Password-Authenticated Key Exchange from Group Actions

Michel Abdalla[1,2], Thorsten Eisenhofer[3], Eike Kiltz[3], Sabrina Kunzweiler[3], Doreen Riepel[3]

[1] DFINITY, Zürich, Switzerland
[2] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
michel.abdalla@ens.fr
[3] Ruhr-Universität Bochum, Bochum, Germany
{thorsten.eisenhofer,eike.kiltz,sabrina.kunzweiler,doreen.riepel}@rub.de

Abstract. We present two provably secure password-authenticated key exchange (PAKE) protocols based on a commutative group action. To date the most important instantiation of isogeny-based group actions is given by CSIDH. To model the properties more accurately, we extend the framework of cryptographic group actions (Alamati et al., ASIACRYPT 2020) by the ability of computing the quadratic twist of an elliptic curve. This property is always present in the CSIDH setting and turns out to be crucial in the security analysis of our PAKE protocols.
Despite the resemblance, the translation of Diffie-Hellman based PAKE protocols to group actions either does not work with known techniques or is insecure ("How not to create an isogeny-based PAKE", Azarderakhsh et al., ACNS 2020). We overcome the difficulties mentioned in previous work by using a "bit-by-bit" approach, where each password bit is considered separately.
Our first protocol $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ can be executed in a single round. Both parties need to send two set elements for each password bit in order to prevent offline dictionary attacks. The second protocol $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$ requires only one set element per password bit, but one party has to send a commitment on its message first. We also discuss different optimizations that can be used to reduce the computational cost. We provide comprehensive security proofs for our base protocols and deduce security for the optimized versions.

Keywords: Password-authenticated key exchange, group actions, CSIDH

Table of Contents

1 Introduction

Password-authenticated key exchange (PAKE) enables two parties to securely establish a joint session key assuming that they only share a low-entropy secret known as the password. This reflects that passwords are often represented in short human-readable formats and are chosen from a small set of possible values, often referred to as dictionary.

Since the introduction of PAKE by Bellovin and Merritt [9], many PAKE protocols have been proposed, including SPEKE [22], SPAKE2 [4], J-PAKE [21] and CPace [20]. In particular over the last few years, the design and construction of PAKE protocols has attracted increasing attention, as the Crypto Forum Research Group (CFRG) which is part of the Internet Research Task Force (IETF) started a selection process to decide which PAKE protocols should be used in IETF protocols. Recently, CPace was selected as the recommended protocol for symmetric PAKE, where both parties share the same password.

Different models have been used to formally prove security of PAKE protocols, like indistinguishability-based models or the universal composability framework. In general, a PAKE protocol should resist offline and online dictionary attacks. On the one hand an adversary should not be able to perform an exhaustive search of the password offline. On the other hand, an active adversary should only be able to try a small number of passwords in one protocol execution. Furthermore, forward security ensures that session keys are still secure, even if the password is leaked at a later point in time. The same should hold if session keys are disclosed, which should not affect security of other session keys.

CSIDH and Group Actions. The PAKE protocols mentioned above are mostly based on a Diffie-Hellman key exchange in a prime order group. A promising post-quantum replacement is isogeny-based key exchange. The different isogeny-based protocols can be divided into two groups. On the one hand there are constructions based on commutative group actions on a set of elliptic curves. The first proposals by Couveignes [14], and Stolbunov and Rostovtsev [31] suggested to use the action of the class group $cl(\mathcal{O})$ on the set of $\mathbb{F}_q$-isomorphism classes of *ordinary* elliptic curves with endomorphism ring $\mathcal{O}$. In 2018, Castryck et al. showed that this idea can also be adapted to the class group action on the set of $\mathbb{F}_p$-isomorphism classes of *supersingular* elliptic curves [13]. The resulting scheme is called CSIDH and constitutes the first practical key exchange scheme based on class group actions.

In [14], Couveignes introduces *hard homogeneous spaces* - an abstract framework for group actions that models isogeny-based assumptions. This framework has been further refined by Alamati et al. in [5]. Using the abstract setting of *cryptographic group actions* the authors develop several new cryptographic primitives that can be instantiated with CSIDH. On the other hand there is the Supersingular Isogeny Diffie-Hellman (SIDH) protocol suggested by Jao and De Feo in 2011 [23]. Here, the set of $\mathbb{F}_{p^2}$-isomorphism classes of *supersingular* elliptic curves is considered. The endomorphism ring of a supersingular elliptic curve over $\mathbb{F}_{p^2}$ is non-commutative, hence protocols based on SIDH do not fall into the group action framework.

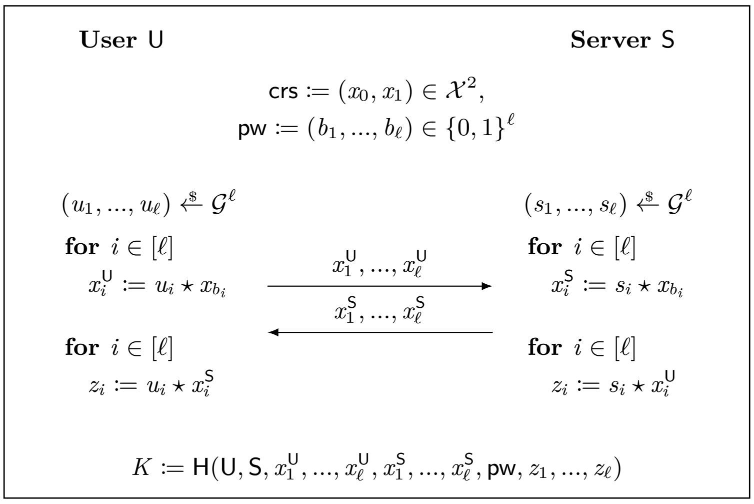
We now recall the framework of (restricted) effective group actions introduced in [5]. Throughout, $\mathcal{G}$ denotes a finite commutative group and $\mathcal{X}$ a set. We assume that $\mathcal{G}$ acts regularly on $\mathcal{X}$ via the operator $\star : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$. Regularity guarantees that for any $x, y \in \mathcal{X}$ there exists precisely one group element $g \in \mathcal{G}$ satisfying $y = g \star x$. Broadly speaking, we are interested in group actions, where evaluation is easy, but the "discrete logarithm problem" is hard. Expressed differently:

- Given $x \in \mathcal{X}$ and $g \in \mathcal{G}$, one can efficiently compute the set element $y = g \star x$.
- Given $x, y \in \mathcal{X}$, it is hard to find the element $g \in \mathcal{G}$ satisfying $y = g \star x$.

These properties facilitate the definition of a Diffie-Hellman key exchange. Let x be some fixed set element. Alice chooses a secret $g_A \in \mathcal{G}$ and publishes $y_A = g_A \star x$. Similarly Bob chooses $g_B \in \mathcal{G}$ and publishes $y_B = g_B \star x$. They can both compute the shared secret $y_{AB} = g_A \star y_B = g_B \star y_A$. The group action computational Diffie-Hellman problem ($\mathsf{GA\text{-}CDH}$) then states that given y_A and y_B, it is hard to compute y_{AB}. We refer to Section 3 for more precise definitions.

Contributions and Technical Details. Our main contributions are the two PAKE protocols $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ and $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$ based on commutative group actions. These are the first two provably secure PAKE protocols that are directly constructed from isogenies.

Group Actions with Twists. To date the most important instantiation of isogeny-based group actions is given by CSIDH. To model this situation more accurately, we suggest an enhancement of the framework



User U
Server S
$\mathsf{crs} := (x_0, x_1) \in \mathcal{X}^2$,
$\mathsf{pw} := (b_1, ..., b_\ell) \in \{0,1\}^\ell$
$(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
for $i \in [\ell]$
$x_i^{\mathsf{U}} := u_i \star x_{b_i}$
$x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}$
$x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}$
for $i \in [\ell]$
$z_i := u_i \star x_i^{\mathsf{S}}$
$(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
for $i \in [\ell]$
$x_i^{\mathsf{S}} := s_i \star x_{b_i}$
for $i \in [\ell]$
$z_i := s_i \star x_i^{\mathsf{U}}$
$K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}, x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}, \mathsf{pw}, z_1, ..., z_\ell)$

Fig. 1. First Attempt: Protocol $\mathsf{GA\text{-}PAKE}_\ell$.

which includes the ability of computing the quadratic twist of an elliptic curve efficiently. This property is inherent to CSIDH (cf. [13]) and it turns out to be crucial in the security analysis of our PAKE protocols. On the one hand, twisting allows us to construct an offline dictionary attack against our first natural PAKE attempt $\mathsf{GA\text{-}PAKE}_\ell$. Notably, this first protocol is secure for group actions where twisting is not possible efficiently. On the other hand, twists play an important role in various security reductions applied to prove the security of our new protocols $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ and $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$. Interestingly, this is also the case when twists are not part of any of the two problems involved in the reduction.

First attempt: $\mathsf{GA\text{-}PAKE}_\ell$. Our two secure PAKE protocols are modifications of $\mathsf{GA\text{-}PAKE}_\ell$. In order to illustrate the main idea behind the protocols, we describe $\mathsf{GA\text{-}PAKE}_\ell$ in more detail here. The protocol (Figure 1) can be seen as an adaption of the simple password exponential key exchange protocol SPEKE [22] to the group action setting. In SPEKE the password is used to hash to a generator of the group. Then the user and the server establish a session key following the Diffie-Hellman key exchange. Directly translating this protocol to the group action setting requires to hash the password to a random set element $x \in \mathcal{X}$. For isogeny-based group actions, this is still an open problem, hence (at the moment) a straight-forward translation of SPEKE is not possible (see also [6, §4.1]). In $\mathsf{GA\text{-}PAKE}_\ell$ we map the password to an ℓ-tuple of elements in $\mathcal{X}$ instead of hashing to one element. More precisely, two elements $\mathsf{crs} = (x_0, x_1) \in \mathcal{X}^2$ are fixed by a trusted party and a password $\mathsf{pw} = (b_1, \dots, b_\ell) \in \{0,1\}^\ell$ is mapped to the tuple $(x_{b_1}, \cdots, x_{b_\ell}) \in \mathcal{X}^\ell$. Then a Diffie-Hellman key exchange is performed with basis x_{b_i} for each $i \in [\ell]$. This means the user generates ℓ random group elements $u_1, \dots, u_\ell$ and computes the elements $x_1^{\mathsf{U}} = u_1 \star x_{b_1}, \dots, x_\ell^{\mathsf{U}} = u_\ell \star x_{b_\ell}$ which it sends to the server. Similarly, the server generates ℓ random group elements $s_1, \dots, s_\ell$ and computes $x_1^{\mathsf{S}} = s_1 \star x_{b_1}, \dots, x_\ell^{\mathsf{S}} = s_\ell \star x_{b_\ell}$ which it sends to the user. Note that the messages may be sent simultaneously in one round. Then both parties compute $z_i = u_i \star x_i^{\mathsf{S}} = s_i \star x_i^{\mathsf{U}}$ for each $i \in [\ell]$. Finally the session key K is computed as $K = \mathsf{H}(\mathsf{U}, \mathsf{S}, x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}, x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}, \mathsf{pw}, z_1, ..., z_\ell)$, where $\mathsf{H} : \{0,1\}^* \to \mathcal{K}$ is a hash function into the key space $\mathcal{K}$.

In Section 5, we present an offline dictionary attack against $\mathsf{GA\text{-}PAKE}_\ell$ for group actions with twists. This attack is not captured by the abstract group action framework defined in [5] which underlines the necessity of our suggested enhancement of the framework. Roughly speaking, the attack uses the fact that an attacker can choose its message in dependence on the other party's message. Using twists, it can then achieve that certain terms in the key derivation cancel out and the session key no longer depends on the other party's input.

Secure PAKE: $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ and $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$. The protocol $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ is a modified version of $\mathsf{GA\text{-}PAKE}_\ell$. Here security is achieved by doubling the message length in the first round of the protocol and tripling it in the key derivation. Intuitively the additional parts of the message can be viewed as an additional challenge for the key derivation that inhibits an attacker from choosing its message depending on the other party's message. The security of the protocol relies on a new computational assumption, $\mathsf{SqInv\text{-}GA\text{-}StCDH}$, in which the adversary needs to compute the square and the inverse of its input at the same time (cf. Definition 7, Theorem 1).

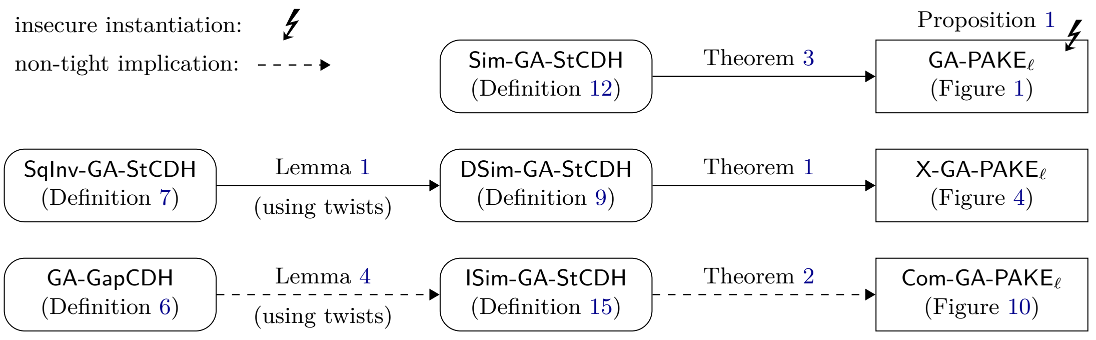

insecure instantiation:
non-tight implication:
Proposition 1
Sim-GA-StCDH (Definition 12)
Theorem 3
GA-PAKE$_\ell$ (Figure 1)
SqInv-GA-StCDH (Definition 7)
Lemma 1 (using twists)
DSim-GA-StCDH (Definition 9)
Theorem 1
X-GA-PAKE$_\ell$ (Figure 4)
GA-GapCDH (Definition 6)
Lemma 4 (using twists)
ISim-GA-StCDH (Definition 15)
Theorem 2
Com-GA-PAKE$_\ell$ (Figure 10)

Fig. 2. Overview of our security implications between assumptions (round boxes) and schemes (square boxes). Note that there exists an attack against protocol GA-PAKE$_\ell$ using twists which makes it insecure for CSIDH. Our two main protocols X-GA-PAKE$_\ell$ and Com-GA-PAKE$_\ell$ are proven secure under protocol-specific assumptions, but we also give reductions to simpler assumptions making use of the twisting property. Solid arrows denote tight reductions, dashed arrows non-tight reductions.

Protocol	$\lvert\mathsf{crs}\rvert$	Elements	Evaluations	Rounds	Assumption	Rew.	ROM
X-GA-PAKE$^{\mathsf{t}}_{\ell,N}$	2^{N-1}	$2\ell/N$	$5\ell/N$	1	SqInv-GA-StCDH	no	yes
$\hookrightarrow (\ell, N) = (128, 8)$	128	32	80				
Com-GA-PAKE$^{\mathsf{t}}_{\ell,N}$	2^{N-1}	ℓ/N (+1)	$2\ell/N$	3	Sq-GA-GapCDH	yes	yes
$\hookrightarrow (\ell, N) = (128, 8)$	128	16 (+1)	32				
OT-based$_\ell$ [27,12]	1	3ℓ ($+6\ell$)	11ℓ	4	GA-CDH	yes	yes
$\hookrightarrow \ell = 128$	1	384 (+768)	1408				
OT-based$_\ell$ [5,29,12]	4	$> \ell^2$	$> \ell^2$	3	GA-DDH + CCA PKE	no	no
$\hookrightarrow \ell = 128$	4	$> 16,000$	$> 16,000$				

Table 1. Overview of our two optimized protocols Com-GA-PAKE$^{\mathsf{t}}_{\ell,N}$ and X-GA-PAKE$^{\mathsf{t}}_{\ell,N}$ and comparison to the only other CSIDH-based constructions. All protocols use a bit-wise approach, i.e., passwords are treated as bitstrings of length ℓ. Sample values for $\ell = 128$ are marked in gray. "Elements" refers to the number of set elements (+ strings or symmetric ciphertexts) that each party has to send. "Evaluations" refers to the number of group action evaluations that each party has to perform. "Rew." indicates that rewinding is used to reduce to the assumption indicated in the table and GA-DDH refers to the group action decisional Diffie-Hellman problem.

The protocol Com-GA-PAKE$_\ell$ is a modification of GA-PAKE$_\ell$ as well. In order to achieve security against offline dictionary attacks, the protocol requires that the server sends a commitment before receiving the first message from the user. This prevents that any party chooses its message depending on the other party's message. We reduce the security of the protocol to the hardness of standard security assumptions in the isogeny-based setting (Theorem 2). An overview of our results is provided in Figure 2.

Optimizations. Both X-GA-PAKE$_\ell$ and Com-GA-PAKE$_\ell$ require to compute multiple group action evaluations. In the last section, we discuss two optimizations that can be used to reduce the number of evaluations and show that these do not affect the security of the protocols. The first makes a tradeoff between the size of the public parameters (the common reference string crs) and the number of elements that have to be sent as well as the group actions that have to be performed. The second optimization relies on the possibility to compute twists efficiently, which is yet another advantage of adding this property to the framework and which allows to decrease the size of the public parameters by a factor of 2. We denote the final optimizations by Com-GA-PAKE$^{\mathsf{t}}_{\ell,N}$ and X-GA-PAKE$^{\mathsf{t}}_{\ell,N}$, where N is a parameter for the crs size. If N equals 1, we omit it. An overview and example of the parameter choice is provided in Table 1.

Difficulties in constructing PAKE from Isogenies. Terada and Yoneyama [34] proposed isogeny-based PAKE based on the EKE approach. The basic idea is that the parties perform an SIDH or CSIDH key exchange where the messages are encrypted with the password. However, as shown in [6], these protocols are not only vulnerable to offline dictionary attacks, but a modified version is even vulnerable to man-in-the-middle attacks. The main reason for the insecurity is that the elliptic curves used in the key exchange and encrypted with the password are distinguishable from random bitstrings. An exhaustive search over all passwords just requires to check if the decrypted message is a valid curve.

Another proposal based on SIDH was made by Taraskin et al. [33]. In this protocol the password is used to obfuscate the auxiliary points that are exchanged during an SIDH key exchange. While their obfuscation method prevents a certain type of offline dictionary attack, the authors were not able to provide a security proof for their protocol. The same is true for a symmetric variant of the protocol proposed by Soukharev and Hess [32]. Until now these are the only PAKE protocol based on isogenies which are not broken.

As noted in [6], other popular Diffie-Hellman constructions may also not be directly translated into the isogeny setting. The main reason is that hashing into the set of supersingular elliptic curves is still an open problem. This approach is for example used in SPEKE. (However, we show how to non-trivially translate the idea.) Also the approach of J-PAKE seems difficult as in this scheme different public keys are combined to obtain certain "mixed" public keys. In isogeny-based protocols, the public keys are elliptic curves and there is no natural ring structure on the set of elliptic curves that would allow to combine two elliptic curves.

In the following, we elaborate known generic constructions of PAKE from hash proof systems (HPS) and oblivious transfer (OT). We explain that the only known isogeny-based HPS is not suitable for generic constructions. On the other hand, the isogeny-based OT protocols from the literature are suited for generic constructions. However, we show that the resulting PAKE protocols are less efficient than our new proposals.

Using the framework of cryptographic group actions, Alamati et al. construct a universal hash proof system [5, §4.1]. Their HPS is defined for the subset membership problem based on the DDH assumption for group actions. However, we need a different type of subset membership problem in order to construct PAKE. In particular, the framework introduced by Gennaro and Lindell [18] and that of follow-up works [19,24] uses an HPS for the language of ciphertexts of a public-key encryption scheme. More concretely, given a public key of the encryption scheme, a pair of message and ciphertext (m, c) is in the language of the HPS if c is a valid decryption of m (under the given public key). Note that the public evaluation of the HPS can use the encryption randomness as a witness. These kinds of HPS have been constructed for ElGamal and Cramer-Shoup encryption in the prime-order group setting (e.g., [10]), however it is less clear how this will work for group actions. We illustrate this for the simpler example of the "group action ElGamal" encryption scheme. The main obstacle here is that due to the limited structure we cannot simply encrypt the message by a one-time pad like operation (see for example [28]). Instead, one can additionally hash the set element that serves as the ElGamal KEM key and then encrypt the message via XOR. However, this destroys all structure and makes it hard to build a hash proof system for ciphertexts of this form. Therefore, we leave it as an interesting open problem to build such an HPS from group actions which can then be used to construct PAKE.

It is well known that PAKE can also be generically constructed from OT. In [12], Canetti et al. describe two different constructions: the first builds upon a UC-secure OT protocol to construct a UC-secure PAKE and the second uses a statistically receiver-private OT protocol to construct PAKE in a game-based security model. In both constructions, the password is interpreted as a bit string. In particular, for each individual password bit, the PAKE user and server run the OT protocol twice: once taking the role of the OT sender for randomly chosen messages and once taking the role of the OT receiver using the password bit to recover one of the messages chosen by the other party. Together with some additional overhead consisting of nonces and/or ciphertexts that need to be sent to compute the shared session key, this results in a PAKE protocol of at least three rounds. That means, even for round-optimal and efficient OT protocols, this approach makes the final construction quite inefficient. To compare against our protocols, we apply the compiler of [12] to the following two OT protocols.

– Alamati et al. propose a two-message statistically *sender*-private OT, however we can construct a similar receiver-private OT protocol based on their dual-mode public-key encryption scheme and the transformation given in [29]. The resulting OT protocol already uses a "bit-by-bit" approach, hence the resulting PAKE will have communication and computation complexity quadratic in the parameter ℓ.

– Recently, Lai et al. proposed a new very efficient CSIDH-based OT protocol using twists and the random oracle model [27]. However, in order to achieve active security the protocol needs four rounds.[4]

Additionally applying the generic PAKE compiler results in a protocol with complexity linear in ℓ. The efficiency of the generic constructions compared to our new protocols is given in Table 1. Note that the computational complexity of the second OT-based protocol as well as the complexity of our protocols is linear in the password length ℓ. However the constants are important for concrete instantiations. For $\ell = 128$ and $N = 8$, our optimized versions of $\mathsf{X\text{-}GA\text{-}PAKE}_{\ell,N}$ (resp. $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell,N}$) perform considerably better. In particular, each party then has to send 32 (resp. 16) set elements and perform 80 (resp. 32) group action evaluations. Whereas each party would have to send 384 set elements and perform 1408 group action evaluations in the OT-based protocol. Additionally, $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell}$ is the only one-round protocol, where both parties send simultaneous flows, which plays an important role for practical applications.

Open Problems and Future Work. Until now, protocols based on CSIDH or group actions that use search problems together with the random oracle model do not consider quantum access to the ROM [35,17,25,26,27]. Since PAKE proofs are already complex, we also did not prove security in the QROM. Although no reprogramming of the random oracle is necessary, the main difficulty in the QROM is to simulate the real session keys using the decision oracle. We leave this as future work. We believe that we can easily allow quantum access to the additional random oracle that is used in $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell}$ to commit on the message. In this case, the output is transferred classically in the first message flow such that extraction is possible using recently developed techniques [16].

As [27], we use rewinding to reduce the interactive assumption underlying $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell}$ to a standard assumption. An interesting open question is whether current techniques enabling quantum rewinding are applicable here.

