf{CorrCRGapCDH}_{n,1,Q_{\text{DDH}}}$ challenger, and forwards it to adversary $\mathcal{A}$. When $\mathcal{A}$ submits a query to CORR or DDH, $\mathcal{B}$ forwards it to the appropriate $\mathsf{CorrCRGapCDH}_{n,1,Q_{\text{DDH}}}$ oracle, and returns the response. At some point, $\mathcal{A}$ will output a forgery $C$. We assume that the forgery is valid, meaning that $C = A_i^{a_j}$ for $i \neq j$ and $i, j \notin \mathcal{L}_A$. The reduction queries $\text{CH}(R = 1)$, and receives a challenge $h$. Finally, $\mathcal{B}$ submits a forgery $C^h = (A_i^{a_j})^h = (A_i^h \cdot 1)^{a_j}$, which is clearly a valid forgery in the $\mathsf{CorrCRGapCDH}_{n,1,Q_{\text{DDH}}}$ game.

*Proof (of Lemma 2).* Let $\mathcal{A}$ be an adversary that solves the $n$-$\mathsf{CorrGapCDH}$ problem. Given a $\mathsf{GapCDH}$ instance $(B_1, B_2) := (g^{b_1}, g^{b_2}) \in \mathbb{G}^2$ and the oracle DDH, $\mathcal{B}$ guesses two distinct $i^*, j^*$ from $[n]$ and define $(A_{i^*}, A_{j^*}) := (B_1, B_2)$ and, for $i \in [n] \setminus \{i^*, j^*\}$, $\mathcal{B}$ chooses $a_i \xleftarrow{\$} \mathbb{Z}_p$ and compute $A_i := g^{a_i}$. Then $\mathcal{B}$ calls $\mathcal{A}(A_1, \ldots, A_n)$ and answer $\mathcal{A}$'s oracle queries in a straightforward way:

– Upon $\text{CORR}(i \in [n])$, if $i = i^*$ or $i = j^*$, $\mathcal{B}$ aborts; otherwise, $\mathcal{B}$ returns $a_i$.

– Upon $\text{DDH}(X, Y, Z)$, $\mathcal{B}$ answers with the Boolean value $[\![Z = Y^{\mathsf{DL}_g(X)}]\!]$.

When $\mathcal{A}$ outputs it forgery $C$ and terminates, if $\mathcal{B}$ guessed $(i^*, j^*)$ correctly and $C$ is valid, namely, $C = A_{i^*}^{a_{j^*}} = B_1^{b_2}$, then $\mathcal{B}$ submits $C$ to break the $\mathsf{GapCDH}$ problem; otherwise, $\mathcal{B}$ aborts. Thus, $\mathcal{B}$ wins if it guesses $i^*, j^*$ correctly and $\mathcal{A}$ wins. Moreover, since two distinct $i^*, j^*$ are chosen randomly from $[n]$, the probability that $\mathcal{B}$ correctly guesses these indices is bounded by $1/n^2$. Hence, we arrive at $\text{Adv}^{\mathsf{GapCDH}}_{Q_{\text{DDH}}}(\mathcal{B}) \geq \frac{1}{n^2} \text{Adv}^{\mathsf{CorrGapCDH}}_{n, Q_{\text{DDH}}}(\mathcal{A})$.

**Lemma 9** **($\mathsf{GapCDH} \to n$-$\mathsf{GapCDH}$).** *For any adversary $\mathcal{A}$ against $n$-$\mathsf{GapCDH}$ $(n > 2)$, there exists an adversary $\mathcal{B}$ against $\mathsf{GapCDH}$ with*

$$\text{Adv}^{\mathsf{GapCDH}}_{n, Q_{\text{DDH}}}(\mathcal{A}) \leq 2\text{Adv}^{\mathsf{GapCDH}}_{2, Q_{\text{DDH}}}(\mathcal{B}).$$

*Proof.* We prove this tight implication by using a re-randomization argument. $\mathcal{B}$ is constructed in the following way to break the $\mathsf{GapCDH}$ assumption: Upon receiving $(B_1, B_2)$ from the $\mathsf{GapCDH}$ challenger, for $i \in [n]$, $\mathcal{B}$ chooses $r_i \xleftarrow{\$} \mathbb{Z}_p$, flips a random coin $\delta_i \in \{0, 1\}$, and computes $A_i := B_1 \cdot g^{r_i}$ if $\delta_i = 0$ or $A_i := B_2 \cdot g^{r_i}$ if $\delta_i = 1$. Then $\mathcal{B}$ calls the adversary $\mathcal{A}(A_1, \ldots, A_n)$. Queries to $\text{DDH}(X, Y, Z)$ are forwarded to the challengers DDH oracle.

Eventually $\mathcal{A}$ outputs its forgery $C$. If $C = A_i^{a_j}$ and $\delta_i \neq \delta_j$, then $\mathcal{B}$ breaks the $\mathsf{GapCDH}$ assumption with $C' := C/(B_1^{r_j} B_2^{r_i} g^{r_i r_j})$, assuming $\delta_i = 0$ and $\delta_j = 1$ w.l.o.g.. We note that $\delta_i$ and $\delta_j$ are two independent random coins and thus $\Pr[\delta_i \neq \delta_j] = \Pr[(\delta_i, \delta_j) = (0, 1)] + \Pr[(\delta_i, \delta_j) = (1, 0)] = 1/2$. This concludes the lemma.

*Proof (of Lemma 3).* We construct a reduction $\mathcal{B}$ as follows: $\mathcal{B}$ receives a $\mathsf{GapCDH}$ instance $(B_1, B_2) := (g^{b_1}, g^{b_2})$ and guesses an index $i^* \xleftarrow{\$} [n_1]$ on which $\mathcal{A}$ will output a forgery. $\mathcal{B}$ defines $A_{i^*} := B_1$ and for $i \in [n_1] \setminus \{i^*\}$ $\mathcal{B}$ chooses $a_i \xleftarrow{\$} \mathbb{Z}_p$ and computes $A_i := g^{a_i}$. For $i \in [n_1 + 1, n]$ $\mathcal{B}$ re-randomizes $B_2$ to get $A_i$ as in Lemma 9, namely, it chooses $r_i \xleftarrow{\$} \mathbb{Z}_p$ and computes $A_i := B_2 \cdot g^{r_i}$. Then $\mathcal{B}$ calls $\mathcal{A}$ with $(A_1, \ldots, A_n)$ and answers $\mathcal{A}$'s oracle queries as follow:

– Upon $\text{DDH}(X, Y, Z)$, $\mathcal{B}$ forwards it to the corresponding $\mathsf{GapCDH}$ oracle.

– Upon $\text{CORR}_{n_1}(i \in n_1)$, if $i = i^*$ then $\mathcal{B}$ aborts; else, $\mathcal{B}$ stores $i$ in $\mathcal{L}_A$ and returns $a_i$.

Eventually, $\mathcal{A}$ outputs its forgery $C$ and terminates. If $C$ is a valid forgery and $\mathcal{B}$ guesses $i^*$ correctly, then $C = A_{i^*}^{b_2 + r_{j^*}} = B_1^{b_2 + r_{j^*}}$ and $\mathcal{B}$ returns $C' := C/B_1^{r_{j^*}}$ as its forgery to $\mathsf{GapCDH}$. We note that the probability that $i^*$ is a correct guess is bounded by $1/n_1$, which concludes the proof.

## B Proof of Theorem 3

*Proof.* Let $\mathcal{A}$ be an adversary against $\mathsf{IND\text{-}wFS}$ security of $\mathsf{X3DH}^-$, where $N$ is the number of parties, $S$ is the maximum number of sessions that $\mathcal{A}$ establishes and $T$ is the maximum number of test sessions. Consider the sequence of games in Figure 14.