Outline. Section 3 sets the framework for our paper. We introduce (restricted) effective group actions with twists and define the computational assumptions underlying the security of our protocols. In Section 4, we give some background on the security model that is used in the subsequent sections. In Section 5 we present our first attempt for a PAKE protocol, $\mathsf{GA\text{-}PAKE}_{\ell}$, and explain its security gap. Section 6 contains a thorough analysis of our new secure protocol $\mathsf{X\text{-}GA\text{-}PAKE}_{\ell}$. In Section 7 we present the protocol $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell}$ and sketch the security proof. A full proof is provided in Appendix E. Finally, we discuss possible optimizations of the protocols in Section 8.

2 Preliminaries

For integers m, n where $m < n$, $[m, n]$ denotes the set $\{m, m+1, ..., n\}$. For $m = 1$, we simply write $[n]$. For a set S, $s \xleftarrow{\$} S$ denotes that s is sampled uniformly and independently at random from S. $y \leftarrow \mathcal{A}(x_1, x_2, ...)$ denotes that on input $x_1, x_2, ...$ the probabilistic algorithm $\mathcal{A}$ returns y. $\mathcal{A}^{\mathrm{O}}$ denotes that algorithm $\mathcal{A}$ has access to oracle O. An adversary is a probabilistic algorithm. We will use code-based games, where $\Pr[\mathsf{G} \Rightarrow 1]$ denotes the probability that the final output of game G is 1.

3 (Restricted) Effective Group Actions (with Twists)

In this section we recall the definition of (restricted) effective group actions from [5], which provides an abstract framework to build cryptographic primitives relying on isogeny-based assumptions such as CSIDH. Moreover, we suggest an enhancement of this framework, by introducing (restricted) effective group actions with twists. This addition is essential for the security analysis of our new PAKE protocols.

Definition 1 (Group Action). *Let $(\mathcal{G}, \cdot)$ be a group with identity element $id \in \mathcal{G}$, and $\mathcal{X}$ a set. A map*

$$\star : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$$

is a group action if it satisfies the following properties:

[4] The original (three-round) version of this protocol was later found to have a (fixable) bug, cf. https://iacr.org/submit/files/slides/2021/eurocrypt/eurocrypt2021/20/slides.pdf.

1. *Identity:* $id \star x = x$ *for all* $x \in \mathcal{X}$.
2. *Compatibility:* $(g \cdot h) \star x = g \star (h \star x)$ *for all* $g, h \in \mathcal{G}$ *and* $x \in \mathcal{X}$.

Remark 1. Throughout this paper, we only consider group actions, where $\mathcal{G}$ is commutative. Moreover we assume that the group action is regular. This means that for any $x, y \in \mathcal{X}$ there exists precisely one $g \in \mathcal{G}$ satisfying $y = g \star x$.

Definition 2 (Effective Group Action). *Let* $(\mathcal{G}, \mathcal{X}, \star)$ *be a group action satisfying the following properties:*
1. *The group* $\mathcal{G}$ *is finite and there exist efficient (PPT) algorithms for membership and equality testing, (random) sampling, group operation and inversion.*
2. *The set* $\mathcal{X}$ *is finite and there exist efficient algorithms for membership testing and to compute a unique representation.*
3. *There exists a distinguished element* $\tilde{x} \in \mathcal{X}$ *with known representation.*
4. *There exists an efficient algorithm to evaluate the group action, i.e. to compute* $g \star x$ *given* g *and* x.

Then we call $\tilde{x} \in \mathcal{X}$ *the origin and* $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ *an effective group action* (EGA).

In practice, the requirements from the definition of EGA are often too strong. Therefore we will consider the weaker notion of restricted effective group actions.

Definition 3 (Restricted Effective Group Action). *Let* $(\mathcal{G}, \mathcal{X}, \star)$ *be a group action and let* $\boldsymbol{g} = (g_1, ..., g_n)$ *be a generating set for* $\mathcal{G}$. *Assume that the following properties are satisfied:*
1. *The group* $\mathcal{G}$ *is finite and* $n = poly(\log(\#\mathcal{G}))$.
2. *The set* $\mathcal{X}$ *is finite and there exist efficient algorithms for membership testing and to compute a unique representation.*
3. *There exists a distinguished element* $\tilde{x} \in \mathcal{X}$ *with known representation.*
4. *There exists an efficient algorithm that given* $g_i \in \boldsymbol{g}$ *and* $x \in \mathcal{X}$, *outputs* $g_i \star x$ *and* $g_i^{-1} \star x$.

Then we call $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ *a restricted effective group action* (REGA).

3.1 Isogeny-based REGAs

An important instantiation of REGAs is provided by isogeny-based group actions. We will focus on the CSIDH setting and present a refined definition of REGAs tailored to this situation.

Let p be a large prime of the form $p = 4 \cdot \ell_1 \cdots \ell_n - 1$, where the ℓ_i are small distinct odd primes. Fix the elliptic curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_p$. The curve E_0 is supersingular and its $\mathbb{F}_p$-rational endomorphism ring is $\mathcal{O} = \mathbb{Z}[\pi]$, where π is the Frobenius endomorphism. Let $\mathcal{E}\ell\ell_p(\mathcal{O})$ be the set of elliptic curves defined over $\mathbb{F}_p$, with endomorphism ring $\mathcal{O}$. The ideal class group $cl(\mathcal{O})$ acts on the set $\mathcal{E}\ell\ell_p(\mathcal{O})$, i.e., there is a map

$$\begin{aligned} \star : cl(\mathcal{O}) \times \mathcal{E}\ell\ell_p(\mathcal{O}) &\to \mathcal{E}\ell\ell_p(\mathcal{O}) \\ ([\mathfrak{a}], E) &\mapsto [\mathfrak{a}] \star E, \end{aligned}$$

satisfying the properties from Definition 1 [13, Theorem 7]. Moreover the analysis in [13] readily shows that $(cl(\mathcal{O}), \mathcal{E}\ell\ell_p(\mathcal{O}), \star, E_0)$ is indeed a REGA.

Elliptic curves in $\mathcal{E}\ell\ell_p(\mathcal{O})$ admit equations of the form $E_A : y^2 = x^3 + Ax^2 + x$, which allows to represent them by their Montgomery coefficient $A \in \mathbb{F}_p$. An intrinsic property of the CSIDH group action which is not covered by Definition 3, is the following. For any curve $E_A = [\mathfrak{a}] \star E_0 \in \mathcal{E}\ell\ell_p(\mathcal{O})$, its quadratic twist is easily computed as $(E_A)^t = E_{-A}$ and satisfies the property $(E_A)^t = [\mathfrak{a}]^{-1} \star E_0$.

Definition 4 ((Restricted) Effective Group Action with Twists). *We say that a* (R)EGA $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ *is a (Restricted) Effective Group Action with Twists* ((R)EGAT) *if there exists an efficient algorithm that given* $x = g \star \tilde{x} \in \mathcal{X}$ *computes* $x^t = g^{-1} \star \tilde{x}$.

As noted in [13, §10], this property contrasts with the classical group-based setting. It has already been used for the design of new cryptographic primitives based on CSIDH such as the signature scheme CSIFiSh [11] and the OT protocol in [27]. Moreover, it is important to consider twists in the security analysis of schemes based on group actions. In Section 5 we use twists to construct an attack on the protocol GA-PAKE$_\ell$ showing that it cannot be securely instantiated with the CSIDH group action. On the other hand, we prove that GA-PAKE$_\ell$ is secure when instantiated with a group action without efficient twisting (Theorem 3).

3.2 Computational Assumptions

For cryptographic applications, we are interested in (restricted) effective group actions that are equipped with the following hardness properties:

- Given $(x, y) \in \mathcal{X}^2$, it is hard to find $g \in \mathcal{G}$ such that $y = g \star x$.
- Given $(x, y_0, y_1) \in \mathcal{X}^3$, it is hard to find $z = (g_0 \cdot g_1) \star x$, where $g_0, g_1 \in \mathcal{G}$ are such that $y_0 = g_0 \star x$ and $y_1 = g_1 \star x$.

In [5] such group actions are called cryptographic group actions, and in [14] they are called hard homogeneous spaces.

The two hardness assumptions are the natural generalizations of the discrete logarithm assumption and the Diffie-Hellman assumption in the traditional group based setting. In analogy to this setting, we introduce the notation

$$\mathsf{GA\text{-}CDH}_x(y_0, y_1) = g_0 \star y_1, \quad \text{where } g_0 \in \mathcal{G} \text{ such that } y_0 = g_0 \star x$$

and define the decision oracle

$$\text{GA-DDH}_x(y_0, y_1, z) = \begin{cases} 1 & \text{if } \mathsf{GA\text{-}CDH}_x(y_0, y_1) = z, \\ 0 & \text{otherwise.} \end{cases}$$

For both, $\mathsf{GA\text{-}CDH}$ and GA-DDH, we omit the index x if $x = \tilde{x}$, i.e. we set $\mathsf{GA\text{-}CDH}_{\tilde{x}}(y_0, y_1) = \mathsf{GA\text{-}CDH}(y_0, y_1)$ and $\text{GA-DDH}_{\tilde{x}}(y_0, y_1, z) = \text{GA-DDH}(y_0, y_1, z)$.

We now introduce three computational problems $\mathsf{GA\text{-}StCDH}$, $\mathsf{GA\text{-}GapCDH}$, $\mathsf{SqInv\text{-}GA\text{-}StCDH}$ (Definitions 5 to 7). The security of our PAKE protocols relies on the hardness of these problems.

The first two problems are variants of the standard Diffie-Hellman problem, where an adversary is either given access to some fixed-basis decision oracles (indicated by the prefix *strong*) or to a general decision oracle (indicated by the prefix *gap*). Note that these problems were already defined and used in previous work [35,17,25,26]. In contrast the problem from Definition 7 has not been studied in any previous work. Therefore, we provide evidence for its hardness in Remark 3.

Definition 5 (Group Action Strong Computational Diffie-Hellman Problem ($\mathsf{GA\text{-}StCDH}$)). *On input $(g \star \tilde{x}, h \star \tilde{x}) \in \mathcal{X}^2$, the $\mathsf{GA\text{-}StCDH}$ problem requires to compute the set element $(g \cdot h) \star \tilde{x}$. To an effective group action $\mathsf{XXX} \in \{\mathsf{EGA}, \mathsf{REGA}, \mathsf{EGAT}, \mathsf{REGAT}\}$, we associate the advantage function of an adversary $\mathcal{A}$ as*

$$\mathsf{Adv}^{\mathsf{GA\text{-}StCDH}}_{\mathsf{XXX}}(\mathcal{A}) := \Pr[\mathcal{A}^{\text{GA-DDH}(g\star\tilde{x},\cdot,\cdot)}(g \star \tilde{x}, h \star \tilde{x}) \Rightarrow (g \cdot h) \star \tilde{x}] \ ,$$

where $(g, h) \xleftarrow{\$} \mathcal{G}^2$ and $\mathcal{A}$ has access to decision oracle $\text{GA-DDH}(g \star \tilde{x}, \cdot, \cdot)$.

Definition 6 (Group Action Gap Computational Diffie-Hellman Problem ($\mathsf{GA\text{-}GapCDH}$)). *On input $(g \star \tilde{x}, h \star \tilde{x}) \in \mathcal{X}^2$, the $\mathsf{GA\text{-}GapCDH}$ problem requires to compute the set element $(g \cdot h) \star \tilde{x}$. To an effective group action $\mathsf{XXX} \in \{\mathsf{EGA}, \mathsf{REGA}, \mathsf{EGAT}, \mathsf{REGAT}\}$, we associate the advantage function of an adversary $\mathcal{A}$ as*

$$\mathsf{Adv}^{\mathsf{GA\text{-}GapCDH}}_{\mathsf{XXX}}(\mathcal{A}) := \Pr[\mathcal{A}^{\text{GA-DDH}_*}(g \star \tilde{x}, h \star \tilde{x}) \Rightarrow (g \cdot h) \star \tilde{x}] \ ,$$

where $(g, h) \xleftarrow{\$} \mathcal{G}^2$ and $\mathcal{A}$ has access to a general decision oracle GA-DDH_*.

Remark 2. A group action where the group action computational Diffie-Hellman problem (without any decision oracle) is hard, is the same as a weak unpredictable group action as defined by Alamati et al. [5]. Further details are given in Appendix A. Also note that the ability to compute the twist of a set element does not help in solving these problems. Hence, all results based on these problems remain true for $\mathsf{(R)EGAT}$.

Definition 7 (Square-Inverse $\mathsf{GA\text{-}StCDH}$ ($\mathsf{SqInv\text{-}GA\text{-}StCDH}$)). *On input $x = g\star\tilde{x}$, the $\mathsf{SqInv\text{-}GA\text{-}StCDH}$ problem requires to find a tuple $(y, y_0, y_1) \in \mathcal{X}^3$ such that $y_0 = g^2 \star y$ and $y_1 = g^{-1} \star y$. For a group action $\mathsf{XXX} \in \{\mathsf{EGA}, \mathsf{REGA}, \mathsf{EGAT}, \mathsf{REGAT}\}$, we define the advantage function of $\mathcal{A}$ as*

$$\mathsf{Adv}^{\mathsf{SqInv\text{-}GA\text{-}StCDH}}_{\mathsf{XXX}}(\mathcal{A}) := \Pr\left[\begin{array}{l} y_0 = \mathsf{GA\text{-}CDH}_{x^t}(x, y) \\ y_1 = \mathsf{GA\text{-}CDH}(x^t, y) \end{array} \middle| \begin{array}{c} g \xleftarrow{\$} \mathcal{G} \\ x = g \star \tilde{x} \\ (y, y_0, y_1) \leftarrow \mathcal{A}^{\mathrm{O}}(x) \end{array}\right] \ ,$$

where $\mathrm{O} = \{\text{GA-DDH}_{x^t}(x, \cdot, \cdot), \text{GA-DDH}(x, \cdot, \cdot)\}$.

Remark 3. Intuitively $\mathsf{SqInv\text{-}GA\text{-}StCDH}$ is hard if we assume that the adversary can only use the group and twist operation. To go into more detail, $\mathcal{A}$ can choose y only based on known elements, that is either based on $\tilde{x}$, its input x or x^t.

If $\mathcal{A}$ chooses $y = \alpha \star \tilde{x}$ for some $\alpha \in \mathcal{G}$, then it can easily compute $y_1 = \alpha \star x^t$, but not $y_0 = \alpha g^2 \star \tilde{x}$. If $\mathcal{A}$ chooses $y = \alpha \star x$, then computing $y_1 = \alpha \star \tilde{x}$ is trivial, but computing $y_0 = \alpha g^3 \star \tilde{x}$ is hard. If $\mathcal{A}$ chooses $y = \alpha \star x^t$, then computing $y_0 = \alpha \star x$ is trivial, but computing $y_1 = \alpha g^{-2} \star \tilde{x}$ is hard.

4 Password Authenticated Key Exchange

Password-authenticated key exchange (PAKE) allows two parties, typically referred to as the user and the server, to establish a shared session key with the help of a short secret, known as a password, which can be drawn from a small set of possible values. To prove security of a PAKE protocol, we use the indistinguishability-based security model by Bellare, Pointcheval and Rogaway [8] and its extension to multiple test queries by Abdalla, Fouque and Pointcheval [2]. In our proofs, we further adapt the game-based pseudocode used in [1].

The name spaces for users $\mathcal{U}$ and servers $\mathcal{S}$ are assumed to be disjoint. Each pair of user and server $(\mathsf{U}, \mathsf{S}) \in \mathcal{U} \times \mathcal{S}$ holds a shared password $\mathsf{pw}_{\mathsf{US}}$. A party P denotes either a user or server. Each party P has multiple instances π_{P}^i and each instance has its own state. We denote the session key space by $\mathcal{K}$. Passwords are bit strings of length ℓ and we define the password space as $\mathcal{PW} \subsetneq \{0,1\}^\ell$.

Instance State. The state of an instance π_{P}^i is a tuple $(\mathsf{e}, \mathsf{tr}, K, \mathrm{acc})$ where

- e stores the (secret) ephemeral values chosen by the party in that instance (in our case group elements).
- tr stores the trace of that instance, i.e., the user and server name involved in the protocol execution and the messages sent and received by that instance.
- K is the accepted session key.
- acc is a Boolean flag that indicates whether the instance has accepted the session key. As long as the instance did not receive the last message, acc $= \perp$.

To access individual components of the state, we write $\pi_{\mathsf{P}}^t.\{\mathsf{e}, \mathsf{tr}, K, \mathrm{acc}\}$.

Partnering. Partnering is defined via matching conversations. In particular, a user instance $\pi_{\mathsf{U}}^{t_0}$ and a server instance $\pi_{\mathsf{S}}^{t_1}$ are partnered iff

$$\pi_{\mathsf{U}}^{t_0}.\mathrm{acc} = \pi_{\mathsf{S}}^{t_1}.\mathrm{acc} = \mathbf{true} \quad \mathbf{and} \quad \pi_{\mathsf{U}}^{t_0}.\mathsf{tr} = \pi_{\mathsf{S}}^{t_1}.\mathsf{tr} \ .$$

Two user instances are never partnered, neither are two server instances. We define a partner predicate $\mathsf{Partner}(\pi_{\mathsf{P}_0}^{t_0}, \pi_{\mathsf{P}_1}^{t_1})$ which outputs 1 if the two instances $\pi_{\mathsf{P}_0}^{t_0}$ and $\pi_{\mathsf{P}_1}^{t_1}$ are partnered and 0 otherwise.

Security Experiment. The security experiment is played between a challenger and an adversary $\mathcal{A}$. The challenger draws a random challenge bit β and creates the public parameters. Then it outputs the public parameters to $\mathcal{A}$. Now $\mathcal{A}$ has access to the following oracles:

- $\textsc{Execute}(\mathsf{U}, t_0, \mathsf{S}, t_1)$: one complete protocol execution between user instance $\pi_{\mathsf{U}}^{t_0}$ and server instance $\pi_{\mathsf{S}}^{t_1}$. This query models security against passive adversaries.
- $\textsc{SendInit}$, $\textsc{SendResp}$, $\textsc{SendTermInit}$, $\textsc{SendTermResp}$: send oracles to model security against active adversaries. $\textsc{SendTermResp}$ is only available for three-message protocols.
- $\textsc{Corrupt}(\mathsf{U}, \mathsf{S})$: outputs the shared password $\mathsf{pw}_{\mathsf{US}}$ of U and S.
- $\textsc{Reveal}(\mathsf{P}, t)$: outputs the session key of instance π_{P}^t.
- $\textsc{Test}(\mathsf{P}, t)$: challenge query. Depending on the challenge bit β, the experiment outputs either the session key of instance π_{P}^t or a uniformly random key. By $\pi_{\mathsf{P}}^t.\mathrm{test} = \mathbf{true}$, we mark an instance as tested.

We denote the experiment by $\mathsf{Exp}_{\mathsf{PAKE}}$. The pseudocode is given in G_0 in Figure 5, instantiated with our first PAKE protocol.

Freshness. During the game, we register if a query is allowed to prevent trivial wins. Therefore, we define a freshness predicate $\mathsf{Fresh}(\mathsf{P}, i)$. An instance π_{P}^t is fresh iff

1. π_{P}^t accepted.
2. π_{P}^t was not queried to Test or Reveal before.
3. At least one of the following conditions holds:
   3.1 π_{P}^t accepted during a query to Execute.
   3.2 There exists more than one partner instance.
   3.3 A unique fresh partner instance exists.
   3.4 No partner exists and Corrupt was not queried.

Definition 8 (Security of PAKE). *We define the security experiment, partnering and freshness conditions as above. The advantage of an adversary $\mathcal{A}$ against a password authenticated key exchange protocol* PAKE *in* $\mathsf{Exp}_{\mathsf{PAKE}}$ *is defined as*

$$\mathsf{Adv}_{\mathsf{PAKE}}(\mathcal{A}) := \left| \Pr[\mathsf{Exp}_{\mathsf{PAKE}} \Rightarrow 1] - \frac{1}{2} \right| .$$

A PAKE is considered secure if the best the adversary can do is to perform an online dictionary attack. More concretely, this means that the advantage of the adversary should be negligibly close to $q_s/|\mathcal{PW}|$ when passwords are drawn uniformly and independently from $\mathcal{PW}$, where q_s is the number of send queries made by the adversary.

Note that this definition captures weak forward secrecy. We will give an extended security definition capturing also perfect forward secrecy in Appendix F, as well as proofs for our protocols.

5 First Attempt: Protocol $\mathsf{GA\text{-}PAKE}_\ell$

The $\mathsf{GA\text{-}PAKE}_\ell$ protocol was already introduced in the introduction (Section 1). We refer to Figure 1 for a description of the protocol. In contrast to the two PAKE protocols from Sections 6 and 7, $\mathsf{GA\text{-}PAKE}_\ell$ is not secure for EGATs, i.e., if it is possible to compute twists of set elements efficiently. In particular it should not be instantiated with the CSIDH-group action. However, it is instructive to examine its security and it serves as a good motivation for the design of the two secure PAKE protocols $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ and $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$.

In this section we present an offline dictionary attack against $\mathsf{GA\text{-}PAKE}_\ell$ for $(\mathsf{R})\mathsf{EGAT}$. However, if twisting is hard, then we can prove security of $\mathsf{GA\text{-}PAKE}_\ell$ based on a hardness assumption that is similar to the simultaneous Diffie-Hellman problem which was introduced to prove the security of TBPEKE and CPace [30,3]. Our proof for $\mathsf{GA\text{-}PAKE}_\ell$ is given in Appendix C.

Proposition 1. *For* EGAT*s, the protocol* $\mathsf{GA\text{-}PAKE}_\ell$ *is vulnerable to offline dictionary attacks.*

Proof. We construct an adversary $\mathcal{A}$ that takes the role of the server. The attack is summarized in Figure 3. After receiving x^{U}, the adversary computes

$$x_i^{\mathsf{S}} = \tilde{s}_i \star (x_i^{\mathsf{U}})^t = \tilde{s}_i \star (u_i \star x_{b_i})^t = (\tilde{s}_i \cdot u_i^{-1}) \star x_{b_i}^t = (\tilde{s}_i \cdot u_i^{-1} \cdot g_{b_i}^{-1}) \star \tilde{x}$$

for each $i \in [\ell]$ and sends $x_1^{\mathsf{S}}, \ldots, x_\ell^{\mathsf{S}}$ to the user. Then the user computes $z_i = u_i \star x_i^{\mathsf{S}} = (\tilde{s}_i \cdot g_{b_i}^{-1}) \star \tilde{x} = \tilde{s}_i \star x_{b_i}^t$. For each $i \in [\ell]$, the adversary $\mathcal{A}$ can now compute z_i for both possibilities $b_i = 0$ and $b_i = 1$. This allows him to compute K for all possible passwords $\mathsf{pw} \in \mathcal{PW} \subsetneq \{0,1\}^\ell$ (being offline). □

This offline attack can easily be used to win the security experiment with high probability. $\mathcal{A}$ only needs to issue two send queries. It chooses any user U, initiates a session and computes its message $x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}$ as described in Figure 3. It reveals the corresponding session key and starts its offline attack by brute forcing all $\mathsf{pw} \in \mathcal{PW}$ until it finds a match for a candidate pw^*. Now $\mathcal{A}$ issues its second send query. This time it computes the message following the protocol using pw^* and derives a key K^*. It issues a test query and gets K_β. If $K^* = K_\beta$, then it outputs 0, otherwise it outputs 1. In case there is more than one password candidate, i.e., two inputs to H lead to the same K^*, then $\mathcal{A}$ can issue another send and reveal query to rule out false positives. In the end, it can still happen that $\beta = 1$ and $K^* = K$, but this event only occurs with probability $1/|\mathcal{K}|$.