GAME $G_0$. This is the original $\mathsf{IND\text{-}wFS}$ game. As in Equation (4), we implicitly assume that all long-term keys and all messages output by $\text{SESSION}_\text{I}$ and $\text{SESSION}_\text{R}$ are different. If such a collision happens, the game will abort. Using the birthday paradox, the probability for that can be upper bounded by $(N + S)^2/(2p)$ as there are $N$ long-term key pairs and at most $S$ messages, where exponents are chosen

```
GAMES $G_0$, $G_1$, $\boxed{G_2}$
00 $\mathrm{cnt_P} := N$
01 for $n \in [N]$
02     $a_n \overset{\$}{\leftarrow} \mathbb{Z}_p$; $A_n := g^{a_n}$
03     $(\mathsf{pk}_n, \mathsf{sk}_n) := (A_n, a_n)$
04 $b \overset{\$}{\leftarrow} \{0,1\}$
05 $b' \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pk}_1, \cdots, \mathsf{pk}_N)$
06 for $\mathrm{sID}^* \in \mathcal{S}$
07     if $\textsc{Fresh}(\mathrm{sID}^*) =$ false
08         return $b$
09     if $\textsc{Valid}(\mathrm{sID}^*) =$ false
10         return $b$
11 return $[\![b = b']\!]$

$\textsc{Session}_{\mathrm{R}}((i,r) \in [\mathrm{cnt_P}] \times [N])$
12 $\mathrm{cnt_S}$ ++
13 $\mathrm{sID} := \mathrm{cnt_S}$
14 $(\mathrm{init[sID]}, \mathrm{resp[sID]}) := (i, r)$
15 $\mathrm{type[sID]} :=$ "Re"
16 $y \overset{\$}{\leftarrow} \mathbb{Z}_p$; $Y := g^y$
17 $(R[\mathrm{sID}], \mathrm{state[sID]}) := (Y, y)$
18 return $(\mathrm{sID}, Y)$

$\textsc{Der}_{\mathrm{R}}(\mathrm{sID} \in [\mathrm{cnt_S}], X)$
19 if $\mathrm{sKey[sID]} \neq \bot$ or $\mathrm{type[sID]} \neq$ "Re"
20     return $\bot$
21 $(i, r) := (\mathrm{init[sID]}, \mathrm{resp[sID]})$
22 $(Y, y) := (R[\mathrm{sID}], \mathrm{state[sID]})$
23 if $\exists \mathrm{sID}'$ s.t. $(\mathrm{type[sID']}, I[\mathrm{sID}']) =$
       ("In", $X$)
24     $\mathcal{P} := \mathcal{P} \cup \{\mathrm{sID}\}$
25 $\mathrm{ctxt} := (A_i, A_r, X, Y)$
26 $K := \mathsf{H}(\mathrm{ctxt}, A_i^y, X^{a_r}, X^y)$
27 $(I[\mathrm{sID}], \mathrm{sKey[sID]}) := (X, K)$
28 return $\varepsilon$

$\textsc{Session}_{\mathrm{I}}((i,r) \in [N] \times [\mathrm{cnt_P}])$
29 $\mathrm{cnt_S}$ ++
30 $\mathrm{sID} := \mathrm{cnt_S}$
31 $(\mathrm{init[sID]}, \mathrm{resp[sID]}) := (i, r)$
32 $\mathrm{type[sID]} :=$ "In"
33 $x \overset{\$}{\leftarrow} \mathbb{Z}_p$; $X := g^x$
34 $(I[\mathrm{sID}], \mathrm{state[sID]}) := (X, x)$
35 return $(\mathrm{sID}, X)$

$\textsc{Der}_{\mathrm{I}}(\mathrm{sID} \in [\mathrm{cnt_S}], Y)$
36 if $\mathrm{sKey[sID]} \neq \bot$ or $\mathrm{type[sID]} \neq$ "In"
37     return $\bot$
38 $(i, r) := (\mathrm{init[sID]}, \mathrm{resp[sID]})$
39 $(X, x) := (I[\mathrm{sID}], \mathrm{state[sID]})$
40 if $\exists \mathrm{sID}'$ s.t. $(\mathrm{type[sID']}, R[\mathrm{sID}']) =$
       ("Re", $Y$)
41     $\mathcal{P} := \mathcal{P} \cup \{\mathrm{sID}\}$
42 $\mathrm{ctxt} := (A_i, A_r, X, Y)$
43 $K := \mathsf{H}(\mathrm{ctxt}, Y^{a_i}, A_r^x, Y^x)$
44 $(R[\mathrm{sID}], \mathrm{sKey[sID]}) := (Y, K)$
45 return $\varepsilon$

$\textsc{Test}(\mathrm{sID})$
46 if $\mathrm{sID} \in \mathcal{S}$ return $\bot$          //already tested
47 if $\mathrm{sKey[sID]} = \bot$ return $\bot$
48 $\mathcal{S} := \mathcal{S} \cup \{\mathrm{sID}\}$
49 $K_0^* := \mathrm{sKey[sID]}$
50 if $\mathrm{sID} \in \mathcal{P}$
51     $K_0^* := \mathrm{sKey[sID]}$
52 else
53     $K_0^* \overset{\$}{\leftarrow} \mathcal{K}$
54 $\boxed{K_0^* \overset{\$}{\leftarrow} \mathcal{K}}$
55 $K_1^* \overset{\$}{\leftarrow} \mathcal{K}$
56 return $K_b^*$

$\mathsf{H}(A_i, A_r, X, Y, Z_1, Z_2, Z_3)$
57 if $\mathsf{H}[A_i, A_r, X, Y, Z_1, Z_2, Z_3] = K$
58     return $K$
59 else
60     $K \overset{\$}{\leftarrow} \mathcal{K}$
61     $\mathsf{H}[A_i, A_r, X, Y, Z_1, Z_2, Z_3] := K$
62     return $K$
```

**Fig. 14.** Games $G_0$-$G_2$ for the proof of Theorem 3. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\textsc{Session}_{\mathrm{I}}, \textsc{Session}_{\mathrm{R}}, \textsc{Der}_{\mathrm{I}}, \textsc{Der}_{\mathrm{R}}, \textsc{Reveal}, \textsc{Corrupt}, \textsc{RegisterLTK}, \textsc{Test}, \mathsf{H}\}$, where RegisterLTK, Corrupt and Reveal are defined as in the original IND-wFS game (Fig. 8). $G_0$ implicitly assumes that no long-term keys or messages generated by the experiment collide.

uniformly at random from $\mathbb{Z}_p$. This rules out attack (0a.), as there will be no two sessions having the same transcript. We get

$$\Pr[\mathsf{IND\text{-}wFS}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_0^{\mathcal{A}} \Rightarrow 1] + \frac{(N+S)^2}{2p} \quad .$$

<u>Game $G_1$</u>. In game $G_1$, the challenge oracle Test outputs a uniformly random key for test sessions which will not have a matching session. Therefore, we initialize a set $\mathcal{P}$ and each time $\textsc{Der}_{\mathrm{I}}$ or $\textsc{Der}_{\mathrm{R}}$ is called, we check if there is a potential matching session (lines 23, 40), i.e., the input message was output by $\textsc{Session}_{\mathrm{I}}$ or $\textsc{Session}_{\mathrm{R}}$. If this is the case, we add the session ID to the set (lines 24, 41). A Test query on such an sID will behave exactly as in $G_0$, whereas for the other sessions, it outputs a random key independently of bit $b$ (line 53). Similar to Equation (5), it holds that