Corollary 1. *For any adversary $\mathcal{A}$ against* $\mathsf{GA\text{-}PAKE}_\ell$ *instantiated with an* EGAT*, we have* $\Pr[\mathsf{Exp}_{\mathsf{GA\text{-}PAKE}_\ell} \Rightarrow 1] = 1 - \frac{1}{|\mathcal{K}|}$.

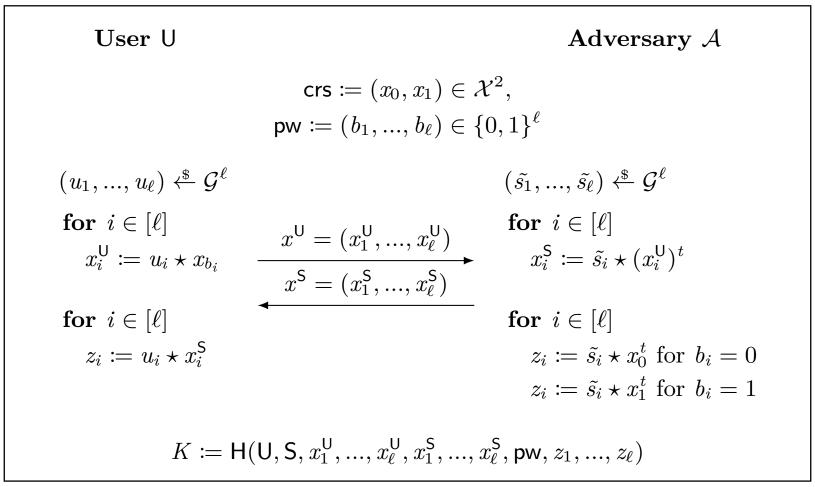

User U
Adversary $\mathcal{A}$
$\mathsf{crs} \coloneqq (x_0, x_1) \in \mathcal{X}^2$,
$\mathsf{pw} \coloneqq (b_1, ..., b_\ell) \in \{0,1\}^\ell$
$(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$(\tilde{s}_1, ..., \tilde{s}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
for $i \in [\ell]$
$x_i^{\mathsf{U}} \coloneqq u_i \star x_{b_i}$
$x^{\mathsf{U}} = (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}})$
for $i \in [\ell]$
$x_i^{\mathsf{S}} \coloneqq \tilde{s}_i \star (x_i^{\mathsf{U}})^t$
$x^{\mathsf{S}} = (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}})$
for $i \in [\ell]$
$z_i \coloneqq u_i \star x_i^{\mathsf{S}}$
for $i \in [\ell]$
$z_i \coloneqq \tilde{s}_i \star x_0^t$ for $b_i = 0$
$z_i \coloneqq \tilde{s}_i \star x_1^t$ for $b_i = 1$
$K \coloneqq \mathsf{H}(\mathsf{U}, \mathsf{S}, x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}, x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}, \mathsf{pw}, z_1, ..., z_\ell)$

Fig. 3. Attack against $\mathsf{GA\text{-}PAKE}_\ell$ using twists.

6 $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$: One-Round PAKE from Group Actions

In the previous section we showed that $\mathsf{GA\text{-}PAKE}_\ell$ is insecure when instantiated with an EGAT. Here, we present the modification $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$, which impedes the offline dictionary attack presented in that section. Broadly speaking, the idea is to double the message size of both parties in the first flow. In the second flow it is then necessary to compute certain "cross products" which is only possible if the previous message has been honestly generated. The letter X in $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ stands for cross product.

By means of these modifications, the protocol $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ is provably secure for EGATs. We show that its security can be reduced to the hardness of the computational problems $\mathsf{GA\text{-}StCDH}$ and $\mathsf{SqInv\text{-}GA\text{-}StCDH}$ (Theorem 1).

6.1 Description of the Protocol

The setup for $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ is the same as for $\mathsf{GA\text{-}PAKE}_\ell$. The $\mathsf{crs} = (x_0, x_1)$ comprises two elements of the set $\mathcal{X}$, and the shared password is a bit string $(b_1, \ldots, b_\ell)$ of length ℓ.

In the first flow of the protocol the user generates $2 \cdot \ell$ random group elements, $u_1, \ldots, u_\ell$ and $\hat{u}_1, \ldots, \hat{u}_\ell$. Using these elements it computes the set elements $x_i^{\mathsf{U}} = u_i \star x_{b_i}$ and $\hat{x}_i^{\mathsf{U}} = \hat{u}_i \star x_{b_i}$ for each $i \in [\ell]$ and sends these to the server.

Simultaneously, the server generates the random group elements $s_1, \ldots, s_\ell$ and $\hat{s}_1, \ldots, \hat{s}_\ell$, which it uses to compute the set elements $x_i^{\mathsf{S}} = s_i \star x_{b_i}$ and $\hat{x}_i^{\mathsf{S}} = \hat{s}_i \star x_{b_i}$ for each $i \in [\ell]$ and sends these to the user.

Upon receiving the set elements from the other party, both the server and the user compute the following three elements

$$z_{i,1} = u_i \star x_i^{\mathsf{S}} = s_i \star x_i^{\mathsf{U}}, \quad z_{i,2} = \hat{u}_i \star x_i^{\mathsf{S}} = s_i \star \hat{x}_i^{\mathsf{U}}, \quad z_{i,3} = u_i \star \hat{x}_i^{\mathsf{S}} = \hat{s}_i \star x_i^{\mathsf{U}},$$

for each $i \in [\ell]$. Finally, these elements are used to compute the session key K. The protocol is sketched in Figure 4.

6.2 Security of $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$

We now prove the security of $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ for EGATs.

Theorem 1 (Security of $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$). *For any adversary $\mathcal{A}$ against $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ that issues at most q_e execute queries and q_s send queries and where H is modeled as a random oracle, there exist an adversary $\mathcal{B}_1$ against $\mathsf{GA\text{-}StCDH}$ and an adversary $\mathcal{B}_2$ against $\mathsf{SqInv\text{-}GA\text{-}StCDH}$ such that*

$$\mathsf{Adv}_{\mathsf{X\text{-}GA\text{-}PAKE}_\ell}(\mathcal{A}) \le \mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{GA\text{-}StCDH}}(\mathcal{B}_1) + \mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{SqInv\text{-}GA\text{-}StCDH}}(\mathcal{B}_2) + \frac{q_s}{|\mathcal{PW}|} + \frac{(q_s + q_e)^2}{|\mathcal{G}|^{2\ell}} \ .$$

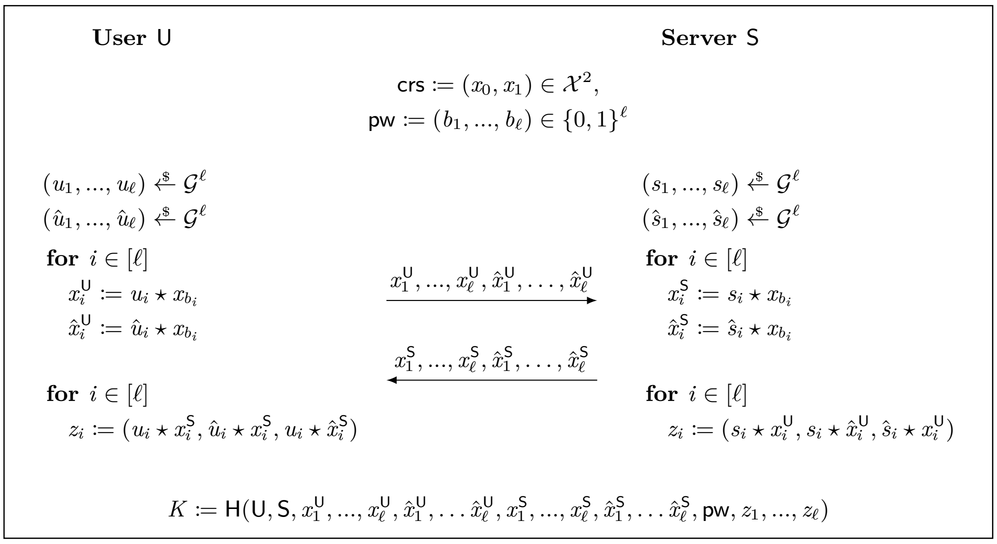

User U
Server S
$\mathsf{crs} := (x_0, x_1) \in \mathcal{X}^2$,
$\mathsf{pw} := (b_1, ..., b_\ell) \in \{0,1\}^\ell$
$(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$(\hat{u}_1, ..., \hat{u}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
for $i \in [\ell]$
$x_i^{\mathsf{U}} := u_i \star x_{b_i}$
$\hat{x}_i^{\mathsf{U}} := \hat{u}_i \star x_{b_i}$
$(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$(\hat{s}_1, ..., \hat{s}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
for $i \in [\ell]$
$x_i^{\mathsf{S}} := s_i \star x_{b_i}$
$\hat{x}_i^{\mathsf{S}} := \hat{s}_i \star x_{b_i}$
$\xrightarrow{x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}, \hat{x}_1^{\mathsf{U}}, \ldots, \hat{x}_\ell^{\mathsf{U}}}$
$\xleftarrow{x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}, \hat{x}_1^{\mathsf{S}}, \ldots, \hat{x}_\ell^{\mathsf{S}}}$
for $i \in [\ell]$
$z_i := (u_i \star x_i^{\mathsf{S}}, \hat{u}_i \star x_i^{\mathsf{S}}, u_i \star \hat{x}_i^{\mathsf{S}})$
for $i \in [\ell]$
$z_i := (s_i \star x_i^{\mathsf{U}}, s_i \star \hat{x}_i^{\mathsf{U}}, \hat{s}_i \star x_i^{\mathsf{U}})$
$K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}, \hat{x}_1^{\mathsf{U}}, \ldots \hat{x}_\ell^{\mathsf{U}}, x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}, \hat{x}_1^{\mathsf{S}}, \ldots \hat{x}_\ell^{\mathsf{S}}, \mathsf{pw}, z_1, ..., z_\ell)$

Fig. 4. PAKE protocol $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ from group actions.

Before proving Theorem 1, we will introduce a new computational assumption which is tailored to the protocol.

Definition 9 (Double Simultaneous GA-StCDH (DSim-GA-StCDH)). *On input* $(x_0, x_1, w_0, w_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x}, h_0 \star \tilde{x}, h_1 \star \tilde{x}) \in \mathcal{X}^4$, *the* DSim-GA-StCDH *problem requires to find a tuple* $(y, y_0, y_1, y_2, y_3) \in \mathcal{X}^5$ *such that*

$$(y_0, y_1, y_2, y_3) = (g_0^{-1} \cdot h_0 \star y,\ g_0^{-1} \cdot h_1 \star y,\ g_1^{-1} \cdot h_0 \star y,\ g_1^{-1} \cdot h_1 \star y).$$

For a group action $\mathsf{XXX} \in \{\mathsf{EGA}, \mathsf{REGA}, \mathsf{EGAT}, \mathsf{REGAT}\}$*, we define the advantage function of an adversary* $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathsf{DSim\text{-}GA\text{-}StCDH}}_{\mathsf{XXX}}(\mathcal{A}) := \Pr\left[\begin{array}{l|c} y_0 = \mathsf{GA\text{-}CDH}_{x_0}(w_0, y) & (g_0, g_1, h_0, h_1) \xleftarrow{\$} \mathcal{G}^4 \\ y_1 = \mathsf{GA\text{-}CDH}_{x_0}(w_1, y) & (x_0, x_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x}) \\ y_2 = \mathsf{GA\text{-}CDH}_{x_1}(w_0, y) & (w_0, w_1) = (h_0 \star \tilde{x}, h_1 \star \tilde{x}) \\ y_3 = \mathsf{GA\text{-}CDH}_{x_1}(w_1, y) & (y, y_0, y_1, y_2, y_3) \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1, w_0, w_1) \end{array}\right],$$

where $\mathrm{O} = \{\mathrm{GA\text{-}DDH}_{x_j}(w_i, \cdot, \cdot)\}_{i,j \in \{0,1\}}$.

Remark 4. Note that DSim-GA-StCDH may be viewed as the doubled version of the Sim-GA-StCDH problem defined in the appendix (cf. Definition 12). The latter is an assumption underlying the security of $\mathsf{GA\text{-}PAKE}_\ell$ and (in the notation of the above problem) it only requires to find the tuple (y, y_0, y_2). For a group action with twists, this admits the trivial solution $(y, y_0, y_2) = (w_0^t, x_0^t, x_1^t)$. Such a trivial solution is inhibited by requiring to find y_1 and y_3 as well.

Lemma 1. *In the* EGAT *setting, the square-inverse* GA-StCDH (SqInv-GA-StCDH) *implies the double simultaneous* GA-StCDH (DSim-GA-StCDH)*. In particular,*

$$\mathsf{Adv}^{\mathsf{DSim\text{-}GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{SqInv\text{-}GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{B})\ .$$

Proof. Given a challenge $x = g \star \tilde{x} \in \mathcal{X}$ and oracles $\mathrm{GA\text{-}DDH}(x, \cdot, \cdot)$, $\mathrm{GA\text{-}DDH}_{x^t}(x, \cdot, \cdot)$ for the SqInv-GA-StCDH problem, we choose three group elements $\alpha, \beta, \gamma \xleftarrow{\$} \mathcal{G}$ and call the adversary for the DSim-GA-StCDH problem on input

$$(x_0, x_1, w_0, w_1) = (x^t, \alpha \star x, \beta \star x, \gamma \star \tilde{x}).$$

The corresponding decision oracles can be simulated using the oracles provided by $\mathsf{SqInv\text{-}GA\text{-}StCDH}$. More precisely, for any $z_1, z_2 \in \mathcal{X}$:

$$\begin{aligned}
\text{GA-DDH}_{x_0}(w_0, z_1, z_2) &= \text{GA-DDH}_{x_t}(x, z_1, \beta^{-1} \star z_2), \\
\text{GA-DDH}_{x_0}(w_1, z_1, z_2) &= \text{GA-DDH}(x, z_1, \gamma^{-1} \star z_2), \\
\text{GA-DDH}_{x_1}(w_0, z_1, z_2) &= \text{GA-DDH}(\tilde{x}, z_1, (\alpha \cdot \beta^{-1}) \star z_2), \\
\text{GA-DDH}_{x_0}(w_1, z_1, z_2) &= \text{GA-DDH}(x, z_1^t, (\alpha^{-1} \cdot \gamma) \star z_2^t).
\end{aligned}$$

For the third oracle note that $\text{GA-DDH}(\tilde{x}, z_1, (\alpha \cdot \beta^{-1}) \star z_2) = 1$ precisely when $z_1 = (\alpha \cdot \beta^{-1}) \star z_2$. For the forth oracle, note that $\text{GA-DDH}_{x_0}(w_1, z_1, z_2) = 1$ iff $z_2 = (\alpha^{-1} \cdot \gamma \cdot g^{-1}) \star z_1$. This implies

$$z_2^t = (\alpha \cdot \gamma^{-1} \cdot g) \star z_1^t = (\alpha \cdot \gamma^{-1}) \cdot \mathsf{GA\text{-}CDH}(x, z_1^t).$$

If the adversary is successful, it returns a tuple (y, y_0, y_1, y_2, y_3), where

$$y_0 = g(\beta g) \star y, \;\; y_1 = g \cdot \gamma \star y, \;\; y_2 = \alpha^{-1} \cdot \beta \star y, \;\; y_3 = (\alpha g)^{-1} \gamma \star y.$$

Consequently, the tuple $(y, y_0', y_1') = (y, \beta^{-1} \star y_0, \alpha \star y_3)$ solves the $\mathsf{SqInv\text{-}GA\text{-}StCDH}$ problem. □

In the following, we give the full proof of Theorem 1.

Proof (of Theorem 1). Let $\mathcal{A}$ be an adversary against $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$. Consider the games in Figures 5, 7, 8.

Game G_0. This is the original game, hence

$$\mathsf{Adv}_{\mathsf{X\text{-}GA\text{-}PAKE}_\ell}(\mathcal{A}) \leq |\Pr[\mathsf{G}_0 \Rightarrow 1] - 1/2| \;\; .$$

Game G_1. In game G_1, we raise flag $\mathbf{bad}_{\text{coll}}$ whenever a server instance computes the same trace as any other accepted instance (line 69) or a user instance computes the same trace as any other accepted user instance (line 84). In this case, SendResp or SendTermInit return $\perp$. We do the same if a trace that is computed in an Execute query collides with one of a previously accepted instance (line 28). Due to the difference lemma,

$$|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_0 \Rightarrow 1]| \leq \Pr[\mathbf{bad}_{\text{coll}}] \;\; .$$

Note that when $\mathbf{bad}_{\text{coll}}$ is not raised, each instance is unique and has at most one partner. In order to bound $\mathbf{bad}_{\text{coll}}$, recall that the trace of an oracle π_{P}^t consists of $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}} = (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}), \hat{x}^{\mathsf{U}} = (\hat{x}_1^{\mathsf{U}}, ...\hat{x}_\ell^{\mathsf{U}}), x^{\mathsf{S}} = (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}), \hat{x}^{\mathsf{S}} = (\hat{x}_1^{\mathsf{S}}, ..., \hat{x}_\ell^{\mathsf{S}}))$, where at least one of the message pairs $(x^{\mathsf{U}}, \hat{x}^{\mathsf{U}})$ or $(x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$ was chosen by the game. Thus, $\mathbf{bad}_{\text{coll}}$ can only happen if all those $2 \cdot \ell$ set elements collide with all $2 \cdot \ell$ set elements of another instance. The probability that this happens for two (fixed) sessions is $|\mathcal{G}|^{-2\ell}$, hence the union bound over q_e and q_s sessions yields

$$|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_0 \Rightarrow 1]| \leq \Pr[\mathbf{bad}_{\text{coll}}] \leq \binom{q_e + q_s}{2} \cdot \frac{1}{|\mathcal{G}|^{2\ell}} \leq \frac{(q_e + q_s)^2}{|\mathcal{G}|^{2\ell}} \;\; .$$

Game G_2. In game G_2, we make the freshness explicit. To each oracle π_{P}^t, we assign an additional variable $\pi_{\mathsf{P}}^t.\mathsf{fr}$ which is updated during the game. In particular, all instances used in execute queries are marked as fresh (line 34).

An instance is fresh if the password was not corrupted yet (lines 72, 89). Otherwise, it is not fresh (lines 74, 91). For user instances we also check if there exists a fresh partner (line 87). If $\mathcal{A}$ issues a Corrupt query later, the freshness variable will also be updated (line 103). When the session key of an instance is revealed, this instance and its potential partner instance are marked as not fresh (line 41). On a query to test, the game then only checks the freshness variable (line 44). These are only a conceptual changes, hence