$$\left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_0^{\mathcal{A}} \Rightarrow 1]\right| = \frac{1}{2}\left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b = 0] - \Pr[G_0^{\mathcal{A}} \Rightarrow 1 \mid b = 0]\right| \ .$$

```
$\mathcal{B}^{\text{CORR},\text{DDH}}(A_1, ..., A_N, B_1, ..., B_S)$
00 $\text{cnt}_\text{P} := N$
01 for $n \in [N]$
02     $(\mathsf{pk}_n, \mathsf{sk}_n) := (A_n, \perp)$
03 $b \xleftarrow{\$} \{0,1\}$
04 $b' \leftarrow \mathcal{A}^{\text{O}}(\mathsf{pk}_1, \cdots, \mathsf{pk}_N)$
05 for $\text{sID}^* \in \mathcal{S}$
06     if $\text{FRESH}(\text{sID}^*) = \textbf{false}$
07         return $b$
08     if $\text{VALID}(\text{sID}^*) = \textbf{false}$
09         return $b$
10 return $C \in \mathsf{Win}$ (see text)

$\text{SESSION}_\text{R}((i,r) \in [\text{cnt}_\text{P}] \times [N])$
11 $\text{cnt}_\text{S}$ ++
12 $\text{sID} := \text{cnt}_\text{S}$
13 $(\text{init}[\text{sID}], \text{resp}[\text{sID}]) := (i, r)$
14 $\text{type}[\text{sID}] :=$ “Re”
15 $Y := B_\text{sID}$
16 $(R[\text{sID}], \text{state}[\text{sID}]) := (Y, \perp)$
17 return $(\text{sID}, Y)$
```

```
$\text{SESSION}_\text{I}((i,r) \in [N] \times [\text{cnt}_\text{P}])$
18 $\text{cnt}_\text{S}$ ++
19 $\text{sID} := \text{cnt}_\text{S}$
20 $(\text{init}[\text{sID}], \text{resp}[\text{sID}]) := (i, r)$
21 $\text{type}[\text{sID}] :=$ “In”
22 $X := B_\text{sID}$
23 $(I[\text{sID}], \text{state}[\text{sID}]) := (X, \perp)$
24 return $(\text{sID}, X)$

$\text{CORRUPT}(n \in [N])$
25 $\text{corrupted}[n] := \textbf{true}$
26 $a_n \leftarrow \text{CORR}(n)$
27 $\mathsf{sk}_n := a_n$
28 return $\mathsf{sk}_n$

$\text{TEST}(\text{sID})$
29 if $\text{sID} \in \mathcal{S}$ return $\perp$
30 if $\text{sKey}[\text{sID}] = \perp$ return $\perp$
31 $\mathcal{S} := \mathcal{S} \cup \{\text{sID}\}$
32 $K_0^* := \text{sKey}[\text{sID}]$
33 if $\text{sID} \in \mathcal{P}$
34     $K_1^* := \text{sKey}[\text{sID}]$
35 else
36     $K_1^* \xleftarrow{\$} \mathcal{K}$
37 return $K_b^*$
```

**Fig. 15.** Adversary $\mathcal{B}$ against $(N+S, N)$-$\mathsf{CorrAGapCDH}$ for the proof of Theorem 3. $\mathcal{A}$ has access to oracles $\text{O} := \{\text{SESSION}_\text{I}, \text{SESSION}_\text{R}, \text{DER}_\text{I}, \text{DER}_\text{R}, \text{REVEAL}, \text{CORRUPT}, \text{REGISTERLTK}, \text{TEST}, \mathsf{H}\}$, where REGISTERLTK and REVEAL are defined as in game $G_1$. Oracles $\text{DER}_\text{I}$, $\text{DER}_\text{R}$ and $\mathsf{H}$ are defined as for adversary $\mathcal{B}$ in Figure 11. Lines written in blue color highlight how $\mathcal{B}$ simulates $G_0$ and $G_1$, respectively.

We construct adversary $\mathcal{B}$ against $(N+S, N)$-$\mathsf{CorrAGapCDH}$ in Figure 15. $\mathcal{B}$ gets as input $(N+S)$ group elements and has access to oracles CORR and DDH. The first $N$ group elements $(A_1, ..., A_N)$ are used as public keys for the parties $\mathsf{P}_1, ..., \mathsf{P}_N$ (line 02). The remaining group elements $(B_1, ..., B_S)$ will be used as messages output by $\text{SESSION}_\text{I}$ and $\text{SESSION}_\text{R}$. This means that whenever $\mathcal{A}$ initiates a session sID, $\mathcal{B}$ increments the session counter and outputs the next group element $B_\text{sID}$ (lines 15, 22). Whenever $\mathcal{A}$ calls $\text{DER}_\text{I}$ or $\text{DER}_\text{R}$, $\mathcal{B}$ behaves exactly as adversary $\mathcal{B}$ in Figure 11. In particular, it keeps the session keys consistent with the random oracle $\mathsf{H}$ using its DDH oracle. Whenever $\mathcal{A}$ corrupts a party $\mathsf{P}_n$, $\mathcal{B}$ queries its own oracle CORR on $n$ (line 26) and outputs the corresponding exponent.