$$\Pr[\mathsf{G}_2 \Rightarrow 1] = \Pr[\mathsf{G}_1 \Rightarrow 1] \;\; .$$

```
GAMES $\mathsf{G}_0$-$\mathsf{G}_4$
$(g_0, g_1) \xleftarrow{\$} \mathcal{G}^2$
$(x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x})$
$(\mathcal{C}, T) := (\varnothing, \varnothing)$
$\mathbf{bad}_{\text{coll}} := \mathbf{false}$
$\beta \xleftarrow{\$} \{0,1\}$
for $(\mathsf{U},\mathsf{S}) \in \mathcal{U} \times \mathcal{S}$
    $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
$\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
return $[\![\beta = \beta']\!]$

EXECUTE$(\mathsf{U}, t_0, \mathsf{S}, t_1)$
if $\pi_{\mathsf{U}}^{t_0} \neq \perp$ or $\pi_{\mathsf{S}}^{t_1} \neq \perp$
    return $\perp$
$(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$    // $\mathsf{G}_0$-$\mathsf{G}_3$
$u := (u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$\hat{u} := (\hat{u}_1, ..., \hat{u}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$s := (s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$\hat{s} := (\hat{s}_1, ..., \hat{s}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star x_{b_1}, ..., u_\ell \star x_{b_\ell})$    // $\mathsf{G}_0$-$\mathsf{G}_3$
$\hat{x}^{\mathsf{U}} := (\hat{x}_1^{\mathsf{U}}, ..., \hat{x}_\ell^{\mathsf{U}}) := (\hat{u}_1 \star x_{b_1}, ..., \hat{u}_\ell \star x_{b_\ell})$    // $\mathsf{G}_0$-$\mathsf{G}_3$
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star x_{b_1}, ..., s_\ell \star x_{b_\ell})$    // $\mathsf{G}_0$-$\mathsf{G}_3$
$\hat{x}^{\mathsf{S}} := (\hat{x}_1^{\mathsf{S}}, ..., \hat{x}_\ell^{\mathsf{S}}) := (\hat{s}_1 \star x_{b_1}, ..., \hat{s}_\ell \star x_{b_\ell})$    // $\mathsf{G}_0$-$\mathsf{G}_3$
for $i \in [\ell]$ :    // $\mathsf{G}_0$-$\mathsf{G}_3$
    $z_i := (z_{i,1}, z_{i,2}, z_{i,3}) := (u_i \star x_i^{\mathsf{S}}, \hat{u}_i \star x_i^{\mathsf{S}}, u_i \star \hat{x}_i^{\mathsf{S}})$    // $\mathsf{G}_0$-$\mathsf{G}_3$
$z := (z_1, \ldots, z_\ell)$    // $\mathsf{G}_0$-$\mathsf{G}_3$
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star \tilde{x}, ..., u_\ell \star \tilde{x})$    // $\mathsf{G}_4$
$\hat{x}^{\mathsf{U}} := (\hat{x}_1^{\mathsf{U}}, ..., \hat{x}_\ell^{\mathsf{U}}) := (\hat{u}_1 \star \tilde{x}, ..., \hat{u}_\ell \star \tilde{x})$    // $\mathsf{G}_4$
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star \tilde{x}, ..., s_\ell \star \tilde{x})$    // $\mathsf{G}_4$
$\hat{x}^{\mathsf{S}} := (\hat{x}_1^{\mathsf{S}}, ..., \hat{x}_\ell^{\mathsf{S}}) := (\hat{s}_1 \star \tilde{x}, ..., \hat{s}_\ell \star \tilde{x})$    // $\mathsf{G}_4$
if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$    // $\mathsf{G}_1$-$\mathsf{G}_4$
    $\mathbf{bad}_{\text{coll}} := \mathbf{true}$    // $\mathsf{G}_1$-$\mathsf{G}_4$
    return $\perp$    // $\mathsf{G}_1$-$\mathsf{G}_4$
$K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$    // $\mathsf{G}_0$-$\mathsf{G}_2$
$K \xleftarrow{\$} \mathcal{K}$    // $\mathsf{G}_3$-$\mathsf{G}_4$
$\pi_{\mathsf{U}}^{t_0} := ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
$\pi_{\mathsf{S}}^{t_1} := ((s, \hat{s}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
$(\pi_{\mathsf{U}}^{t_0}.\mathsf{fr}, \pi_{\mathsf{S}}^{t_1}.\mathsf{fr}) := (\mathbf{true}, \mathbf{true})$    // $\mathsf{G}_2$-$\mathsf{G}_4$
return $(\mathsf{U}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \mathsf{S}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$

REVEAL$(\mathsf{P}, t)$
if $\pi_{\mathsf{P}}^t.\mathrm{acc} \neq \mathbf{true}$ or $\pi_{\mathsf{P}}^t.\mathrm{test} = \mathbf{true}$
    return $\perp$
if $\exists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$
and $\pi_{\mathsf{P}'}^{t'}.\mathrm{test} = \mathbf{true}$
    return $\perp$
$\forall (\mathsf{P}', t')$ s. t. $\pi_{\mathsf{P}'}^{t'}.\mathsf{tr} = \pi_{\mathsf{P}}^t.\mathsf{tr}$    // $\mathsf{G}_2$-$\mathsf{G}_4$
    $\pi_{\mathsf{P}'}^{t'}.\mathsf{fr} := \mathbf{false}$    // $\mathsf{G}_2$-$\mathsf{G}_4$
return $\pi_{\mathsf{P}}^t.K$

TEST$(\mathsf{P}, t)$
if $\mathsf{Fresh}(\pi_{\mathsf{P}}^t) = \mathbf{false}$ return $\perp$    // $\mathsf{G}_0$-$\mathsf{G}_1$
if $\pi_{\mathsf{P}}^t.\mathsf{fr} = \mathbf{false}$ return $\perp$    // $\mathsf{G}_2$-$\mathsf{G}_4$
$K_0^* := \mathrm{REVEAL}(\mathsf{P}, t)$
if $K_0^* = \perp$ return $\perp$
$K_1^* \xleftarrow{\$} \mathcal{K}$
$\pi_{\mathsf{P}}^t.\mathrm{test} := \mathbf{true}$
return $K_\beta^*$

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z)$
if $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z] = K \neq \perp$
    return $K$
$T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, Z] \xleftarrow{\$} \mathcal{K}$
return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z]$

SENDINIT$(\mathsf{U}, t, \mathsf{S})$
if $\pi_{\mathsf{U}}^t \neq \perp$ return $\perp$
$(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
$u := (u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$\hat{u} := (\hat{u}_1, ..., \hat{u}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star x_{b_1}, ..., u_\ell \star x_{b_\ell})$
$\hat{x}^{\mathsf{U}} := (\hat{x}_1^{\mathsf{U}}, ..., \hat{x}_\ell^{\mathsf{U}}) := (\hat{u}_1 \star x_{b_1}, ..., \hat{u}_\ell \star x_{b_\ell})$
$\pi_{\mathsf{U}}^t := ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \perp, \perp), \perp, \perp)$
$\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{false}$    // $\mathsf{G}_2$-$\mathsf{G}_4$
return $(\mathsf{U}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}})$

SENDRESP$(\mathsf{S}, t, \mathsf{U}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}})$
if $\pi_{\mathsf{S}}^t \neq \perp$ return $\perp$
$(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
$(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star x_{b_1}, ..., s_\ell \star x_{b_\ell})$
$\hat{x}^{\mathsf{S}} := (\hat{x}_1^{\mathsf{S}}, ..., \hat{x}_\ell^{\mathsf{S}}) := (\hat{s}_1 \star x_{b_1}, ..., \hat{s}_\ell \star x_{b_\ell})$
if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$    // $\mathsf{G}_1$-$\mathsf{G}_4$
    $\mathbf{bad}_{\text{coll}} := \mathbf{true}$    // $\mathsf{G}_1$-$\mathsf{G}_4$
    return $\perp$    // $\mathsf{G}_1$-$\mathsf{G}_4$
if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$    // $\mathsf{G}_2$-$\mathsf{G}_4$
    $\pi_{\mathsf{S}}^t.\mathsf{fr} := \mathbf{true}$    // $\mathsf{G}_2$-$\mathsf{G}_4$
else    // $\mathsf{G}_2$-$\mathsf{G}_4$
    $\pi_{\mathsf{S}}^t.\mathsf{fr} := \mathbf{false}$    // $\mathsf{G}_2$-$\mathsf{G}_4$
for $i \in [\ell]$ :
    $z_i := (z_{i,1}, z_{i,2}, z_{i,3}) := (s_i \star x_i^{\mathsf{U}}, s \star \hat{x}_i^{\mathsf{U}}, \hat{s}_i \star x_i^{\mathsf{U}})$
$z := (z_1, ..., z_\ell)$
$K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$
$\pi_{\mathsf{S}}^t := ((s, \hat{s}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
return $(\mathsf{S}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$

SENDTERMINIT$(\mathsf{U}, t, \mathsf{S}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
if $\pi_{\mathsf{U}}^t \neq ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \perp, \perp), \perp, \perp)$
    return $\perp$
if $\exists \mathsf{P} \in \mathcal{U}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$    // $\mathsf{G}_1$-$\mathsf{G}_4$
    $\mathbf{bad}_{\text{coll}} := \mathbf{true}$    // $\mathsf{G}_1$-$\mathsf{G}_4$
    return $\perp$    // $\mathsf{G}_1$-$\mathsf{G}_4$
if $\exists t'$ s. t. $\pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
and $\pi_{\mathsf{S}}^{t'}.\mathsf{fr} = \mathbf{true}$    // $\mathsf{G}_2$-$\mathsf{G}_4$
    $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{true}$    // $\mathsf{G}_2$-$\mathsf{G}_4$
else if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$    // $\mathsf{G}_2$-$\mathsf{G}_4$
    $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{true}$    // $\mathsf{G}_2$-$\mathsf{G}_4$
else    // $\mathsf{G}_2$-$\mathsf{G}_4$
    $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{false}$    // $\mathsf{G}_2$-$\mathsf{G}_4$
for $i \in [\ell]$ :
    $z_i := (z_{i,1}, z_{i,2}, z_{i,3}) := (u_i \star x_i^{\mathsf{S}}, \hat{u}_i \star x_i^{\mathsf{S}}, u_i \star \hat{x}_i^{\mathsf{S}})$
$z := (z_1, ..., z_\ell)$
$K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$
$\pi_{\mathsf{U}}^t := ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}} x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
return true

CORRUPT$(\mathsf{U}, \mathsf{S})$
if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$ return $\perp$
for $\mathsf{P} \in \{\mathsf{U}, \mathsf{S}\}$
    if $\exists t$ s. t. $\pi_{\mathsf{P}}^t.\mathrm{test} = \mathbf{true}$
    and $\nexists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$
        return $\perp$
    $\forall \pi_{\mathsf{P}}^t$ : if $\nexists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$    // $\mathsf{G}_2$-$\mathsf{G}_4$
        $\pi_{\mathsf{P}}^t.\mathsf{fr} = \mathbf{false}$    // $\mathsf{G}_2$-$\mathsf{G}_4$
$\mathcal{C} := \mathcal{C} \cup \{(\mathsf{U}, \mathsf{S})\}$
return $\mathsf{pw}_{\mathsf{US}}$
```

Fig. 5. Games G_0-G_4 for the proof of Theorem 1. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$EXECUTE, SENDINIT, SENDRESP, SENDTERMINIT, REVEAL, CORRUPT, TEST, $\mathsf{H}\}$.

```
$\mathcal{B}_1^{\text{GA-DDH}(x,\cdot,\cdot)}(x, y)$
00 $(g_0, g_1) \xleftarrow{\$} \mathcal{G}^2$
01 $(x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x})$
02 $(\mathcal{C}, T, T_e) := (\varnothing, \varnothing, \varnothing)$
03 $\mathbf{bad}_{\text{coll}} := \mathbf{false}$
04 $\beta \xleftarrow{\$} \{0, 1\}$
05 for $(\mathsf{U}, \mathsf{S}) \in \mathcal{U} \times \mathcal{S}$
06     $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
07 $\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
08 Stop.

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z)$
09 if $\exists (u, \hat{u}, s, \hat{s})$
     s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, u, \hat{u}, s, \hat{s}) \in T_e$
10     $(b_1, ..., b_\ell) := \mathsf{pw}$
11     for $i \in [\ell]$
12         $(z_{i,1}, z_{i,2}, z_{i,3}) := z_i$
13         if $\text{GA-DDH}(x, x_i^{\mathsf{S}}, (u_i^{-1} \cdot g_{b_i}) \star z_{i,1}) = 1$
14             Stop with $(u_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_{i,1}$
15         if $\text{GA-DDH}(x, x_i^{\mathsf{S}}, (\hat{u}_i^{-1} \cdot g_{b_i}) \star z_{i,2}) = 1$
16             Stop with $(\hat{u}_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_{i,2}$
17         if $\text{GA-DDH}(x, x_i^{\mathsf{S}}, (u_i^{-1} \cdot g_{b_i}) \star z_{i,3}) = 1$
18             Stop with $(u_i^{-1} \cdot \hat{s}_i^{-1} \cdot g_{b_i}) \star z_{i,3}$
19 if $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z] = K \neq \bot$
20     return $K$
21 $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z] \xleftarrow{\$} \mathcal{K}$
22 return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z]$

$\textsc{Execute}(\mathsf{U}, t_0, \mathsf{S}, t_1)$
23 if $\pi_{\mathsf{U}}^{t_0} \neq \bot$ or $\pi_{\mathsf{S}}^{t_1} \neq \bot$
24     return $\bot$
25 $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
26 $u := (u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
27 $\hat{u} := (\hat{u}_1, ..., \hat{u}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
28 $s := (s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
29 $\hat{s} := (\hat{s}_1, ..., \hat{s}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
30 $x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star x, ..., u_\ell \star x)$
31 $\hat{x}^{\mathsf{U}} := (\hat{x}_1^{\mathsf{U}}, ..., \hat{x}_\ell^{\mathsf{U}}) := (\hat{u}_1 \star x, ..., \hat{u}_\ell \star x)$
32 $x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star y, ..., s_\ell \star y)$
33 $\hat{x}^{\mathsf{S}} := (\hat{x}_1^{\mathsf{S}}, ..., \hat{x}_\ell^{\mathsf{S}}) := (\hat{s}_1 \star y, ..., \hat{s}_\ell \star y)$
34 if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
35     $\mathbf{bad}_{\text{coll}} := \mathbf{true}$
36     return $\bot$
37 $\forall z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
38     for $i \in [\ell]$
39         $(z_{i,1}, z_{i,2}, z_{i,3}) := z_i$
40         if $\text{GA-DDH}(x, x_i^{\mathsf{S}}, (u_i^{-1} \cdot g_{b_i}) \star z_{i,1}) = 1$
41             Stop with $(u_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_{i,1}$
42         if $\text{GA-DDH}(x, x_i^{\mathsf{S}}, (\hat{u}_i^{-1} \cdot g_{b_i}) \star z_{i,2}) = 1$
43             Stop with $(\hat{u}_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_{i,2}$
44         if $\text{GA-DDH}(x, x_i^{\mathsf{S}}, (u_i^{-1} \cdot g_{b_i}) \star z_{i,3}) = 1$
45             Stop with $(u_i^{-1} \cdot \hat{s}_i^{-1} \cdot g_{b_i}) \star z_{i,3}$
46 $T_e := T_e \cup \{\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, u, \hat{u}, s, \hat{s}\}$
47 $K \xleftarrow{\$} \mathcal{K}$
48 $\pi_{\mathsf{U}}^{t_0} := ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
49 $\pi_{\mathsf{S}}^{t_1} := ((s, \hat{s}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
50 $(\pi_{\mathsf{U}}^{t_0}.\mathsf{fr}, \pi_{\mathsf{S}}^{t_1}.\mathsf{fr}) := (\mathbf{true}, \mathbf{true})$
51 return $(\mathsf{U}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \mathsf{S}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
```

Fig. 6. Adversary $\mathcal{B}_1$ against GA-StCDH for the proof of Theorem 1. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$Execute, SendInit, SendResp, SendTermInit, Reveal, Corrupt, Test, H$\}$. Oracles SendInit, SendResp, SendTermInit, Reveal, Corrupt and Test are defined as in G_2. Lines written in blue show how $\mathcal{B}_1$ simulates the game.

Game G_3. In game G_3, we choose random keys for instances queried to Execute. We construct adversary $\mathcal{B}_1$ against GA-StCDH in Figure 6 and show that

$$|\Pr[\mathsf{G}_3 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1]| \leq \mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{GA\text{-}StCDH}}(\mathcal{B}_1) \ .$$

Adversary $\mathcal{B}_1$ inputs a GA-StCDH challenge $(x, y) = (g \star \tilde{x}, h \star \tilde{x})$ and has access to a decision oracle $\text{GA-DDH}(x, \cdot, \cdot)$. First, it generates the crs elements (x_0, x_1) as in game G_3 and then runs adversary $\mathcal{A}$. Queries to Execute are simulated as follows: It chooses random group elements u_i, $\hat{u}_i$ and s_i, $\hat{s}_i$ for user and server instances and $i \in [\ell]$, but instead of using (x_0, x_1) to compute the set elements, $\mathcal{B}_1$ uses x for the user instance and y for the server instance, independent of the password bits b_i (lines 30 to 33). We can rewrite this as

$$x_i^{\mathsf{U}} = u_i \star x = (u_i \cdot g) \star \tilde{x} = (u_i \cdot g \cdot g_{b_i} \cdot g_{b_i}^{-1}) \star \tilde{x} = \underbrace{(u_i \cdot g \cdot g_{b_i}^{-1})}_{u_i'} \star x_{b_i} \ ,$$

where u_i' is the group element that the user actually needs in order to compute the session key. In the same way, $\hat{u}_i' = \hat{u}_i \cdot g \cdot g_{b_i}^{-1}$, $s_i' = s_i \cdot h \cdot g_{b_i}^{-1}$ and $\hat{s}_i' = \hat{s}_i \cdot h \cdot g_{b_i}^{-1}$. Note that $z_i = (z_{i,1}, z_{i,2}, z_{i,3})$ is implicitly set to

$$\begin{aligned} z_{i,1} &= (u_i' \cdot s_i') \star x_{b_i} = u_i \cdot g \cdot s_i \cdot h \cdot g_{b_i}^{-1} \star \tilde{x} \ , \\ z_{i,2} &= (\hat{u}_i' \cdot s_i') \star x_{b_i} = \hat{u}_i \cdot g \cdot s_i \cdot h \cdot g_{b_i}^{-1} \star \tilde{x} \ , \\ z_{i,3} &= (u_i' \cdot \hat{s}_i') \star x_{b_i} = u_i \cdot g \cdot \hat{s}_i \cdot h \cdot g_{b_i}^{-1} \star \tilde{x} \ . \end{aligned}$$

Before choosing a random session key, we check if there has been a query to the random oracle H that matches the session key (line 37-45). We iterate over the entries in T, where U, S, x^{U}, $\hat{x}^{\mathsf{U}}$, x^{S}, $\hat{x}^{\mathsf{S}}$ and $\mathsf{pw}_{\mathsf{US}}$

match, and check if one of the entries in z is correct. Note that we can use the following equivalences:

$$\begin{aligned}
\mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = z_{i,1} \quad &\Leftrightarrow \quad \mathsf{GA\text{-}CDH}(x, x_i^{\mathsf{S}}) = (u_i^{-1} \cdot g_{b_i}) \star z_{i,1}, \\
\mathsf{GA\text{-}CDH}_{x_{b_i}}(\hat{x}_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = z_{i,2} \quad &\Leftrightarrow \quad \mathsf{GA\text{-}CDH}(x, x_i^{\mathsf{S}}) = (\hat{u}_i^{-1} \cdot g_{b_i}) \star z_{i,2}, \\
\mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{U}}, \hat{x}_i^{\mathsf{S}}) = z_{i,3} \quad &\Leftrightarrow \quad \mathsf{GA\text{-}CDH}(x, \hat{x}_i^{\mathsf{S}}) = (u_i^{-1} \cdot g_{b_i}) \star z_{i,3},
\end{aligned}$$

which allows us to use the restricted decision oracle $\mathrm{GA\text{-}DDH}(x, \cdot, \cdot)$. If one of $z_{i,1}, z_{i,2}, z_{i,3}$ is correct, $\mathcal{B}_1$ aborts and outputs the solution $(g \cdot h) \star \tilde{x}$ which is respectively given by $(u_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_{i,1}$, $(\hat{u}_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_{i,2}$ or $(u_i^{-1} \cdot \hat{s}_i^{-1} \cdot g_{b_i}) \star z_{i,3}$.

Otherwise, we store the values $u_i, \hat{u}_i$ and $s_i, \hat{s}_i$ in list T_{e} together with the trace and the password (line 46) and choose a session key uniformly at random. We need list T_{e} to identify relevant queries to H. In particular, if the trace and password appear in a query, we retrieve the values $u_i, \hat{u}_i$ and $s_i, \hat{s}_i$ to check whether the provided z_i are correct. We do this in the same way as described above using the decision oracle (lines 09-18). If the oracle returns 1 for any $z_{i,j}$, $\mathcal{B}_1$ aborts and outputs the solution for $(g \cdot h) \star \tilde{x}$ which is respectively given by $(u_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_{i,1}$, $(\hat{u}_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_{i,2}$ or $(u_i^{-1} \cdot \hat{s}_i^{-1} \cdot g_{b_i}) \star z_{i,3}$.

Game G_4. In game G_4, we remove the password from execute queries. In particular, we do not compute $x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}$ to the basis x_{b_i}, but simply use $\tilde{x}$. Note that the values have the same distribution as in the previous game. Also, the group elements $u, \hat{u}, s$ and $\hat{s}$ are not used to to derive the key. Hence, this change is not observable by $\mathcal{A}$ and

$$\Pr[\mathsf{G}_4 \Rightarrow 1] = \Pr[\mathsf{G}_3 \Rightarrow 1] \ .$$

Game G_5. G_5 is given in Figure 7. In this game we want to replace the session keys by random for all fresh instances in oracles SendResp and SendTermInit (lines 62, 83). Therefore, we introduce an additional independent random oracle T_{s} which maps only the trace of an instance to a key (lines 63, 84). We keep partner instances consistent, i.e., in case the adversary queries SendTermInit for a user instance and there exists a fresh partner instance, then we retrieve the corresponding key from T_{s} and also assign it to this instance (line 78). For all instances that are not fresh, we simply compute the correct key using random oracle H (lines 66-69, 87-90). If a session is fresh and there is an inconsistency between T and T_{s}, we raise flag **bad**. This happens in the following cases:

– a server instance is about to compute the session key, the password was not corrupted, but there already exists an entry in T with the correct password and z (lines 60-61).
– a user instance is about to compute the session key, there exists no partner instance and the password was not corrupted, but there already exists an entry in T with the correct password and z (lines 81-82).
– the random oracle is queried on some trace that appears in T_{s} together with the correct password and z (lines 36-47). At this point, we also check if the password was corrupted in the meantime and if this is the case and the adversary issues the correct query, we simply output the key stored in T_{s} (line 46) as this instance cannot be tested. This case corresponds to perfect forward secrecy which we cover in Appendix F.2.

When **bad** is not raised, there is no difference between G_4 and G_5. Hence,

$$|\Pr[\mathsf{G}_5 \Rightarrow 1] - \Pr[\mathsf{G}_4 \Rightarrow 1]| \leq \Pr[\mathsf{G}_5 \Rightarrow \mathbf{bad}] \ .$$

Game G_6. G_6 is given in Figure 8. In this game we remove the password from send queries and generate passwords as late as possible, that is either when the adversary issues a corrupt query (line 21) or after it has stopped with output β' (line 07). In SendInit and SendResp we still choose group elements $u_i, \hat{u}_i, s_i$ and $\hat{s}_i$ uniformly at random, but now compute $x_i^{\mathsf{U}}, \hat{x}_i^{\mathsf{U}}, x_i^{\mathsf{S}}$ and $\hat{x}_i^{\mathsf{S}}$ using the origin element (lines 26, 27, 51 and 52). Thus, depending on which password is chosen afterwards, we implicitly set

$$x_i^{\mathsf{U}} = u_i \cdot \tilde{x} = (u_i \cdot g_0^{-1}) \star x_0 = (u_i \cdot g_1^{-1}) \star x_1$$

and analogously for $\hat{x}_i^{\mathsf{U}}, x_i^{\mathsf{S}}$ and $\hat{x}_i^{\mathsf{S}}$. For all instances that are not fresh, we have to compute the real session key using $z_i = (s_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{U}}, s_i \cdot g_{b_i}^{-1} \star \hat{x}_i^{\mathsf{U}}, \hat{s}_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{U}})$ (line 70) or $z_i = (u_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{S}}, \hat{u}_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{S}}, u_i \cdot g_{b_i}^{-1} \star \hat{x}_i^{\mathsf{S}})$ (line 97). Note that the password is already defined for these instances.