The TEST oracle outputs the real session key for all queries if $b = 0$ (line 32). If $b = 1$, it outputs a random key for all those test session that will not have a matching session (line 36), and the real session key otherwise (line 34). Due to the exclusion of collisions, a particular (test) session cannot be recreated, i.e., the adversary cannot create two sessions sID, sID$'$ of the same type that compute the same session key. Thus, the only way to distinguish the two games is to query the random oracle on the correct input for a test session which does not have a matching session. Next we show that if this happens, $\mathcal{B}$ is able to output a solution $C \in \mathsf{Win}$ to the $\mathsf{CorrAGapCDH}$ problem.

Let $\text{sID}^* \in \mathcal{S}$ be such a test session and $\mathsf{H}[A_{i^*}, A_{r^*}, X^*, Y^*, Z_1^*, Z_2^*, Z_3^*, 1] = \text{sKey}[\text{sID}^*]$ be the corresponding entry in the list of hash queries. $\mathcal{B}$ has to find this query in the list and depending on which reveal queries $\mathcal{A}$ has made (i.e., which attack was performed), $\mathcal{B}$ returns either $Z_1^*$, $Z_2^*$ or $Z_3^*$ as described below. Therefore, we will now argue that for those attacks in Table 2 which consider non-matching sessions, there will be a valid solution for $\mathsf{CorrAGapCDH}$.

ATTACK (13.). The test session is of type “In” and $\mathcal{A}$ has queried the initiator's long-term secret keys $a_{i^*}$. Furthermore, message $X^*$ is chosen by the reduction $\mathcal{B}$ as $B_{\text{sID}^*}$, whereby $Y^*$ is chosen by $\mathcal{A}$. In order to distinguish the session key, $\mathcal{A}$ has to compute $Z_2^* = \mathsf{DH}(A_{r^*}, X^*) = \mathsf{DH}(A_{r^*}, B_{\text{sID}^*})$ which is a correct solution for $\mathsf{CorrAGapCDH}$.

ATTACK (16.). The test session is of type “Re” and $\mathcal{A}$ has queried the responder's long-term secret keys $a_{r^*}$. Furthermore, message $Y^*$ is chosen by the reduction $\mathcal{B}$ as $B_{\text{sID}^*}$, whereby $X^*$ is chosen by $\mathcal{A}$. In order to distinguish the session key, $\mathcal{A}$ has to compute $Z_1^* = \mathsf{DH}(A_{i^*}, Y^*) = \mathsf{DH}(A_{i^*}, B_{\text{sID}^*})$ which is a correct solution for $\mathsf{CorrAGapCDH}$.

The number of queries to the DDH oracle is upper bounded by $3 \cdot Q_{\mathsf{H}}$. Thus,

$$\left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b=0] - \Pr[G_0^{\mathcal{A}} \Rightarrow 1 \mid b=0]\right| \leq \mathrm{Adv}^{\mathsf{CorrAGapCDH}}_{N+S,\, N,\, 3Q_{\mathsf{H}}}(\mathcal{B}) \ .$$

<u>GAME $G_2$</u>. In game $G_2$, the challenge oracle TEST always outputs a uniformly random key, independent from the bit $b$ (Fig. 14, line 54). As $\Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b=1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b=1]$, it holds that

$$\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| = \frac{1}{2}\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b=0] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b=0]\right| .$$