Recall that event **bad** in game G_5 is raised whenever there is an inconsistency in the random oracle queries and the keys of fresh instances. In this game, we split event **bad** into two different events:

```
GAME G_5
00 $(g_0, g_1) \xleftarrow{\$} \mathcal{G}^2$
01 $(x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x})$
02 $(\mathcal{C}, T, T_s) := (\varnothing, \varnothing, \varnothing)$
03 bad := false
04 $\beta \xleftarrow{\$} \{0,1\}$
05 for $(\mathsf{U}, \mathsf{S}) \in \mathcal{U} \times \mathcal{S}$
06    $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
07 $\beta' \leftarrow \mathcal{A}^{\mathsf{O}}(x_0, x_1)$
08 return $[\![\beta = \beta']\!]$

EXECUTE$(\mathsf{U}, t_0, \mathsf{S}, t_1)$
09 if $\pi_{\mathsf{U}}^{t_0} \neq \bot$ or $\pi_{\mathsf{S}}^{t_1} \neq \bot$: return $\bot$
10 $u := (u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
11 $\hat{u} := (\hat{u}_1, ..., \hat{u}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
12 $s := (s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
13 $\hat{s} := (\hat{s}_1, ..., \hat{s}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
14 $x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star \tilde{x}, ..., u_\ell \star \tilde{x})$
15 $\hat{x}^{\mathsf{U}} := (\hat{x}_1^{\mathsf{U}}, ..., \hat{x}_\ell^{\mathsf{U}}) := (\hat{u}_1 \star \tilde{x}, ..., \hat{u}_\ell \star \tilde{x})$
16 $x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star \tilde{x}, ..., s_\ell \star \tilde{x})$
17 $\hat{x}^{\mathsf{S}} := (\hat{x}_1^{\mathsf{S}}, ..., \hat{x}_\ell^{\mathsf{S}}) := (\hat{s}_1 \star \tilde{x}, ..., \hat{s}_\ell \star \tilde{x})$
18 if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
19    return $\bot$
20 $K \xleftarrow{\$} \mathcal{K}$
21 $\pi_{\mathsf{U}}^{t_0} := ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
22 $\pi_{\mathsf{S}}^{t_1} := ((s, \hat{s}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
23 $(\pi_{\mathsf{U}}^{t_0}.\mathsf{fr}, \pi_{\mathsf{S}}^{t_1}.\mathsf{fr}) := (\mathbf{true}, \mathbf{true})$
24 return $(\mathsf{U}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \mathsf{S}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$

SENDINIT$(\mathsf{U}, t, \mathsf{S})$
25 if $\pi_{\mathsf{U}}^t \neq \bot$ return $\bot$
26 $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
27 $u := (u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
28 $\hat{u} := (\hat{u}_1, ..., \hat{u}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
29 $x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star x_{b_1}, ..., u_\ell \star x_{b_\ell})$
30 $\hat{x}^{\mathsf{U}} := (\hat{x}_1^{\mathsf{U}}, ..., \hat{x}_\ell^{\mathsf{U}}) := (\hat{u}_1 \star x_{b_1}, ..., \hat{u}_\ell \star x_{b_\ell})$
31 $\pi_{\mathsf{U}}^t := ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \bot, \bot), \bot, \bot)$
32 $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ false
33 return $(\mathsf{U}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}})$

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z)$
34 if $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z] = K \neq \bot$
35    return $K$
36 if $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}) \in T_s$ and $\mathsf{pw} = \mathsf{pw}_{\mathsf{US}}$
37    if $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] = (\mathsf{U}, (u, \hat{u}), K)$
38       for $i \in [\ell]$
39          $z_i' := (u_i \star x_i^{\mathsf{S}}, \hat{u}_i \star x_i^{\mathsf{S}}, u_i \star \hat{x}_i^{\mathsf{S}})$
40       $z' := (z_1', ..., z_\ell')$
41    if $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] = (\mathsf{S}, (s, \hat{s}), K)$
42       for $i \in [\ell]$
43          $z_i' := (s_i \star x_i^{\mathsf{U}}, s_i \star \hat{x}_i^{\mathsf{U}}, \hat{s}_i \star x_i^{\mathsf{U}})$
44       $z' := (z_1', ..., z_\ell')$
45    if $z = z'$
46       if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$: return $K$
47       if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$: bad := true
48 $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z] \xleftarrow{\$} \mathcal{K}$
49 return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z]$

SENDRESP$(\mathsf{S}, t, \mathsf{U}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}})$
50 if $\pi_{\mathsf{S}}^t \neq \bot$ return $\bot$
51 $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
52 $s := (s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
53 $\hat{s} := (\hat{s}_1, ..., \hat{s}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
54 $x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star x_{b_1}, ..., s_\ell \star x_{b_\ell})$
55 $\hat{x}^{\mathsf{S}} := (\hat{x}_1^{\mathsf{S}}, ..., \hat{x}_\ell^{\mathsf{S}}) := (\hat{s}_1 \star x_{b_1}, ..., \hat{s}_\ell \star x_{b_\ell})$
56 if $\exists P \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
57    return $\bot$
58 if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
59    $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ true
60    if $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
      and $z_i := (s_i \star x_i^{\mathsf{U}}, s_i \star \hat{x}_i^{\mathsf{U}}, \hat{s}_i \star x_i^{\mathsf{U}})\ \forall i \in [\ell]$
61       bad := true
62    $K \xleftarrow{\$} \mathcal{K}$
63    $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] := (\mathsf{S}, (s, \hat{s}), K)$
64 else
65    $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ false
66    for $i \in [\ell]$
67       $z_i := (s_i \star x_i^{\mathsf{U}}, s_i \star \hat{x}_i^{\mathsf{U}}, \hat{s}_i \star x_i^{\mathsf{U}})$
68    $z := (z_1, ..., z_\ell)$
69    $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$
70 $\pi_{\mathsf{S}}^t := ((s, \hat{s}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
71 return $(\mathsf{S}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$

SENDTERMINIT$(\mathsf{U}, t, \mathsf{S}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
72 if $\pi_{\mathsf{U}}^t \neq ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \bot, \bot), \bot, \bot)$
73    return $\bot$
74 if $\exists \mathsf{P} \in \mathcal{U}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
75    return $\bot$
76 if $\exists t'$ s. t. $\pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
   and $\pi_{\mathsf{S}}^{t'}.\mathsf{fr} =$ true
77    $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ true
78    $(\mathsf{S}, (s, \hat{s}), K) := T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}]$
79 else if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
80    $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ true
81    if $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
      and $z_i := (u_i \star x_i^{\mathsf{S}}, \hat{u}_i \star x_i^{\mathsf{S}}, u_i \star \hat{x}_i^{\mathsf{S}})\ \forall i \in [\ell]$
82       bad := true
83    $K \xleftarrow{\$} \mathcal{K}$
84    $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] := (\mathsf{U}, (u, \hat{u}), K)$
85 else
86    $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ false
87    for $i \in [\ell]$
88       $z_i := (u_i \star x_i^{\mathsf{S}}, \hat{u}_i \star x_i^{\mathsf{S}}, u_i \star \hat{x}_i^{\mathsf{S}})$
89    $z := (z_1, ..., z_\ell)$
90    $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$
91 $\pi_{\mathsf{U}}^t := ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
92 return true
```

Fig. 7. Game G_5 for the proof of Theorem 1. $\mathcal{A}$ has access to oracles $\mathsf{O} := \{$EXECUTE, SENDINIT, SENDRESP, SENDTERMINIT, REVEAL, CORRUPT, TEST, $\mathsf{H}\}$. REVEAL, TEST and CORRUPT are defined as in Figure 5. Differences to G_4 are highlighted in blue.

– $\mathbf{bad}_{\mathsf{pw}}$ captures the event that there exists more than one valid entry in T for the same trace of a fresh instance, but different passwords.
– $\mathbf{bad}_{\text{guess}}$ happens only if $\mathbf{bad}_{\mathsf{pw}}$ does not happen and is raised if there exists a valid entry in T for the trace of a fresh instance and the correct password, where the password was not corrupted when

```
GAME G_6
$(g_0, g_1) \xleftarrow{\$} \mathcal{G}^2$
$(x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x})$
$(\mathcal{C}, T, T_s, T_{\text{bad}}) := (\varnothing, \varnothing, \varnothing, \varnothing)$
$(\mathbf{bad}_{\text{guess}}, \mathbf{bad}_{\mathsf{pw}}) := (\mathbf{false}, \mathbf{false})$
$\beta \xleftarrow{\$} \{0, 1\}$
$\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
$\mathbf{for}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{U} \times \mathcal{S} \setminus \mathcal{C}$
  $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
$\mathbf{if}\ \exists \mathsf{pw}, \mathsf{pw}', (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, z, z')$
 $\text{s.t. } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z) \in T_{\text{bad}}$
 $\mathbf{and}\ (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}', z') \in T_{\text{bad}}$
  $\mathbf{bad}_{\mathsf{pw}} := \mathbf{true}$
$\mathbf{else}$
  $\mathbf{if}\ \exists \mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, z$
  $\text{s.t. } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T_{\text{bad}}$
    $\mathbf{bad}_{\text{guess}} := \mathbf{true}$
$\mathbf{return}\ [\![\beta = \beta']\!]$

CORRUPT(U, S)
$\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{return}\ \perp$
$\mathbf{for}\ \mathsf{P} \in \{\mathsf{U}, \mathsf{S}\}$
  $\mathbf{if}\ \exists t \text{ s.t. } \pi_{\mathsf{P}}^t.\mathsf{test} = \mathbf{true}$
  $\mathbf{and}\ \nexists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t' \text{ s.t. } \mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$
    $\mathbf{return}\ \perp$
  $\forall \pi_{\mathsf{P}}^t : \mathbf{if}\ \nexists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t' \text{ s.t. } \mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$
    $\pi_{\mathsf{P}}^t.\mathsf{fr} = \mathbf{false}$
$\mathcal{C} := \mathcal{C} \cup \{(\mathsf{U}, \mathsf{S})\}$
$\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
$\mathbf{return}\ \mathsf{pw}_{\mathsf{US}}$

SENDINIT(U, t, S)
$\mathbf{if}\ \pi_{\mathsf{U}}^t \neq \perp\ \mathbf{return}\ \perp$
$u := (u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$\hat{u} := (\hat{u}_1, ..., \hat{u}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star \tilde{x}, ..., u_\ell \star \tilde{x})$
$\hat{x}^{\mathsf{U}} := (\hat{x}_1^{\mathsf{U}}, ..., \hat{x}_\ell^{\mathsf{U}}) := (\hat{u}_1 \star \tilde{x}, ..., u_\ell \star \tilde{x})$
$\pi_{\mathsf{U}}^t := ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \perp, \perp), \perp, \perp)$
$\pi_{\mathsf{U}}^t.\mathsf{fr} := \perp$
$\mathbf{return}\ (\mathsf{U}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}})$

H(U, S, x^U, x̂^U, x^S, x̂^S, pw, z)
$\mathbf{if}\ T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z] = K \neq \perp$
  $\mathbf{return}\ K$
$\mathbf{if}\ (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}) \in T_s$
  $(b_1, ..., b_\ell) := \mathsf{pw}$
  $\mathbf{if}\ T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] = (\mathsf{U}, (u, \hat{u}), K)$
    $\mathbf{for}\ i \in [\ell]$
      $z_i' := (u_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{S}}, \hat{u}_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{S}}, u_i \cdot g_{b_i}^{-1} \star \hat{x}_i^{\mathsf{S}})$
    $z' := (z_1', ..., z_\ell')$
  $\mathbf{if}\ T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] = (\mathsf{S}, (s, \hat{s}), K)$
    $\mathbf{for}\ i \in [\ell]$
      $z_i' := (s_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{U}}, s_i \cdot g_{b_i}^{-1} \star \hat{x}_i^{\mathsf{U}}, \hat{s}_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{U}})$
    $z' := (z_1', ..., z_\ell')$
  $\mathbf{if}\ z = z'$
    $\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{and}\ \mathsf{pw} = \mathsf{pw}_{\mathsf{US}}\text{: }\mathbf{return}\ K$
    $\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \notin \mathcal{C}\text{: } T_{\text{bad}} := T_{\text{bad}} \cup \{\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z\}$
$T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z] \xleftarrow{\$} \mathcal{K}$
$\mathbf{return}\ T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z]$

SENDRESP(S, t, U, x^U, x̂^U)
$\mathbf{if}\ \pi_{\mathsf{S}}^t \neq \perp\ \mathbf{return}\ \perp$
$s := (s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$\hat{s} := (\hat{s}_1, ..., \hat{s}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star \tilde{x}, ..., s_\ell \star \tilde{x})$
$\hat{x}^{\mathsf{S}} := (\hat{x}_1^{\mathsf{S}}, ..., \hat{x}_\ell^{\mathsf{S}}) := (\hat{s}_1 \star \tilde{x}, ..., \hat{s}_\ell \star \tilde{x})$
$\mathbf{if}\ \exists P \in \mathcal{U} \cup \mathcal{S}, t' \text{ s.t. } \pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
  $\mathbf{return}\ \perp$
$\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
  $\pi_{\mathsf{S}}^t.\mathsf{fr} := \mathbf{true}$
  $\forall \mathsf{pw}, z \text{ s.t. } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z) \in T$
    $(b_1, ..., b_\ell) := \mathsf{pw}$
    $\mathbf{for}\ i \in [\ell]$
      $z_i' := (s_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{U}}, s_i \cdot g_{b_i}^{-1} \star \hat{x}_i^{\mathsf{U}}, \hat{s}_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{U}})$
    $z' := (z_1', ..., z_\ell')$
    $\mathbf{if}\ z = z'$
      $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z)\}$
  $K \xleftarrow{\$} \mathcal{K}$
  $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] := (\mathsf{S}, (s, \hat{s}), K)$
$\mathbf{else}$
  $\pi_{\mathsf{S}}^t.\mathsf{fr} := \mathbf{false}$
  $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
  $\mathbf{for}\ i \in [\ell]$
    $z_i := (s_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{U}}, s_i \cdot g_{b_i}^{-1} \star \hat{x}_i^{\mathsf{U}}, \hat{s}_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{U}})$
  $z := (z_1, ..., z_\ell)$
  $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$
$\pi_{\mathsf{S}}^t := ((s, \hat{s}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
$\mathbf{return}\ (\mathsf{S}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$

SENDTERMINIT(U, t, S, x^S, x̂^S)
$\mathbf{if}\ \pi_{\mathsf{U}}^t \neq ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \perp, \perp), \perp, \perp)$
  $\mathbf{return}\ \perp$
$\mathbf{if}\ \exists \mathsf{P} \in \mathcal{U}, t' \text{ s.t. } \pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
  $\mathbf{return}\ \perp$
$\mathbf{if}\ \exists t' \text{ s.t. } \pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
$\mathbf{and}\ \pi_{\mathsf{S}}^{t'}.\mathsf{fr} = \mathbf{true}$
  $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{true}$
  $(\mathsf{S}, (s, \hat{s}), K) := T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}]$
$\mathbf{else\ if}\ (\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
  $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{true}$
  $\forall \mathsf{pw}, z \text{ s.t. } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z) \in T$
    $(b_1, ..., b_\ell) := \mathsf{pw}$
    $\mathbf{for}\ i \in [\ell]$
      $z_i' := (u_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{S}}, \hat{u}_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{S}}, u_i \cdot g_{b_i}^{-1} \star \hat{x}_i^{\mathsf{S}})$
    $z' := (z_1', ..., z_\ell')$
    $\mathbf{if}\ z = z'$
      $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z)\}$
  $K \xleftarrow{\$} \mathcal{K}$
  $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] := (\mathsf{U}, (u, \hat{u}), K)$
$\mathbf{else}$
  $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{false}$
  $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
  $\mathbf{for}\ i \in [\ell]$
    $z_i := (u_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{S}}, \hat{u}_i \cdot g_{b_i}^{-1} \star x_i^{\mathsf{S}}, u_i \cdot g_{b_i}^{-1} \star \hat{x}_i^{\mathsf{S}})$
  $z := (z_1, ..., z_\ell)$
  $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$
$\pi_{\mathsf{U}}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
$\mathbf{return\ true}$
```

Fig. 8. Game G_6 for the proof of Theorem 1. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$EXECUTE, SENDINIT, SENDRESP, SENDTERMINIT, REVEAL, CORRUPT, TEST, H$\}$. Oracles REVEAL and TEST are defined as in game G_4 in Figure 5. Oracle EXECUTE is defined as in Figure 7. Differences to G_5 are highlighted in blue.

the query to H was made.

To identify the different events, we introduce a new set T_{bad}. For all fresh instances in SENDRESP and SENDTERMINIT, we now iterate over all entries in T that contain the corresponding trace. We check if the given password and z are valid for this trace by computing the real values z' in the same way as for non-fresh instances. If $z = z'$, we add this entry to the set T_{bad} (lines 57-63, 84-90). We essentially do the same when the random oracle H is queried on a trace that appears in T_{s}. Here, the adversary specifies the password and we check if z is valid for that password using the $u_i, \hat{u}_i$ stored in T_{s} for user instances and $s_i, \hat{s}_i$ for server instances. If z is valid and the instance is still fresh, we add the query to T_{bad} (lines 33-45). In case the password was corrupted in the meantime, we output the key stored in T_{s} as introduced in the previous game.

After the adversary terminates, we check T_{bad} whether event $\mathbf{bad}_{\mathsf{pw}}$ (line 09) or event $\mathbf{bad}_{\text{guess}}$ (line 12) occurred. We will bound these events below. First note that whenever $\mathbf{bad}$ is raised in G_5, then either flag $\mathbf{bad}_{\text{guess}}$ or $\mathbf{bad}_{\mathsf{pw}}$ is raised in G_6, thus

$$\Pr[\mathsf{G}_5 \Rightarrow \mathbf{bad}] \leq \Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\mathsf{pw}}] + \Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\text{guess}}] \ .$$

Finally, we bound the probabilities of the two events. We start with $\mathbf{bad}_{\mathsf{pw}}$. In Figure 9, we construct adversary $\mathcal{B}_2$ against $\mathsf{DSim\text{-}GA\text{-}StCDH}$ that simulates G_6.

We show that when $\mathbf{bad}_{\mathsf{pw}}$ occurs, then $\mathcal{B}_2$ can solve $\mathsf{DSim\text{-}GA\text{-}StCDH}$. Hence,

$$\Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\mathsf{pw}}] \leq \mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{DSim\text{-}GA\text{-}StCDH}}(\mathcal{B}_2) \ .$$

Adversary $\mathcal{B}_2$ inputs (x_0, x_1, w_0, w_1), where $x_0 = g_0 \star \tilde{x}$, $x_1 = g_1 \star \tilde{x}$, $w_0 = h_0 \star \tilde{x}$ and $w_1 = h_1 \star \tilde{x}$ for group elements $g_0, g_1, h_0, h_1 \in \mathcal{G}$ chosen uniformly at random. Adversary $\mathcal{B}_2$ also has access to decision oracles $\text{GA-DDH}_{x_j}(w_i, \cdot, \cdot)$ for $(i, j) \in \{0,1\}^2$. It runs adversary $\mathcal{A}$ on (x_0, x_1). Queries to SENDINIT are simulated as follows: $\mathcal{B}_2$ chooses group elements u_i and $\hat{u}_i$ uniformly at random and sets

$$x_i^{\mathsf{U}} = u_i \star w_0 = (u_i \cdot h_0 \cdot g_0^{-1}) \star x_0 = (u_i \cdot h_0 \cdot g_1^{-1}) \star x_1 \ ,$$
$$\hat{x}_i^{\mathsf{U}} = \hat{u}_i \star w_1 = (\hat{u}_i \cdot h_1 \cdot g_0^{-1}) \star x_0 = (\hat{u}_i \cdot h_1 \cdot g_1^{-1}) \star x_1 \ .$$

The simulation of x_i^{S} and $\hat{x}_i^{\mathsf{S}}$ in SENDRESP is done in the same way, choosing random s_i and $\hat{s}_i$. In case the server instance is fresh, we must check if there already exists an entry in T that causes an inconsistency. As in G_6, we iterate over all pw, z, in T that contain the trace of this instance. In particular, we must check whether

$$z_{i,1} = \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) \quad \Leftrightarrow \quad \mathsf{GA\text{-}CDH}_{x_{b_i}}(w_0, x_i^{\mathsf{U}}) = s_i^{-1} \star z_{i,1} \ ,$$
$$z_{i,2} = \mathsf{GA\text{-}CDH}_{x_{b_i}}(\hat{x}_i^{\mathsf{U}}, x_i^{\mathsf{S}}) \quad \Leftrightarrow \quad \mathsf{GA\text{-}CDH}_{x_{b_i}}(w_0, \hat{x}_i^{\mathsf{U}}) = s_i^{-1} \star z_{i,2} \ ,$$
$$z_{i,3} = \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{U}}, \hat{x}_i^{\mathsf{S}}) \quad \Leftrightarrow \quad \mathsf{GA\text{-}CDH}_{x_{b_i}}(w_1, x_i^{\mathsf{U}}) = \hat{s}_i^{-1} \star z_{i,3} \ ,$$

which can be done with the decision oracles $\text{GA-DDH}_{x_{b_i}}(w_j, \cdot, \cdot)$. If all z_i are valid, then we add this entry to T_{bad} (lines 56-59).

If the instance is not fresh, then we have to compute the correct key. We check list T for a valid entry z as explained above and if it exists, we assign this value to the session key (line 66). Otherwise, we choose a random key and add a special entry to T, which instead of z contains the secret group elements s_i and $\hat{s}_i$ (line 69) so that we can patch the random oracle later. SENDTERMINIT is simulated analogously, using the secret group elements u_i and $\hat{u}_i$.

Now we look at the random oracle queries. If the trace is contained in set T_{s} which means the corresponding instance was fresh when the send query was issued, we check if z is valid using the GA-DDH oracle. We do this as described above, depending on whether it is a user or a server instance (lines 25, 31). In case z is valid, we first check if the instance is still fresh (i.e., the password was not corrupted in the meantime) and if this is the case, we add the query to T_{bad} (lines 28, 34). Otherwise, if the password was corrupted and is specified in the query, we return the session key stored in T_{s} (lines 30, 36).

Next, we check if the query matches a special entry in T that was added in SENDRESP or SENDTERMINIT for a non-fresh instance, which means we have to output the same key that was chosen before. Again, we can use the GA-DDH oracle and differentiate between user and server instances (lines 37-44).

After $\mathcal{A}$ terminates with output β', $\mathcal{B}_2$ chooses the passwords which have not been generated in a CORRUPT query yet. If $\mathbf{bad}_{\mathsf{pw}}$ occurred (lines 05-13), then there must be two entries in T_{bad} for the same

```
$\mathcal{B}_2^{\{\text{GA-DDH}_{x_j}(w_i,\cdot,\cdot)\}_{i,j\in\{0,1\}}}(x_0, x_1, w_0, w_1)$
$(\mathcal{C}, T, T_{\mathrm{s}}, T_{\mathrm{bad}}) := (\varnothing, \varnothing, \varnothing, \varnothing)$
$\beta \xleftarrow{\$} \{0,1\}$
$\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
for $(\mathsf{U}, \mathsf{S}) \in \mathcal{U} \times \mathcal{S} \setminus \mathcal{C}$
  $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
if $\exists \mathsf{pw}, \mathsf{pw}', (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, z, z')$
 s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z) \in T_{\mathrm{bad}}$
 and $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}', z') \in T_{\mathrm{bad}}$
  $(b_1, \ldots, b_\ell) := \mathsf{pw}$
  $(b'_1, \ldots, b'_\ell) := \mathsf{pw}'$
  Find first index $i$ such that $b_i \neq b'_i$
  W.l.o.g. let $b_i = 0,\ b'_i = 1$
  if $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] = (\mathsf{U}, (u, \hat{u}), K)$
    Stop with $(x_i^{\mathsf{S}}, u_i^{-1} \star z_{i,1}, \hat{u}_i^{-1} \star z_{i,2}, u_i^{-1} \star z'_{i,1}, \hat{u}_i^{-1} \star z'_{i,2})$
  if $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] = (\mathsf{S}, (s, \hat{s}), K)$
    Stop with $(x_i^{\mathsf{U}}, s_i^{-1} \star z_{i,1}, \hat{s}_i^{-1} \star z_{i,3}, s_i^{-1} \star z'_{i,1}, \hat{s}_i^{-1} \star z'_{i,3})$

SendInit$(\mathsf{U}, t, \mathsf{S})$
if $\pi_{\mathsf{U}}^t \neq \bot$ return $\bot$
$u := u_1, \ldots, u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$\hat{u} := (\hat{u}_1, \ldots, \hat{u}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, \ldots, x_\ell^{\mathsf{U}}) := (u_1 \star w_0, \ldots, u_\ell \star w_0)$
$\hat{x}^{\mathsf{U}} := (\hat{x}_1^{\mathsf{U}}, \ldots, \hat{x}_\ell^{\mathsf{U}}) := (\hat{u}_1 \star w_1, \ldots, \hat{u}_\ell \star w_1)$
$\pi_{\mathsf{U}}^t := ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \bot, \bot), \bot, \bot)$
return $(\mathsf{U}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}})$

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z)$
if $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z] = K \neq \bot$
  return $K$
if $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}) \in T_{\mathrm{s}}$
  $(b_1, \ldots, b_\ell) := \mathsf{pw}$
  if $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] = (\mathsf{U}, (u, \hat{u}), K)$
    if $\text{GA-DDH}_{x_{b_i}}(w_0, x_i^{\mathsf{S}}, u_i^{-1} \star z_{i,1}) = 1\ \forall i \in [\ell]$
    and $\text{GA-DDH}_{x_{b_i}}(w_1, x_i^{\mathsf{S}}, \hat{u}_i^{-1} \star z_{i,2}) = 1\ \forall i \in [\ell]$
    and $\text{GA-DDH}_{x_{b_i}}(w_0, \hat{x}_i^{\mathsf{S}}, u_i^{-1} \star z_{i,3}) = 1\ \forall i \in [\ell]$
      if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
        $T_{\mathrm{bad}} := T_{\mathrm{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z)\}$
      if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$ and $\mathsf{pw} = \mathsf{pw}_{\mathsf{US}}$
        return $K$
  if $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] = (\mathsf{S}, (s, \hat{s}), K)$
    if $\text{GA-DDH}_{x_{b_i}}(w_0, x_i^{\mathsf{U}}, s_i^{-1} \star z_{i,1}) = 1\ \forall i \in [\ell]$
    and $\text{GA-DDH}_{x_{b_i}}(w_0, \hat{x}_i^{\mathsf{U}}, s_i^{-1} \star z_{i,2}) = 1\ \forall i \in [\ell]$
    and $\text{GA-DDH}_{x_{b_i}}(w_1, x_i^{\mathsf{U}}, \hat{s}_i^{-1} \star z_{i,3}) = 1\ \forall i \in [\ell]$
      if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
        $T_{\mathrm{bad}} := T_{\mathrm{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z)\}$
      if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$ and $\mathsf{pw} = \mathsf{pw}_{\mathsf{US}}$
        return $K$
if $\exists (u, \hat{u})$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, (u, \hat{u})) \in T$
  $(b_1, \ldots, b_\ell) := \mathsf{pw}$
  if $\text{GA-DDH}_{x_{b_i}}(w_0, x_i^{\mathsf{S}}, u_i^{-1} \star z_{i,1}) = 1\ \forall i \in [\ell]$
  and $\text{GA-DDH}_{x_{b_i}}(w_1, x_i^{\mathsf{S}}, \hat{u}_i^{-1} \star z_{i,2}) = 1\ \forall i \in [\ell]$
  and $\text{GA-DDH}_{x_{b_i}}(w_0, \hat{x}_i^{\mathsf{S}}, u_i^{-1} \star z_{i,3}) = 1\ \forall i \in [\ell]$
    return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, (u, \hat{u})]$
else if $\exists (s, \hat{s})$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, (s, \hat{s}) \in T$
  $(b_1, \ldots, b_\ell) := \mathsf{pw}$
  if $\text{GA-DDH}_{x_{b_i}}(w_0, x_i^{\mathsf{U}}, s_i^{-1} \star z_{i,1}) = 1\ \forall i \in [\ell]$
  and $\text{GA-DDH}_{x_{b_i}}(w_0, \hat{x}_i^{\mathsf{U}}, s_i^{-1} \star z_{i,2}) = 1\ \forall i \in [\ell]$
  and $\text{GA-DDH}_{x_{b_i}}(w_1, x_i^{\mathsf{U}}, \hat{s}_i^{-1} \star z_{i,3}) = 1\ \forall i \in [\ell]$
    return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, (s, \hat{s})]$
$T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z] \xleftarrow{\$} \mathcal{K}$
return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z]$

SendResp$(\mathsf{S}, t, \mathsf{U}, x^{\mathsf{U}})$
if $\pi_{\mathsf{S}}^t \neq \bot$ return $\bot$
$s := (s_1, \ldots, s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$\hat{s} := (\hat{s}_1, \ldots, \hat{s}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, \ldots, x_\ell^{\mathsf{S}}) := (s_1 \star w_0, \ldots, s_\ell \star w_0)$
$\hat{x}^{\mathsf{S}} := (\hat{x}_1^{\mathsf{S}}, \ldots, \hat{x}_\ell^{\mathsf{S}}) := (\hat{s}_1 \star w_1, \ldots, \hat{s}_\ell \star w_1)$
if $\exists P \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
  return $\bot$
if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
  $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ true
  $\forall \mathsf{pw}, z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z) \in T$
    $(b_1, \ldots, b_\ell) := \mathsf{pw}$
    if $\text{GA-DDH}_{x_{b_i}}(w_0, x_i^{\mathsf{U}}, s_i^{-1} \star z_{i,1}) = 1\ \forall i \in [\ell]$
    and $\text{GA-DDH}_{x_{b_i}}(w_0, \hat{x}_i^{\mathsf{U}}, s_i^{-1} \star z_{i,2}) = 1\ \forall i \in [\ell]$
    and $\text{GA-DDH}_{x_{b_i}}(w_1, x_i^{\mathsf{U}}, \hat{s}_i^{-1} \star z_{i,3}) = 1\ \forall i \in [\ell]$
      $T_{\mathrm{bad}} := T_{\mathrm{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z)\}$
  $K \xleftarrow{\$} \mathcal{K}$
  $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] := (S, (s, \hat{s}), K)$
else
  $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ false
  $(b_1, \ldots, b_\ell) := \mathsf{pw}_{\mathsf{US}}$
  if $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
  and $\text{GA-DDH}_{x_{b_i}}(w_0, x_i^{\mathsf{U}}, s_i^{-1} \star z_{i,1}) = 1\ \forall i \in [\ell]$
  and $\text{GA-DDH}_{x_{b_i}}(w_0, \hat{x}_i^{\mathsf{U}}, s_i^{-1} \star z_{i,2}) = 1\ \forall i \in [\ell]$
  and $\text{GA-DDH}_{x_{b_i}}(w_1, x_i^{\mathsf{U}}, \hat{s}_i^{-1} \star z_{i,3}) = 1\ \forall i \in [\ell]$
    $K := T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z]$
  else
    $K \xleftarrow{\$} \mathcal{K}$
    $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, (s, \hat{s})] := K$
$\pi_{\mathsf{S}}^t := ((s, \hat{s}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
return $(\mathsf{S}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$

SendTermInit$(\mathsf{U}, t, \mathsf{S}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
if $\pi_{\mathsf{U}}^t \neq ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \bot, \bot), \bot, \bot)$ return $\bot$
if $\exists \mathsf{P} \in \mathcal{U}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$ return $\bot$
if $\exists t'$ s. t. $\pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$ and $\pi_{\mathsf{S}}^{t'}.\mathsf{fr} =$ true
  $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ true
  $(\mathsf{S}, (s, \hat{s}), K) := T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}]$
else if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
  $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ true
  $\forall \mathsf{pw}, z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z) \in T$
    $(b_1, \ldots, b_\ell) := \mathsf{pw}$
    if $\text{GA-DDH}_{x_{b_i}}(w_0, x_i^{\mathsf{S}}, u_i^{-1} \star z_{i,1}) = 1\ \forall i \in [\ell]$
    and $\text{GA-DDH}_{x_{b_i}}(w_1, x_i^{\mathsf{S}}, \hat{u}_i^{-1} \star z_{i,2}) = 1\ \forall i \in [\ell]$
    and $\text{GA-DDH}_{x_{b_i}}(w_0, \hat{x}_i^{\mathsf{S}}, u_i^{-1} \star z_{i,3}) = 1\ \forall i \in [\ell]$
      $T_{\mathrm{bad}} := T_{\mathrm{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z)\}$
  $K \xleftarrow{\$} \mathcal{K}$
  $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] := (\mathsf{U}, (u, \hat{u}), K)$
else
  $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ false
  $(b_1, \ldots, b_\ell) := \mathsf{pw}_{\mathsf{US}}$
  if $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
  and $\text{GA-DDH}_{x_{b_i}}(w_0, x_i^{\mathsf{S}}, u_i^{-1} \star z_{i,1}) = 1\ \forall i \in [\ell]$
  and $\text{GA-DDH}_{x_{b_i}}(w_1, x_i^{\mathsf{S}}, \hat{u}_i^{-1} \star z_{i,2}) = 1\ \forall i \in [\ell]$
  and $\text{GA-DDH}_{x_{b_i}}(w_0, \hat{x}_i^{\mathsf{S}}, u_i^{-1} \star z_{i,3}) = 1\ \forall i \in [\ell]$
    $K := T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z]$
  else
    $K \xleftarrow{\$} \mathcal{K}$
    $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, (u, \hat{u})] := K$
$\pi_{\mathsf{U}}^t := ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
return true
```

Fig. 9. Adversary $\mathcal{B}_2$ against $\mathsf{DSim\text{-}GA\text{-}StCDH}$ for the proof of Theorem 1. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$Execute, SendInit, SendResp, SendTermInit, Reveal, Corrupt, Test, $\mathsf{H}\}$. Oracles Execute, Reveal, Corrupt and Test are defined as in G_6. Lines written in blue show how $\mathcal{B}_2$ simulates the game.

trace and different passwords $\mathsf{pw} \neq \mathsf{pw}'$ along with values z and z'. Let i be the first index where the two passwords differ, i.e., $b_i \neq b'_i$. Without loss of generality assume that $b_i = 0$ and $b'_i = 1$, otherwise swap pw, z and pw', z'. If the entries in T_{bad} are those of a user instance, we retrieve the secret group elements $u_, \hat{u}_i$ from T_{s}.

Recall that the DSim-GA-StCDH problem requires to compute the four values $y_0 = \mathsf{GA\text{-}CDH}_{x_0}(w_0, y)$, $y_1 = \mathsf{GA\text{-}CDH}_{x_0}(w_1, y)$, $y_2 = \mathsf{GA\text{-}CDH}_{x_1}(w_0, y)$ and $y_3 = \mathsf{GA\text{-}CDH}_{x_1}(w_1, y)$, where y can be chosen by the adversary. $\mathcal{B}_2$ sets $y = x_i^{\mathsf{S}}$, and outputs y and

$$\begin{aligned}
y_0 &= u_i^{-1} \star z_{i,1} = \mathsf{GA\text{-}CDH}_{x_0}(u_i^{-1} \star x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = \mathsf{GA\text{-}CDH}_{x_0}(w_0, x_i^{\mathsf{S}}) \ , \\
y_1 &= \hat{u}_i^{-1} \star z_{i,2} = \mathsf{GA\text{-}CDH}_{x_0}(\hat{u}_i^{-1} \star \hat{x}_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = \mathsf{GA\text{-}CDH}_{x_0}(w_1, x_i^{\mathsf{S}}) \ , \\
y_2 &= u_i^{-1} \star z'_{i,1} = \mathsf{GA\text{-}CDH}_{x_1}(u_i^{-1} \star x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = \mathsf{GA\text{-}CDH}_{x_1}(w_0, x_i^{\mathsf{S}}) \ , \\
y_3 &= \hat{u}_i^{-1} \star z'_{i,2} = \mathsf{GA\text{-}CDH}_{x_1}(\hat{u}_i^{-1} \star \hat{x}_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = \mathsf{GA\text{-}CDH}_{x_1}(w_1, x_i^{\mathsf{S}}) \ .
\end{aligned}$$

If the instance is a server instance, $\mathcal{B}_2$ outputs $(y, y_0, y_1, y_2, y_3) = (x_i^{\mathsf{U}}, s_i^{-1} \star z_{i,1}, \hat{s}_i^{-1} \star z_{i,3}, s_i^{-1} \star z'_{i,1}, \hat{s}_i^{-1} \star z'_{i,3})$. This concludes the analysis of $\mathbf{bad}_{\mathsf{pw}}$.

Next, we analyze event $\mathbf{bad}_{\text{guess}}$. Recall that $\mathbf{bad}_{\text{guess}}$ happens only if $\mathbf{bad}_{\mathsf{pw}}$ does not happen. Hence, for each instance there is at most one entry in T_{bad} and the size of T_{bad} is at most q_s. As all entries were added before the corresponding password was sampled, the probability is bounded by

$$\Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\text{guess}}] \leq \frac{q_s}{|\mathcal{PW}|} \ .$$

Finally, note that if none of the bad events happens in G_6, all session keys output by TEST are uniformly random and the adversary can only guess β. Hence, $\Pr[\mathsf{G}_6 \Rightarrow 1] = \frac{1}{2}$. Collecting the probabilities and using Equation Lemma 1 yields the bound in Theorem 1. □

7 Com-GA-PAKE$_\ell$: Three-Round PAKE from Group Actions

In this section we present a second modification of GA-PAKE$_\ell$, which can be securely instantiated with an EGAT. The protocol Com-GA-PAKE$_\ell$ extends GA-PAKE$_\ell$ by a commitment that has to be sent before sending the actual messages. In the first round, the server sends a commitment on those set elements it will send in the next round, thus ensuring that the server cannot choose the set elements depending on the message it receives from the user. This is the crucial step in the attack against GA-PAKE$_\ell$. In the second round, the user sends its message to the server and only after receiving that message, the servers sends its message to the user. While this protocol adds two rounds to the original protocol, the total computational cost is lower than for X-GA-PAKE$_\ell$.

7.1 Description of the Protocol

The setup for Com-GA-PAKE$_\ell$ is the same as for GA-PAKE$_\ell$, where the $\mathsf{crs} = (x_0, x_1)$ comprises two set elements, and the shared password is a bit string $(b_1, \ldots, b_\ell)$ of length ℓ.

The difference to GA-PAKE$_\ell$ is that before sending the set elements x^{U} and x^{S}, the server commits on x^{S}. More precisely, in the first round the server sends $\mathsf{com} = \mathsf{G}(x^{\mathsf{S}})$, where $\mathsf{G} : \{0,1\}^* \to \{0,1\}^\lambda$ is a hash function and λ is a parameter of the protocol. The user only accepts the session key after verifying that com corresponds to x^{S}. The session key K is derived as in GA-PAKE$_\ell$ but additionally takes the commitment com as input. The protocol is sketched in Figure 10. The security of Com-GA-PAKE$_\ell$ for EGATs is established in the following theorem.

Theorem 2 (Security of Com-GA-PAKE$_\ell$). *For any adversary $\mathcal{A}$ against* Com-GA-PAKE$_\ell$ *that issues at most q_e execute queries, q_s send queries and at most q_{G} and q_{H} queries to random oracles* G *and* H*, there exist an adversary $\mathcal{B}_1$ against* GA-StCDH *and an adversary $\mathcal{B}_2$ against* GA-GapCDH *such that*

$$\mathsf{Adv}_{\mathsf{Com\text{-}GA\text{-}PAKE}_\ell}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{GA\text{-}StCDH}}(\mathcal{B}_1) + q_s \ell \cdot \sqrt{\mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{GA\text{-}GapCDH}}(\mathcal{B}_2)} + \frac{(q_s + q_e)^2}{|\mathcal{G}|^\ell} + \frac{q_{\mathsf{G}} q_s}{|\mathcal{G}|^\ell} + \frac{2 \cdot (q_{\mathsf{G}} + q_s + q_e)^2}{2^\lambda} + \frac{q_s}{|\mathcal{PW}|} \ ,$$

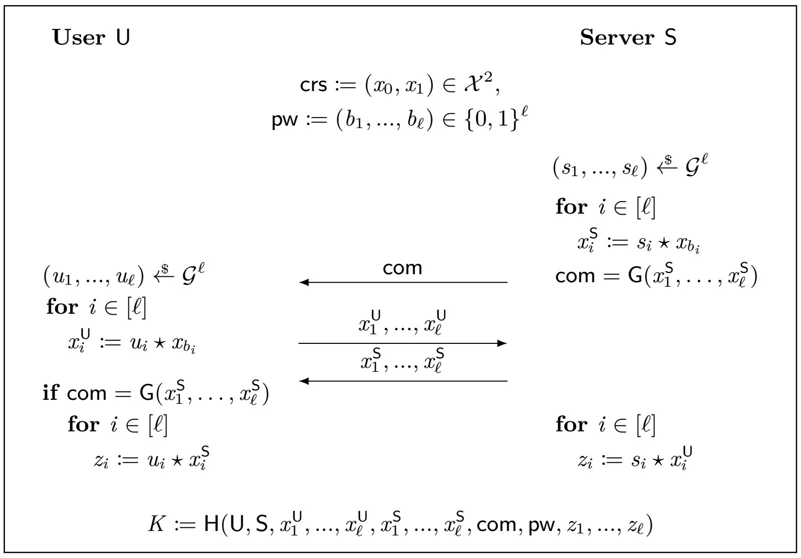

User U
Server S
$\mathsf{crs} := (x_0, x_1) \in \mathcal{X}^2$,
$\mathsf{pw} := (b_1, ..., b_\ell) \in \{0,1\}^\ell$
$(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
for $i \in [\ell]$
$x_i^{\mathsf{S}} := s_i \star x_{b_i}$
$\mathsf{com} = \mathsf{G}(x_1^{\mathsf{S}}, \dots, x_\ell^{\mathsf{S}})$
com
$(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
for $i \in [\ell]$
$x_i^{\mathsf{U}} := u_i \star x_{b_i}$
$x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}$
$x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}$
if $\mathsf{com} = \mathsf{G}(x_1^{\mathsf{S}}, \dots, x_\ell^{\mathsf{S}})$
for $i \in [\ell]$
$z_i := u_i \star x_i^{\mathsf{S}}$
for $i \in [\ell]$
$z_i := s_i \star x_i^{\mathsf{U}}$
$K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}, x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z_1, ..., z_\ell)$

Fig. 10. PAKE protocol $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$ from group actions.

where λ is the output length of G in bits.

The proof is similar to the one of Theorem 1 so we will only sketch it here. The full proof is given in Appendix E.

Proof (Sketch). After ensuring that all traces are unique, we need to deal with the commitment and in particular collisions. First, we require that there are never two inputs to the random oracle G that return the same commitment. This is to ensure that the adversary cannot open a commitment to a different value, which might depend on previous messages.

Second, we need to ensure that after the adversary has seen a commitment, it does not query G on the input, which is the hiding property of the commitment. What we actually do here is that we choose a random commitment in the first round. Only later we choose the input and patch the random oracle accordingly.

Now we can replace the session keys of instances which are used in execute queries. Here, the freshness condition allows the adversary to corrupt the password. However, as both x^{S} and x^{U} are generated by the experiment, the only chance to notice this change is to solve the $\mathsf{GA\text{-}StCDH}$ problem, where the decision oracle is required to simulate instances correctly.

In order to replace the session keys of fresh instances which are used in send queries, we make the key independent of the password. The session key of a fresh instance is now defined by the trace of that instance. The only issue that may arise here is an inconsistency between the session key that is derived using the trace and the session key that is derived using the random oracle H. Whenever such an inconsistency occurs, we differentiate between two cases:

– There exists more than one valid entry in T_{H} for the same trace of a fresh instance, but different passwords.
– There exists a valid entry in T_{H} for the trace of a fresh instance and the correct password, where the password was not corrupted when the query to H was made.

Finally, we bound the probabilities of the two cases. Similar to Theorem 1, we will define a new computational problem that reflects exactly the interaction in the protocol. We show that this problem is implied by $\mathsf{GA\text{-}GapCDH}$ using the reset lemma. The general idea is that the adversary can always compute the session key for one password guess, but not for a second one. After excluding this, we choose the actual password, which is possible because session keys are computed independently of the password. Thus, looking at one fixed instance, the probability that the adversary guessed the password correctly is $1/|\mathcal{PW}|$. $\square$

8 Variants of the PAKE Protocols

Both protocols $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ and $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$ require that the user and the server generate multiple random group elements and evaluate their action on certain set elements. In this section we present two optimizations that allow us to reduce the number of random group elements and more importantly the number of necessary group action evaluations.

8.1 Increasing the Number of Public Parameters

In $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ and $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$ the common reference string is set to $\mathsf{crs} \coloneqq (x_0, x_1) \in \mathcal{X}^2$. Increasing the number of public parameters allows to reduce the number of group action evaluations in the execution of the protocol. The idea is similar to the optimizations deployed to speed up the CSIDH-based signatures schemes SeaSign [15] and CSI-FiSh [11]. We refer to Table 1 in the introduction for an overview and example of the parameter choice.

We explain the changes on the basis of protocol $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$. A security analysis for the variant is provided in Appendix D.1. The analysis for the variant of $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$ is similar and is given in Appendix E.2. For some positive integer N dividing ℓ, we set

$$\mathsf{crs} \coloneqq (x_0, \ldots, x_{2^N-1}) \in \mathcal{X}^{2^N}.$$

As before, the password is a bitstring of length ℓ, but now we divide it into ℓ/N blocks of length N and write

$$\mathsf{pw} = (b_1, ..., b_{\ell/N}) \in \{0, ..., 2^N - 1\}^{\ell/N}.$$

In particular x_{b_i} refers to one of the 2^N different set elements in the crs. The general outline of the protocol does not change. The only difference is that in the first step both the server and the user only generate $2 \cdot \ell/N$ random group elements (instead of $2 \cdot \ell$). Hence they only need to perform $2 \cdot \ell/N$ group action evaluations in the first round and $3 \cdot \ell/N$ evaluations in the session key derivation. We write $\mathsf{X\text{-}GA\text{-}PAKE}_{\ell,N}$ for this variant of the protocol.

8.2 Using Twists in the Setup

Both $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ and $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$ require that some trusted party generates two random set elements $\mathsf{crs} = (x_0, x_1)$. Here, we shortly discuss the setup where x_1 is replaced by the twist of x_0, i.e. $\mathsf{crs} \coloneqq (x_0, x_0^t)$.

This simplification is particularly helpful when applied to one of the variants from the previous subsection. These modified versions require to generate 2^N random set elements for the crs. Using twists it suffices to generate 2^{N-1} random set elements. More precisely, a trusted party provides $(x_0, \ldots, x_{2^{N-1}-1}) \in \mathcal{X}^{2^{N-1}}$, then user and server set $x_{i+2^{N-1}} = x_i^t$ for each $i \in [0, 2^{N-1} - 1]$.

The security of $\mathsf{X\text{-}GA\text{-}PAKE}^{\mathsf{t}}_\ell$ and $\mathsf{Com\text{-}GA\text{-}PAKE}^{\mathsf{t}}_\ell$ (the twisted versions of $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ and $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$) are discussed in Appendices D.2 and E.2, respectively.