We construct adversary $\mathcal{C}$ against $S$-$\mathsf{GapCDH}$ in Figure 16. $\mathcal{C}$ gets as input $S$ group elements $(B_1, ..., B_S)$ and has access to oracle DDH. The long-term key pairs are chosen by $\mathcal{C}$ (line 02). The input elements will be used as messages output by SESSION$_{\mathsf{I}}$ and SESSION$_{\mathsf{R}}$ as in adversary $\mathcal{B}$ described before. Again, we use a flag $f$ to identify correct queries and thus reduce the number of queries to DDH. Whenever $\mathcal{A}$ calls DER$_{\mathsf{I}}$ or DER$_{\mathsf{R}}$, $\mathcal{C}$ first computes the two Diffie-Hellman tuples which use the long-term secret keys (lines 19, 37). For the third one, it checks the list of random oracle queries for an entry with $f = 1$ (lines 20, 38) and outputs the corresponding session key. If this is not the case, it checks if there is an entry with an unknown Diffie-Hellman tuple (lines 22, 40) or chooses a session key uniformly at random (lines 25, 43) and adds such an entry to the list. If $\mathcal{A}$ issues a random oracle query which has not been asked before, $\mathcal{C}$ checks if the third Diffie-Hellman tuple is correct using the DDH oracle (line 59). In this case, it sets the flag $f$ to 1. Furthermore, if there is an entry with an unknown value, it updates the entry with the correct value (line 61) and outputs the corresponding key. Otherwise, $f$ is set to 0. $\mathcal{C}$ chooses a key uniformly at random (line 66), adds an entry with $f$ to the list and outputs the key.

The TEST oracle outputs a random key for all queries if $b = 1$ (line 55). If $b = 0$, it outputs a random key for all those test session that will not have a matching session (line 54), and the real session key otherwise (line 52). As a particular (test) session cannot be recreated, the only way to distinguish the two games is to query the random oracle on the correct input for a test session which is in the set $\mathcal{P}$ of potential matching sessions. Let $\mathrm{sID}^* \in \mathcal{P}$ be such a test session and $\mathsf{H}[A_{i^*}, A_{r^*}, X^*, Y^*, Z_1^*, Z_2^*, Z_3^*, 1] = \mathrm{sKey}[\mathrm{sID}^*]$ be the corresponding entry for such a test session in the list of hash queries. At this point, it does not matter whether the adversary completes the potential matching session or not. What matters is that both messages $X^*$ and $Y^*$ are chosen by the reduction $\mathcal{B}$ as $B_{\mathrm{sID}^*}$ and $B_{\mathrm{sID}'}$. We assume that the adversary may query both long-term keys thus covering attacks (1.)+(2.) of Table 2. In order to distinguish the session key, $\mathcal{A}$ has to compute $Z_3^* = \mathsf{DH}(X^*, Y^*) = \mathsf{DH}(B_{\mathrm{sID}^*}, B_{\mathrm{sID}'})$ which is a correct solution $C \in \mathsf{Win}$ to the $S$-$\mathsf{GapCDH}$ problem.

The number of queries to the DDH oracle is upper bounded by $Q_{\mathsf{H}}$. Thus,

$$\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b=0] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b=0]\right| \leq \mathrm{Adv}^{\mathsf{GapCDH}}_{S,\, Q_{\mathsf{H}}}(\mathcal{C}) \ .$$

Finally, the output of the TEST oracle in $G_2$ is independent of the bit $b$, so we have