Acknowledgments

Thorsten Eisenhofer, Eike Kiltz, Sabrina Kunzweiler and Doreen Riepel were supported by the DFG under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

References

1. Abdalla, M., Barbosa, M.: Perfect forward security of SPAKE2. Cryptology ePrint Archive, Report 2019/1194 (2019), https://eprint.iacr.org/2019/1194
2. Abdalla, M., Fouque, P.A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (Jan 2005). https://doi.org/10.1007/978-3-540-30580-4_6
3. Abdalla, M., Haase, B., Hesse, J.: Security analysis of CPace. Cryptology ePrint Archive, Report 2021/114 (2021), https://eprint.iacr.org/2021/114

4. Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (Feb 2005). https://doi.org/10.1007/978-3-540-30574-3_14
5. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 411–439. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_14
6. Azarderakhsh, R., Jao, D., Koziel, B., LeGrow, J.T., Soukharev, V., Taraskin, O.: How not to create an isogeny-based PAKE. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) ACNS 20, Part I. LNCS, vol. 12146, pp. 169–186. Springer, Heidelberg (Oct 2020). https://doi.org/10.1007/978-3-030-57808-4_9
7. Bellare, M., Palacio, A.: GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 162–177. Springer, Heidelberg (Aug 2002). https://doi.org/10.1007/3-540-45708-9_11
8. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (May 2000). https://doi.org/10.1007/3-540-45539-6_11
9. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: 1992 IEEE Symposium on Security and Privacy. pp. 72–84. IEEE Computer Society Press (May 1992). https://doi.org/10.1109/RISP.1992.213269
10. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for SPHFs and efficient one-round PAKE protocols. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 449–475. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40041-4_25
11. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 227–247. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34578-5_9
12. Canetti, R., Dachman-Soled, D., Vaikuntanathan, V., Wee, H.: Efficient password authenticated key exchange via oblivious transfer. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 449–466. Springer, Heidelberg (May 2012). https://doi.org/10.1007/978-3-642-30057-8_27
13. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03332-3_15
14. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
15. De Feo, L., Galbraith, S.D.: SeaSign: Compact isogeny signatures from class group actions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 759–789. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17659-4_26
16. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Online-extractability in the quantum random-oracle model. Cryptology ePrint Archive, Report 2021/280 (2021), https://eprint.iacr.org/2021/280
17. Fujioka, A., Takashima, K., Yoneyama, K.: One-round authenticated group key exchange from isogenies. In: Steinfeld, R., Yuen, T.H. (eds.) ProvSec 2019. LNCS, vol. 11821, pp. 330–338. Springer, Heidelberg (Oct 2019). https://doi.org/10.1007/978-3-030-31919-9_20
18. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9_33, https://eprint.iacr.org/2003/032.ps.gz
19. Groce, A., Katz, J.: A new framework for efficient password-based authenticated key exchange. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 2010. pp. 516–525. ACM Press (Oct 2010). https://doi.org/10.1145/1866307.1866365
20. Haase, B., Labrique, B.: AuCPace: Efficient verifier-based PAKE protocol tailored for the IIoT. IACR TCHES **2019**(2), 1–48 (2019). https://doi.org/10.13154/tches.v2019.i2.1-48, https://tches.iacr.org/index.php/TCHES/article/view/7384
21. Hao, F., Ryan, P.: J-PAKE: Authenticated key exchange without PKI. Cryptology ePrint Archive, Report 2010/190 (2010), https://eprint.iacr.org/2010/190
22. Jablon, D.P.: Strong password-only authenticated key exchange. ACM SIGCOMM Computer Communication Review **26**(5), 5–26 (1996)
23. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.Y. (ed.) Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011. pp. 19–34. Springer, Heidelberg (Nov / Dec 2011). https://doi.org/10.1007/978-3-642-25405-5_2
24. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_18
25. Kawashima, T., Takashima, K., Aikawa, Y., Takagi, T.: An efficient authenticated key exchange from random self-reducibility on CSIDH. In: Hong, D. (ed.) ICISC 20. LNCS, vol. 12593, pp. 58–84. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-68890-5_4

26. de Kock, B., Gjøsteen, K., Veroni, M.: Practical isogeny-based key-exchange with optimal tightness. In: Dunkelman, O., Jacobson, Jr., M.J., O'Flynn, C. (eds.) Selected Areas in Cryptography. pp. 451–479. Springer International Publishing, Cham (2021)
27. Lai, Y.F., Galbraith, S.D., de Saint Guilhem, C.: Compact, efficient and UC-secure isogeny-based oblivious transfer. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 213–241. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_8
28. Moriya, T., Onuki, H., Takagi, T.: SiGamal: A supersingular isogeny-based PKE and its application to a PRF. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 551–580. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_19
29. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5_31
30. Pointcheval, D., Wang, G.: VTBPEKE: Verifier-based two-basis password exponential key exchange. In: Karri, R., Sinanoglu, O., Sadeghi, A.R., Yi, X. (eds.) ASIACCS 17. pp. 301–312. ACM Press (Apr 2017)
31. Rostovtsev, A., Stolbunov, A.: Public-Key Cryptosystem Based On Isogenies. Cryptology ePrint Archive, Report 2006/145 (2006), https://eprint.iacr.org/2006/145
32. Soukharev, V., Hess, B.: PQDH: A quantum-safe replacement for Diffie-Hellman based on SIDH. Cryptology ePrint Archive, Report 2019/730 (2019), https://eprint.iacr.org/2019/730
33. Taraskin, O., Soukharev, V., Jao, D., LeGrow, J.: An isogeny-based password-authenticated key establishment protocol. Cryptology ePrint Archive, Report 2018/886 (2018), https://eprint.iacr.org/2018/886
34. Terada, S., Yoneyama, K.: Password-based authenticated key exchange from standard isogeny assumptions. In: Steinfeld, R., Yuen, T.H. (eds.) ProvSec 2019. LNCS, vol. 11821, pp. 41–56. Springer, Heidelberg (Oct 2019). https://doi.org/10.1007/978-3-030-31919-9_3
35. Yoneyama, K.: Post-quantum variants of iso/iec standards: Compact chosen ciphertext secure key encapsulation mechanism from isogeny. In: Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop. p. 13–21. SSR'19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3338500.3360336

Overview of Appendices

In Appendix A, we establish the relation between our assumptions and the computational assumptions in [5]. In Appendix B, we recall the Reset Lemma which is used in several security reductions. In Appendix C, we show that our first attempt $\mathsf{GA\text{-}PAKE}_\ell$ would be secure if an adversary was not able to compute twists efficiently. In Appendix D, we provide a security analysis for the two variants of $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$. Appendix E contains the security analysis of $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$ as well as an analysis for different variants of the protocol. Finally, Appendix F deals with the perfect forward secrecy of $\mathsf{GA\text{-}PAKE}_\ell$, $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ and $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$.

A Relation to Assumptions from ADMP20 [5]

We first recall the definitions of a weak unpredictable permutation and weak unpredictable group action from [5] and then relate them to the group action computational Diffie-Hellman problem from Section 3.

Definition 10 (Weak Unpredictable Permutation [5]). *Let K, X and Y be sets indexed by λ, and let D_K and D_X be distributions on K and X respectively. Let $F_k^\$$ be a* randomized *oracle that when queried, samples $x \leftarrow D_X$ and outputs $(x, F(k,x))$. A (D_K, D_X)-weak UP (wUP) is a family of efficiently computable permutations $\{F(k,\cdot) : X \to X\}_{k \in K}$ such that for all PPT adversaries $\mathcal{A}$ we have*

$$\Pr[\mathcal{A}^{F_k^\$}(x^*) = F(k, x^*)] \leq negl(\lambda),$$

where $k \leftarrow D_K$, and $x^* \leftarrow D_X$. If D_K and D_X are uniform distributions, then we simply speak of a wUP family.

Definition 11 (Weak Unpredictable Group Action [5]). *A group action $(\mathcal{G}, \mathcal{X}, \star)$ is $(D_\mathcal{G}, D_\mathcal{X})$-weakly unpredictable if the family of efficiently computable permutations $\{\pi_g : X \to X\}_{g \in \mathcal{G}}$ is $(D_\mathcal{G}, D_\mathcal{X})$-weakly unpredictable, where π_g is defined as $\pi_g : x \mapsto g \star x$ and $D_\mathcal{G}$, $D_\mathcal{X}$ are distributions on $\mathcal{G}$, $\mathcal{X}$ respectively.*

Proposition 2. *If the group action computational Diffie-Hellman problem is hard for a group action, then the group action is weak unpredictable.*

Proof. Let $\mathcal{A}$ be an adversary against weak unpredictability, i.e., given access to an oracle π_g, where $g \leftarrow D_\mathcal{G}$ and $x^* \leftarrow D_\mathcal{X}$, $\mathcal{A}$ will compute $g \star x^*$. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ against the group action computational Diffie-Hellman problem. $\mathcal{B}$ inputs $(x, y) = (g \star \tilde{x}, h \star \tilde{x})$ and has to compute $gh \star \tilde{x}$. Therefore, it runs $\mathcal{A}$ on $x^* := y$. On a query to π_g, $\mathcal{B}$ chooses $h' \leftarrow D_\mathcal{G}$ and computes $x' = h' \star \tilde{x}$ (instead of $x' \leftarrow D_\mathcal{X}$) and returns $(x', h' \star x)$ to $\mathcal{A}$. Note that $h' \star x = g \star x'$. Finally, $\mathcal{A}$ outputs $g \star x^*$, which $\mathcal{B}$ forwards as a solution to the group action computational Diffie-Hellman problem, since $g \star x^* = gh \star \tilde{x}$. $\square$

The other direction is a bit more intricate. We can easily show that a more general definition of the group action computational Diffie-Hellman problem, namely where the basis is not the origin element $\tilde{x}$, but a random set element, is tightly implied by the weak unpredictability property. However, we can also use the standard group action computational Diffie-Hellman problem, but with a non-tight reduction. Therefore, we use the fact that $\mathsf{GA\text{-}CDH}_x(y_0, y_1) = \mathsf{GA\text{-}CDH}(\mathsf{GA\text{-}CDH}(y_0, y_1), x^t)$.

Proposition 3. *If a group action is weak unpredictable, then the group action computational Diffie-Hellman problem is hard for the group action.*

Proof. Let $\mathcal{A}$ be an adversary against the group action computational Diffie-Hellman problem, i.e., on input $(x, y) = (g \star \tilde{x}, h \star \tilde{x})$ it computes $gh \star \tilde{x}$. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ against weak unpredictability. $\mathcal{B}$ inputs $x^* \leftarrow D_\mathcal{X}$ and has access to an oracle π_g, where $g \leftarrow D_\mathcal{G}$. It queries π_g once to receive $(x, g \star x)$. Let $x = g' \star \tilde{x}$. $\mathcal{B}$ runs $\mathcal{A}$ on $(g \star x, x^*)$ and $\mathcal{A}$ outputs $gg' \star x^*$. Now $\mathcal{B}$ runs $\mathcal{A}$ a second time, this time on input $(gg' \star x^*, x^t)$. Note that $\mathsf{GA\text{-}CDH}(gg' \star x^*, x^t) = gg'(g')^{-1} \star x^*$, which is the solution for the weak unpredictability experiment. $\square$

B Reset Lemma

We recall the reset lemma given by Bellare and Palacio [7, Lemma 3.1], which we will need to relate some of our new assumptions.

Lemma 2 (Reset Lemma [7]). *Fix a non-empty set H. Let $\mathcal{B}$ be an adversary that on input (I, h) returns a pair, where the first element is a bit b and the second element σ is some side output. Let IG be a randomized algorithm that we call instance generator. The accepting probability of $\mathcal{B}$ is defined as*

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

The reset algorithm $\mathcal{R}_\mathcal{B}$ associated to $\mathcal{B}$ is defined as in Figure 11. Let $\mathrm{res} = \Pr[b^* = 1 : I \xleftarrow{\$} \mathsf{IG}; (b^*, \sigma, \sigma') \xleftarrow{\$} \mathcal{R}_\mathcal{B}(I)]$. *Then*

$$\mathrm{acc} \le \sqrt{\mathrm{res}} + \frac{1}{|H|} .$$

```
$\underline{\mathcal{R}_\mathcal{B}(I)}$
00 Pick random coins $\rho$ for $\mathcal{B}$
01 $h \xleftarrow{\$} H$; $(b, \sigma) \leftarrow \mathcal{B}(I, h; \rho)$
02 $h' \xleftarrow{\$} H$; $(b', \sigma') \leftarrow \mathcal{B}(I, h'; \rho)$
03 if $b = b' = 1$ and $h \neq h'$
04     return $(1, \sigma, \sigma')$
05 return $(0, \epsilon, \epsilon)$
```

Fig. 11. Reset algorithm $\mathcal{R}_\mathcal{B}$ associated to adversary $\mathcal{B}$.

C Security of $\mathsf{GA\text{-}PAKE}_\ell$ in the EGA Setting

We introduce a new security assumption for EGA and REGA, namely the simultaneous $\mathsf{GA\text{-}StCDH}$, which is used in the traditional Diffie-Hellman setting to prove security of several PAKE protocols [30,3].

Definition 12 (Simultaneous $\mathsf{GA\text{-}StCDH}$ ($\mathsf{Sim\text{-}GA\text{-}StCDH}$)). *On input $(x, x_0, x_1) = (g\star\tilde{x}, g_0\star\tilde{x}, g_1\star\tilde{x})$, the $\mathsf{Sim\text{-}GA\text{-}StCDH}$ problem requires to compute the set elements $y_0 = ({g_0}^{-1} \cdot g) \star y$, $y_1 = ({g_1}^{-1} \cdot g) \star y$, where $y \in \mathcal{X}$ can be chosen by the adversary $\mathcal{A}$. To an effective group action $\mathsf{XXX} \in \{\mathsf{EGA}, \mathsf{REGA}\}$, we associate the advantage function of $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathsf{XXX}}^{\mathsf{Sim\text{-}GA\text{-}StCDH}}(\mathcal{A}) := \Pr\left[\begin{array}{l} y_0 = \mathsf{GA\text{-}CDH}_{x_0}(x, y), \\ y_1 = \mathsf{GA\text{-}CDH}_{x_1}(x, y) \end{array}\middle|\begin{array}{c} (g, g_0, g_1) \xleftarrow{\$} \mathcal{G}^3 \\ (x, x_0, x_1) := (g \star \tilde{x}, g_0 \star \tilde{x}, g_1 \star \tilde{x}) \\ (y, y_0, y_1) \leftarrow \mathcal{A}^{\mathrm{O}}(x, x_0, x_1) \end{array}\right]$$

where $\mathrm{O} = \{\mathrm{GA\text{-}DDH}_{x_0}(x, \cdot, \cdot), \mathrm{GA\text{-}DDH}_{x_1}(x, \cdot, \cdot)\}$.

Note that the $\mathsf{Sim\text{-}GA\text{-}StCDH}$ problem is easy in the EGAT and REGAT setting, where the group action allows to twist elements efficiently (see Definition 4). The attack works exactly as the attack against $\mathsf{GA\text{-}PAKE}_\ell$ given in Section 5. An adversary can solve the $\mathsf{Sim\text{-}GA\text{-}StCDH}$ problem by choosing $(y, y_0, y_1) = (x^t, x_0^t, x_1^t)$.

On the other hand, if a group action does not allow for efficient twisting, the $\mathsf{Sim\text{-}GA\text{-}StCDH}$ problem is believed to be hard. Pointcheval and Wand [30] analyzed the generic hardness of the assumption in the traditional Diffie-Hellman setting, where $\mathcal{G} = \mathcal{X} = \mathbb{F}_p^*$.

Theorem 3 (Security of $\mathsf{GA\text{-}PAKE}_\ell$). *For any adversary $\mathcal{A}$ against $\mathsf{GA\text{-}PAKE}_\ell$ that issues at most q_e execute queries and q_s send queries and where H is modeled as a random oracle, there exist adversary $\mathcal{B}_1$ against $\mathsf{GA\text{-}StCDH}$ and adversary $\mathcal{B}_2$ against $\mathsf{Sim\text{-}GA\text{-}StCDH}$ such that*

$$\mathsf{Adv}_{\mathsf{GA\text{-}PAKE}_\ell}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{GA\text{-}StCDH}}(\mathcal{B}_1) + \mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{Sim\text{-}GA\text{-}StCDH}}(\mathcal{B}_2) + \frac{q_s}{|\mathcal{PW}|} + \frac{(q_s + q_e)^2}{|\mathcal{G}|^\ell} \quad .$$

Proof. The proof follows the one of Theorem 1 very closely, so we will give the full games and adversaries in pseudocode, but leave descriptions short.

Let $\mathcal{A}$ be an adversary against $\mathsf{GA\text{-}PAKE}_\ell$. Consider the games in Figure 12.

GAME G_0. This is the original game, hence

$$\mathsf{Adv}_{\mathsf{GA\text{-}PAKE}_\ell}(\mathcal{A}) \leq |\Pr[\mathsf{G}_0 \Rightarrow 1] - 1/2| \quad .$$

GAME G_1. In game G_1, we raise flag $\mathbf{bad}_{\mathrm{coll}}$ and output $\perp$ whenever a server instance computes the same trace as any other accepted instance or a user instance computes the same trace as any other accepted user instance. As user and server messages consist of ℓ group elements each, we have

$$|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_0 \Rightarrow 1]| \leq \Pr[\mathbf{bad}_{\mathrm{coll}}] \leq \frac{(q_e + q_s)^2}{|\mathcal{G}|^\ell} \quad .$$

GAME G_2. In game G_2, we make the freshness explicit. To each oracle π_{P}^t, we assign an additional variable $\pi_{\mathsf{P}}^t.\mathsf{fr}$ which is updated during the game. These are only a conceptual changes, hence

$$\Pr[\mathsf{G}_2 \Rightarrow 1] = \Pr[\mathsf{G}_1 \Rightarrow 1] \quad .$$

GAME G_3. In game G_3, we choose random keys for all instances queried to EXECUTE. We construct adversary $\mathcal{B}_1$ against $\mathsf{GA\text{-}StCDH}$ in Figure 13 and show that

$$|\Pr[\mathsf{G}_3 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1]| \leq \mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{GA\text{-}StCDH}}(\mathcal{B}_1) \quad . \qquad (1)$$

Adversary $\mathcal{B}_1$ inputs a $\mathsf{GA\text{-}StCDH}$ challenge $(x, y) = (g \star \tilde{x}, h \star \tilde{x})$ and has access to a decision oracle $\mathrm{GA\text{-}DDH}(x, \cdot, \cdot)$. First, it generates the crs elements (x_0, x_1) as in game G_3 and then runs adversary $\mathcal{A}$.

```
GAMES G_0-G_4
$(g_0, g_1) \xleftarrow{\$} \mathcal{G}^2$
$(x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x})$
$(\mathcal{C}, T) := (\varnothing, \varnothing)$
$\mathbf{bad}_{\mathrm{coll}} := \mathbf{false}$
$\beta \xleftarrow{\$} \{0,1\}$
for $(\mathsf{U}, \mathsf{S}) \in \mathcal{U} \times \mathcal{S}$
    $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
$\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
return $[\![\beta = \beta']\!]$

EXECUTE$(\mathsf{U}, t_0, \mathsf{S}, t_1)$
if $\pi_{\mathsf{U}}^{t_0} \neq \bot$ or $\pi_{\mathsf{S}}^{t_1} \neq \bot$
    return $\bot$
$(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$                                   //G_0-G_3
$(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star x_{b_1}, ..., u_\ell \star x_{b_\ell})$        //G_0-G_3
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star x_{b_1}, ..., s_\ell \star x_{b_\ell})$        //G_0-G_3
$z := (z_1, ..., z_\ell) := (u_1 \star x_1^{\mathsf{S}}, ..., u_\ell \star x_\ell^{\mathsf{S}})$            //G_0-G_3
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star \tilde{x}, ..., u_\ell \star \tilde{x})$               //G_4
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star \tilde{x}, ..., s_\ell \star \tilde{x})$               //G_4
if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$         //G_1-G_4
    $\mathbf{bad}_{\mathrm{coll}} := \mathbf{true}$                                   //G_1-G_4
    return $\bot$                                            //G_1-G_4
$K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$                       //G_0-G_2
$K \xleftarrow{\$} \mathcal{K}$                                               //G_3-G_4
$\pi_{\mathsf{U}}^{t_0} := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
$\pi_{\mathsf{S}}^{t_1} := ((s_1, ..., s_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
$(\pi_{\mathsf{U}}^{t_0}.\mathsf{fr}, \pi_{\mathsf{S}}^{t_1}.\mathsf{fr}) := (\mathbf{true}, \mathbf{true})$                       //G_2-G_4
return $(\mathsf{U}, x^{\mathsf{U}}, \mathsf{S}, x^{\mathsf{S}})$

REVEAL$(\mathsf{P}, t)$
if $\pi_{\mathsf{P}}^t.\mathrm{acc} \neq \mathbf{true}$ or $\pi_{\mathsf{P}}^t.\mathrm{test} = \mathbf{true}$
    return $\bot$
if $\exists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$
and $\pi_{\mathsf{P}'}^{t'}.\mathrm{test} = \mathbf{true}$
    return $\bot$
$\forall (\mathsf{P}', t')$ s. t. $\pi_{\mathsf{P}'}^{t'}.\mathsf{tr} = \pi_{\mathsf{P}}^t.\mathsf{tr}$                          //G_2-G_4
    $\pi_{\mathsf{P}'}^{t'}.\mathsf{fr} := \mathbf{false}$                                //G_2-G_4
return $\pi_{\mathsf{P}}^t.K$

TEST$(\mathsf{P}, t)$)
if $\mathsf{Fresh}(\pi_{\mathsf{P}}^t) = \mathbf{false}$ return $\bot$                         //G_0-G_1
if $\pi_{\mathsf{P}}^t.\mathsf{fr} = \mathbf{false}$ return $\bot$                           //G_2-G_4
$K_0^* :=$ REVEAL$(\mathsf{P}, t)$
if $K_0^* = \bot$ return $\bot$
$K_1^* \xleftarrow{\$} \mathcal{K}$
$\pi_{\mathsf{P}}^t.\mathrm{test} := \mathbf{true}$
return $K_\beta^*$

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z)$
if $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z] = K \neq \bot$
    return $K$
$T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, Z] \xleftarrow{\$} \mathcal{K}$
return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z]$

SENDINIT$(\mathsf{U}, t, \mathsf{S})$
if $\pi_{\mathsf{U}}^t \neq \bot$ return $\bot$
$(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
$(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star x_{b_1}, ..., u_\ell \star x_{b_\ell})$
$\pi_{\mathsf{U}}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \bot), \bot, \bot)$
$\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{false}$                                           //G_2-G_4
return $(\mathsf{U}, x^{\mathsf{U}})$

SENDRESP$(\mathsf{S}, t, \mathsf{U}, x^{\mathsf{U}})$
if $\pi_{\mathsf{S}}^t \neq \bot$ return $\bot$
$(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
$(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star x_{b_1}, ..., s_\ell \star x_{b_\ell})$
if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$         //G_1-G_4
    $\mathbf{bad}_{\mathrm{coll}} := \mathbf{true}$                                   //G_1-G_4
    return $\bot$                                            //G_1-G_4
if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$                                         //G_2-G_4
    $\pi_{\mathsf{S}}^t.\mathsf{fr} := \mathbf{true}$                                     //G_2-G_4
else                                                         //G_2-G_4
    $\pi_{\mathsf{S}}^t.\mathsf{fr} := \mathbf{false}$                                    //G_2-G_4
$z := (z_1, ..., z_\ell) := (s_1 \star x_1^{\mathsf{U}}, ..., s_\ell \star x_\ell^{\mathsf{U}})$
$K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$
$\pi_{\mathsf{S}}^t := ((s_1, ..., s_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
return $(\mathsf{S}, x^{\mathsf{S}})$

SENDTERMINIT$(\mathsf{U}, t, \mathsf{S}, x^{\mathsf{S}})$
if $\pi_{\mathsf{U}}^t \neq ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \bot), \bot, \bot)$
    return $\bot$
if $\exists \mathsf{P} \in \mathcal{U}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$              //G_1-G_4
    $\mathbf{bad}_{\mathrm{coll}} := \mathbf{true}$                                   //G_1-G_4
    return $\bot$                                            //G_1-G_4
if $\exists t'$ s. t. $\pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
and $\pi_{\mathsf{S}}^{t'}.\mathsf{fr} = \mathbf{true}$                                       //G_2-G_4
    $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{true}$                                     //G_2-G_4
else if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$                                    //G_2-G_4
    $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{true}$                                     //G_2-G_4
else                                                         //G_2-G_4
    $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{false}$                                    //G_2-G_4
$z := (z_1, ..., z_\ell) := (u_1 \star x_1^{\mathsf{S}}, ..., u_\ell \star x_\ell^{\mathsf{S}})$
$K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$
$\pi_{\mathsf{U}}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
return true

CORRUPT$(\mathsf{U}, \mathsf{S})$
if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$ return $\bot$
for $\mathsf{P} \in \{\mathsf{U}, \mathsf{S}\}$
    if $\exists t$ s. t. $\pi_{\mathsf{P}}^t.\mathrm{test} = \mathbf{true}$
    and $\nexists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$
        return $\bot$
    $\forall \pi_{\mathsf{P}}^t$ : if $\nexists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$  //G_2-G_4
        $\pi_{\mathsf{P}}^t.\mathsf{fr} = \mathbf{false}$                                 //G_2-G_4
$\mathcal{C} := \mathcal{C} \cup \{(\mathsf{U}, \mathsf{S})\}$
return $\mathsf{pw}_{\mathsf{US}}$
```

Fig. 12. Games G_0-G_4 for the proof of Theorem 3. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$EXECUTE, SENDINIT, SENDRESP, SENDTERMINIT, REVEAL, CORRUPT, TEST, $\mathsf{H}\}$.