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2} \ .$$

Collecting the probabilities yields the bound stated in Theorem 3.

```
$\mathcal{C}^{\text{DDH}}(B_1, ..., B_S)$
cnt$_\text{P}$ $:= N$
for $n \in [N]$
    $a_n \xleftarrow{\$} \mathbb{Z}_p$; $A_n := g^{a_n}$
    $(\mathsf{pk}_n, \mathsf{sk}_n) := (A_n, a_n)$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\text{O}}(\mathsf{pk}_1, \cdots, \mathsf{pk}_N)$
for sID$^* \in \mathcal{S}$
    if FRESH(sID$^*$) = false
        return $b$
    if VALID(sID$^*$) = false
        return $b$
return $C \in$ Win (see text)
```

```
DER$_\text{R}$(sID $\in$ [cnt$_\text{S}$], $X$)
if sKey[sID] $\neq \bot$ or type[sID] $\neq$ "Re"
    return $\bot$
$(i, r) :=$ (init[sID], resp[sID])
$Y := R$[sID]
ctxt $:= (A_i, A_r, X, Y)$
if $\exists$sID$'$ s.t. (type[sID$'$], $I$[sID$'$]) = ("In", $X$)
    $\mathcal{P} := \mathcal{P} \cup$ {sID}
$(Z_1, Z_2) := (Y^{a_i}, X^{a_r})$
if $\exists Z_3$ s.t. H[ctxt, $Z_1, Z_2, Z_3, 1] = K$
    sKey[sID] $:= K$
elseif H[ctxt, $Z_1, Z_2, \bot, \bot] = K$
    sKey[sID] $:= K$
else
    $K \xleftarrow{\$} \mathcal{K}$
    H[ctxt, $Z_1, Z_2, \bot, \bot] := K$
    sKey[sID] $:= K$
($I$[sID], sKey[sID]) $:= (X, K)$
return $\varepsilon$
```

```
DER$_\text{I}$(sID $\in$ [cnt$_\text{S}$], $Y$)
if sKey[sID] $\neq \bot$ or type[sID] $\neq$ "In"
    return $\bot$
$(i, r) :=$ (init[sID], resp[sID])
$X := I$[sID]
ctxt $:= (A_i, A_r, X, Y)$
if $\exists$sID$'$ s.t. (type[sID$'$], $R$[sID$'$]) = ("Re", $Y$)
    $\mathcal{P} := \mathcal{P} \cup$ {sID}
$(Z_1, Z_2) := (Y^{a_i}, X^{a_r})$
if $\exists Z_3$ s.t. H[ctxt, $Z_1, Z_2, Z_3, 1] = K$
    sKey[sID] $:= K$
elseif H[ctxt, $Z_1, Z_2, \bot, \bot] = K$
    sKey[sID] $:= K$
else
    $K \xleftarrow{\$} \mathcal{K}$
    H[ctxt, $Z_1, Z_2, \bot, \bot] := K$
    sKey[sID] $:= K$
($R$[sID], sKey[sID]) $:= (Y, K)$
return $\varepsilon$
```

```
TEST(sID)
if sID $\in \mathcal{S}$ return $\bot$
if sKey[sID] $= \bot$ return $\bot$
$\mathcal{S} := \mathcal{S} \cup$ {sID}
if sID $\in \mathcal{P}$
    $K_0^* :=$ sKey[sID]
else
    $K_0^* \xleftarrow{\$} \mathcal{K}$
$K_1^* \xleftarrow{\$} \mathcal{K}$
return $K_b^*$
```

```
H$(A_i, A_r, X, Y, Z_1, Z_2, Z_3)$
if H$[A_i, A_r, X, Y, Z_1, Z_2, Z_3, \cdot\,] = K$
    return $K$
if DDH$(X, Y, Z_3) = 1$
    $f := 1$
    if H$[A_i, A_r, X, Y, Z_1, Z_2, \bot, \bot] = K$
        replace $(\bot, \bot)$ with $(Z_3, f)$
        return $K$
else
    $f := 0$
$K \xleftarrow{\$} \mathcal{K}$
H$[A_i, A_r, X, Y, Z_1, Z_2, Z_3, f] := K$
return $K$
```

**Fig. 16.** Adversary $\mathcal{C}$ against $S$-GapCDH for the proof of Theorem 3. $\mathcal{A}$ has access to oracles O := {SESSION$_\text{I}$, SESSION$_\text{R}$, DER$_\text{I}$, DER$_\text{R}$, REVEAL, CORRUPT, REGISTERLTK, TEST, H}, where SESSION$_\text{I}$ and SESSION$_\text{R}$ are defined as for adversary $\mathcal{B}$ in Figure 15. REGISTERLTK, REVEAL and CORRUPT are defined as in game $G_1$. Lines written in blue color highlight how $\mathcal{C}$ simulates $G_1$ and $G_2$, respectively.