In EXECUTE queries, $\mathcal{B}_1$ chooses random group elements u_i and s_i and computes x^{U} using x and x^{S} using y independent of the password such that

$$x_i^{\mathsf{U}} = u_i \star x = (u_i \cdot g) \star \tilde{x} = (u_i \cdot g \cdot g_{b_i} \cdot g_{b_i}^{-1}) \star \tilde{x} = (u_i \cdot g \cdot g_{b_i}^{-1}) \star x_{b_i}$$

and analogously for server instances. Note that the value z_i is implicitly set to

$$z_i = u_i \cdot g \cdot s_i \cdot h \cdot g_{b_i}^{-1} \star \tilde{x}$$

Before choosing a random session key now, we check if there has been a query to the random oracle H with the correct z. This can be done using the decision oracle and the following equality:

$$\begin{aligned}\mathsf{GA\text{-}CDH}(x, x_i^{\mathsf{S}}) = (u_i^{-1} \cdot g_{b_i}) \star z_i \;&\Leftrightarrow\; \mathsf{GA\text{-}CDH}(x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = g_{b_i} \star z_i \\ &\Leftrightarrow\; \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = z_i \ .\end{aligned}$$

If one z_i is correct, $\mathcal{B}_1$ aborts and outputs the solution $(u_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_i = (g \cdot h) \star \tilde{x}$.

Otherwise, we store the values u_i and s_i in list T_{e} together with the trace and the password and choose a session key uniformly at random. List T_{e} is used to identify relevant queries to H. In particular, if the trace and password appear in a query, we retrieve the values u_i and s_i to check whether the provided z_i are correct as described above. If the oracle returns 1 for any i, $\mathcal{B}_1$ aborts and outputs $(u_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_i$.

```
$\mathcal{B}_1^{\mathrm{GA\text{-}DDH}(x,\cdot,\cdot)}(x, y)$
$(g_0, g_1) \xleftarrow{\$} \mathcal{G}^2$
$(x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x})$
$(\mathcal{C}, T, T_{\mathrm{e}}) := (\varnothing, \varnothing, \varnothing)$
$\beta \xleftarrow{\$} \{0, 1\}$
for $(\mathsf{U}, \mathsf{S}) \in \mathcal{U} \times \mathcal{S}$
    $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
$\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
Stop.

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z)$
if $\exists (u_1, ..., u_\ell, s_1, ..., s_\ell)$
s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, u_1, ..., u_\ell, s_1, ..., s_\ell) \in T_{\mathrm{e}}$
    $(b_1, ..., b_\ell) := \mathsf{pw}$
    for $i \in [\ell]$
        if $\mathrm{GA\text{-}DDH}(x, x_i^{\mathsf{S}}, (u_i^{-1} \cdot g_{b_i}) \star z_i) = 1$
            Stop with $(u_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_i$
if $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z] = K \neq \perp$
    return $K$
$T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z] \xleftarrow{\$} \mathcal{K}$
return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z]$
```

```
Execute$(\mathsf{U}, t_0, \mathsf{S}, t_1)$
if $\pi_{\mathsf{U}}^{t_0} \neq \perp$ or $\pi_{\mathsf{S}}^{t_1} \neq \perp$
    return $\perp$
$(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
$(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star x, ..., u_\ell \star x)$
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star y, ..., s_\ell \star y)$
if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
    return $\perp$
$\forall z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
    for $i \in [\ell]$
        if $\mathrm{GA\text{-}DDH}(x, x_i^{\mathsf{S}}, (u_i^{-1} \cdot g_{b_i}) \star z_i) = 1$
            Stop with $(u_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_i$
$T_{\mathrm{e}} := T_{\mathrm{e}} \cup \{\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, u_1, ..., u_\ell, s_1, ..., s_\ell\}$
$K \xleftarrow{\$} \mathcal{K}$
$\pi_{\mathsf{U}}^{t_0} := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
$\pi_{\mathsf{S}}^{t_1} := ((s_1, ..., s_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
$(\pi_{\mathsf{U}}^{t_0}.\mathsf{fr}, \pi_{\mathsf{S}}^{t_1}.\mathsf{fr}) := (\mathbf{true}, \mathbf{true})$
return $(\mathsf{U}, x^{\mathsf{U}}, \mathsf{S}, x^{\mathsf{S}})$
```

Fig. 13. Adversary $\mathcal{B}_1$ against $\mathsf{GA\text{-}StCDH}$ for the proof of Theorem 3. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$Execute, SendInit, SendResp, SendTermInit, Reveal, Corrupt, Test, $\mathsf{H}\}$. Oracles SendInit, SendResp, SendTermInit, Reveal, Corrupt and Test are defined as in G_2. Lines written in blue show how $\mathcal{B}_1$ simulates the game.

Game G_4. In game G_4, we remove the password from execute queries and use $\tilde{x}$ as the basis to compute x_i^{U} and x_i^{S}. This change is not observable by $\mathcal{A}$ and

$$\Pr[\mathsf{G}_4 \Rightarrow 1] = \Pr[\mathsf{G}_3 \Rightarrow 1] \ .$$

Game G_5. G_5 is given in Figure 14. In this game we want to replace the session keys by random for all fresh instances in oracles SendResp and SendTermInit. Therefore, we introduce an additional independent random oracle T_{s} which maps only the trace of an instance to a key. For all instances that are not fresh, we simply compute the correct key using random oracle H. If an instance is fresh and there is an inconsistency between T and T_{s}, we raise flag **bad**. This happens in the following cases:

– a fresh user or server instance is about to compute the session key, but there already exists a valid entry in T.
– the random oracle is queried on some trace of a fresh instance that appears in T_{s} together with the correct password and z.

Note that when **bad** is not raised, there is no difference between G_4 and G_5. Hence,

$$|\Pr[\mathsf{G}_5 \Rightarrow 1] - \Pr[\mathsf{G}_4 \Rightarrow 1]| \leq \Pr[\mathsf{G}_5 \Rightarrow \mathbf{bad}] \ .$$

```
GAME G_5
$(g_0, g_1) \xleftarrow{\$} \mathcal{G}^2$
$(x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x})$
$(\mathcal{C}, T, T_s) := (\emptyset, \emptyset, \emptyset)$
$\mathbf{bad} := \mathbf{false}$
$\beta \xleftarrow{\$} \{0,1\}$
for $(\mathsf{U}, \mathsf{S}) \in \mathcal{U} \times \mathcal{S}$
    $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
$\beta' \leftarrow \mathcal{A}^{\mathsf{O}}(x_0, x_1)$
return $[\![\beta = \beta']\!]$

EXECUTE$(\mathsf{U}, t_0, \mathsf{S}, t_1)$
if $\pi_{\mathsf{U}}^{t_0} \neq \perp$ or $\pi_{\mathsf{S}}^{t_1} \neq \perp$
    return $\perp$
$(u_1, \dots, u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$(s_1, \dots, s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, \dots, x_\ell^{\mathsf{U}}) := (u_1 \star \tilde{x}, \dots, u_\ell \star \tilde{x})$
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, \dots, x_\ell^{\mathsf{S}}) := (s_1 \star \tilde{x}, \dots, s_\ell \star \tilde{x})$
if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s.t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
    return $\perp$
$K \xleftarrow{\$} \mathcal{K}$
$\pi_{\mathsf{U}}^{t_0} := ((u_1, \dots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
$\pi_{\mathsf{S}}^{t_1} := ((s_1, \dots, s_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
$(\pi_{\mathsf{U}}^{t_0}.\mathsf{fr}, \pi_{\mathsf{S}}^{t_1}.\mathsf{fr}) := (\mathbf{true}, \mathbf{true})$
return $(\mathsf{U}, x^{\mathsf{U}}, \mathsf{S}, x^{\mathsf{S}})$

SENDINIT$(\mathsf{U}, t, \mathsf{S})$
if $\pi_{\mathsf{U}}^{t} \neq \perp$ return $\perp$
$(b_1, \dots, b_\ell) := \mathsf{pw}_{\mathsf{US}}$
$(u_1, \dots, u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, \dots, x_\ell^{\mathsf{U}}) := (u_1 \star x_{b_1}, \dots, u_\ell \star x_{b_\ell})$
$\pi_{\mathsf{U}}^{t} := ((u_1, \dots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \perp), \perp, \perp)$
$\pi_{\mathsf{U}}^{t}.\mathsf{fr} := \mathbf{false}$
return $(\mathsf{U}, x^{\mathsf{U}})$

H$(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z)$
if $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z] = K \neq \perp$
    return $K$
if $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}) \in T_s$ and $\mathsf{pw} = \mathsf{pw}_{\mathsf{US}}$
    if $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] = (\mathsf{U}, u_1, \dots, u_\ell, K)$
        $z' := (z'_1, \dots, z'_\ell) := (u_1 \star x_1^{\mathsf{S}}, \dots, u_\ell \star x_\ell^{\mathsf{S}})$
    if $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] = (\mathsf{S}, s_1, \dots, s_\ell, K)$
        $z' := (z'_1, \dots, z'_\ell) := (s_1 \star x_1^{\mathsf{U}}, \dots, s_\ell \star x_\ell^{\mathsf{U}})$
    if $z = z'$
        if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$
            return $K$
        if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
            $\mathbf{bad} := \mathbf{true}$
$T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z] \xleftarrow{\$} \mathcal{K}$
return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z]$

SENDRESP$(\mathsf{S}, t, \mathsf{U}, x^{\mathsf{U}})$
if $\pi_{\mathsf{S}}^{t} \neq \perp$ return $\perp$
$(b_1, \dots, b_\ell) := \mathsf{pw}_{\mathsf{US}}$
$(s_1, \dots, s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, \dots, x_\ell^{\mathsf{S}}) := (s_1 \star x_{b_1}, \dots, s_\ell \star x_{b_\ell})$
if $\exists P \in \mathcal{U} \cup \mathcal{S}, t'$ s.t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
    return $\perp$
if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
    $\pi_{\mathsf{S}}^{t}.\mathsf{fr} := \mathbf{true}$
    if $\exists z$ s.t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
    and $z_i = s_i \star x_i^{\mathsf{U}}\ \forall i \in [\ell]$
        $\mathbf{bad} := \mathbf{true}$
    $K \xleftarrow{\$} \mathcal{K}$
    $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] := (\mathsf{S}, (s_1, \dots, s_\ell), K)$
else
    $\pi_{\mathsf{S}}^{t}.\mathsf{fr} := \mathbf{false}$
    $z := (z_1, \dots, z_\ell) := (s_1 \star x_1^{\mathsf{U}}, \dots, s_\ell \star x_\ell^{\mathsf{U}})$
    $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$
$\pi_{\mathsf{S}}^{t} := ((s_1, \dots, s_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
return $(\mathsf{S}, x^{\mathsf{S}})$

SENDTERMINIT$(\mathsf{U}, t, \mathsf{S}, x^{\mathsf{S}})$
if $\pi_{\mathsf{U}}^{t} \neq ((u_1, \dots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \perp), \perp, \perp)$
    return $\perp$
if $\exists \mathsf{P} \in \mathcal{U}, t'$ s.t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
    return $\perp$
if $\exists t'$ s.t. $\pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
and $\pi_{\mathsf{S}}^{t'}.\mathsf{fr} = \mathbf{true}$
    $\pi_{\mathsf{U}}^{t}.\mathsf{fr} := \mathbf{true}$
    $(\mathsf{S}, (s_1, \dots, s_\ell), K) := T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}]$
else if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
    $\pi_{\mathsf{U}}^{t}.\mathsf{fr} := \mathbf{true}$
    if $\exists z$ s.t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
    and $z_i = u_i \star x_i^{\mathsf{S}}\ \forall i \in [\ell]$
        $\mathbf{bad} := \mathbf{true}$
    $K \xleftarrow{\$} \mathcal{K}$
    $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] := (\mathsf{U}, (u_1, \dots, u_\ell), K)$
else
    $\pi_{\mathsf{U}}^{t}.\mathsf{fr} := \mathbf{false}$
    $z := (z_1, \dots, z_\ell) := (u_1 \star x_1^{\mathsf{S}}, \dots, u_\ell \star x_\ell^{\mathsf{S}})$
    $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$
$\pi_{\mathsf{U}}^{t} := ((u_1, \dots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
return true
```

Fig. 14. Game G_5 for the proof of Theorem 3. $\mathcal{A}$ has access to oracles O := {EXECUTE, SENDINIT, SENDRESP, SENDTERMINIT, REVEAL, CORRUPT, TEST, H}. Oracles REVEAL, CORRUPT and TEST are defined as in Figure 12. Differences to game G_4 are highlighted in blue.

GAME G_6. G_6 is given in Figure 15. In this game we remove the password from send queries and generate passwords as late as possible. In SENDINIT and SENDRESP we then compute x^{U} and x^{S} using $\tilde{x}$ such that

$$x_i^{\mathsf{U}} = u_i \cdot \tilde{x} = (u_i \cdot g_0^{-1}) \star x_0 = (u_i \cdot g_1^{-1}) \star x_1$$

and equivalently for server instances. For all instances that are not fresh, we have to compute the real session key using $z_i = (s_i \cdot g_{b_i}^{-1}) \star x_i^{\mathsf{U}}$ or $z_i = (u_i \cdot g_{b_i}^{-1}) \star x_i^{\mathsf{S}}$. Now we split event **bad** into two different events:

– $\mathbf{bad}_{\mathsf{pw}}$ captures the event that there exists more than one valid entry in T for the same trace of a fresh instance, but different passwords.
– $\mathbf{bad}_{\text{guess}}$ happens only if $\mathbf{bad}_{\mathsf{pw}}$ does not happen and if there exists a valid entry in T for the trace of a fresh instance and the correct password.

To identify the different events, we introduce a new set T_{bad}. For all fresh instances in SendResp and SendTermInit, we now iterate over all entries in T that contain the corresponding trace. We check if the given password and z are valid for this trace by computing the real values z' in the same way as for non-fresh instances. If $z = z'$, we add this entry to the set T_{bad}. We essentially do the same when the random oracle H is queried on a trace that appears in T_{s}. Here, the adversary specifies the password and we check if z is valid for that password using the u_i stored in T_{s} for user instances and s_i for server instances. If z is valid and the instance is still fresh, we add the query to T_{bad}. In case the password was corrupted in the meantime, we output the key stored in T_{s} as introduced in the previous game.

After the adversary terminates, we check T_{bad} whether event $\mathbf{bad}_{\mathsf{pw}}$ or event $\mathbf{bad}_{\text{guess}}$ occurred. We will bound these events below. First note that whenever **bad** is raised in G_5, then either flag $\mathbf{bad}_{\text{guess}}$ or $\mathbf{bad}_{\mathsf{pw}}$ is raised in G_6, thus

$$\Pr[\mathsf{G}_5 \Rightarrow \mathbf{bad}] \leq \Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\mathsf{pw}}] + \Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\text{guess}}] \ .$$

To bound $\mathbf{bad}_{\mathsf{pw}}$, we construct adversary $\mathcal{B}_2$ against $\mathsf{Sim\text{-}GA\text{-}StCDH}$ in Figure 16. When $\mathbf{bad}_{\mathsf{pw}}$ occurs, then $\mathcal{B}_2$ can solve $\mathsf{Sim\text{-}GA\text{-}StCDH}$. Hence,

$$\Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\mathsf{pw}}] \leq \mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{Sim\text{-}GA\text{-}StCDH}}(\mathcal{B}_2) \ .$$

Adversary $\mathcal{B}_2$ inputs (x, x_0, x_1), where $x = g \star \tilde{x}$, $x_0 = g_0 \star \tilde{x}$ and $x_1 = g_1 \star \tilde{x}$ for uniformly random group elements $g, g_0, g_1 \in \mathcal{G}$. It also has access to decision oracles $\text{GA-DDH}_{x_0}(x, \cdot, \cdot)$ and $\text{GA-DDH}_{x_1}(x, \cdot, \cdot)$. It runs adversary $\mathcal{A}$ on (x_0, x_1). On a query to SendInit, $\mathcal{B}_2$ embeds x in x^{U} such that

$$x_i^{\mathsf{U}} = u_i \star x = (u_i \cdot g_0^{-1} \cdot g) \star x_0 = (u_i \cdot g_1^{-1} \cdot g) \star x_1 \ .$$

The simulation of x_i^{S} in SendResp is done in the same way. In case the server instance is fresh, we must check if there already exists an entry in T that causes an inconsistency, iterating over all pw, z, in T and using the decision oracles to check

$$z_i = \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{U}}, s_i \star x) \quad \Leftrightarrow \quad \mathsf{GA\text{-}CDH}_{x_{b_i}}(x, x_i^{\mathsf{U}}) = s_i^{-1} \star z_i \ ,$$

If all z_i are valid, then we add this entry to T_{bad}.

If the instance is not fresh, then we have to compute the correct key. We check list T for a valid entry z as explained above or choose a random key and add a special entry to T, which instead of z contains the secret group elements s_i so that we can patch the random oracle later. SendTermInit is simulated analogously.

Now we look at the random oracle queries. If the trace is contained in set T_{s}, we check if z is valid using the GA-DDH oracle. In case z is valid, we first check if the instance is still fresh and we add the query to T_{bad}. Otherwise, if the password was corrupted and is specified in the query, we return the session key stored in T_{s}. Next, we check if the query matches a special entry in T that was added in SendResp or SendTermInit for a non-fresh instance to keep the output consistent.

After $\mathcal{A}$ terminates with output β', $\mathcal{B}_2$ chooses the passwords which have not been generated yet. If $\mathbf{bad}_{\mathsf{pw}}$ occurred, then there must be two entries in T_{bad} for the same trace and different passwords pw and pw' along with values z and z'. As $\mathsf{pw} \neq \mathsf{pw}'$, we look for the first index i where the two passwords differ, i.e., $b_i \neq b_i'$. Recall that the $\mathsf{Sim\text{-}GA\text{-}StCDH}$ problem requires to compute $y_0 = \mathsf{GA\text{-}CDH}_{x_0}(x, y)$, $y_1 = \mathsf{GA\text{-}CDH}_{x_1}(x, y)$, where y can be chosen by the adversary. If the entries in T_{bad} belong to a user instance, $\mathcal{B}_2$ sets $y = x_i^{\mathsf{S}}$ and outputs y together with

$$y_0 = u_i^{-1} \star z_i = \mathsf{GA\text{-}CDH}_{x_0}(u_i^{-1} \star x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = \mathsf{GA\text{-}CDH}_{x_0}(x, x_i^{\mathsf{S}}) \ ,$$
$$y_1 = u_i^{-1} \star z_i' = \mathsf{GA\text{-}CDH}_{x_1}(u_i^{-1} \star x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = \mathsf{GA\text{-}CDH}_{x_1}(x, x_i^{\mathsf{S}}) \ .$$

```
GAME G_6
$(g_0, g_1) \xleftarrow{\$} \mathcal{G}^2$
$(x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x})$
$(\mathcal{C}, T, T_s, T_{\text{bad}}) := (\emptyset, \emptyset, \emptyset, \emptyset)$
$(\mathbf{bad}_{\text{guess}}, \mathbf{bad}_{\mathsf{pw}}, \mathbf{bad}_{\text{pfs}}) = (\mathbf{false}, \mathbf{false}, \mathbf{false})$
$\beta \xleftarrow{\$} \{0, 1\}$
$\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
for $(\mathsf{U}, \mathsf{S}) \in \mathcal{U} \times \mathcal{S} \setminus \mathcal{C}$
  $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
if $\exists \mathsf{pw}, \mathsf{pw}', (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, z, z')$
  s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z) \in T_{\text{bad}}$
  and $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}', z') \in T_{\text{bad}}$
  $\mathbf{bad}_{\mathsf{pw}} := \mathbf{true}$
else
  if $\exists \mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, z$
    s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T_{\text{bad}}$
    $\mathbf{bad}_{\text{guess}} := \mathbf{true}$
return $[\![\beta = \beta']\!]$

CORRUPT(U, S)
if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$ return $\bot$
for $\mathsf{P} \in \{\mathsf{U}, \mathsf{S}\}$
  if $\exists t$ s. t. $\pi^t_{\mathsf{P}}.\text{test} = \mathbf{true}$
  and $\nexists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\mathsf{Partner}(\pi^t_{\mathsf{P}}, \pi^{t'}_{\mathsf{P}'}) = 1$
    return $\bot$
  $\forall \pi^t_{\mathsf{P}}$ : if $\nexists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\mathsf{Partner}(\pi^t_{\mathsf{P}}, \pi^{t'}_{\mathsf{P}'}) = 1$
    $\pi^t_{\mathsf{P}}.\mathsf{fr} = \mathbf{false}$
$\mathcal{C} := \mathcal{C} \cup \{(\mathsf{U}, \mathsf{S})\}$
$\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
return $\mathsf{pw}_{\mathsf{US}}$

SENDINIT(U, t, S)
if $\pi^t_{\mathsf{U}} \neq \bot$ return $\bot$
$(u_1, \ldots, u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{U}} := (x^{\mathsf{U}}_1, \ldots, x^{\mathsf{U}}_\ell) := (u_1 \star \tilde{x}, \ldots, u_\ell \star \tilde{x})$
$\pi^t_{\mathsf{U}} := ((u_1, \ldots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \bot), \bot, \bot)$
$\pi^t_{\mathsf{U}}.\mathsf{fr} := \bot$
return $(\mathsf{U}, x^{\mathsf{U}})$

H(U, S, x^U, x^S, pw, z)
if $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z] = K \neq \bot$
  return $K$
if $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}) \in T_s$
  $(b_1, \ldots, b_\ell) := \mathsf{pw}$
  if $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] = (\mathsf{U}, u_1, \ldots, u_\ell, K)$
    $z' := ((u_1 \cdot g^{-1}_{b_1}) \star x^{\mathsf{S}}_1, \ldots, (u_\ell \cdot g^{-1}_{b_\ell}) \star x^{\mathsf{S}}_\ell)$
  if $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] = (\mathsf{S}, s_1, \ldots, s_\ell, K)$
    $z' := ((s_1 \cdot g^{-1}_{b_1}) \star x^{\mathsf{U}}_1, \ldots, (s_\ell \cdot g^{-1}_{b_\ell}) \star x^{\mathsf{U}}_\ell)$
  if $z = z'$
    if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$ and $\mathsf{pw} = \mathsf{pw}_{\mathsf{US}}$
      return $K$
    if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
      $T_{\text{bad}} := T_{\text{bad}} \cup \{\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z\}$
$T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z] \xleftarrow{\$} \mathcal{K}$
return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z]$

SENDRESP(S, t, U, x^U)
if $\pi^t_{\mathsf{S}} \neq \bot$ return $\bot$
$(s_1, \ldots, s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{S}} := (x^{\mathsf{S}}_1, \ldots, x^{\mathsf{S}}_\ell) := (s_1 \star \tilde{x}, \ldots, s_\ell \star \tilde{x})$
if $\exists P \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi^{t'}_{\mathsf{P}}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
  return $\bot$
if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
  $\pi^t_{\mathsf{S}}.\mathsf{fr} := \mathbf{true}$
  $\forall \mathsf{pw}, z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z) \in T$
    $(b_1, \ldots, b_\ell) := \mathsf{pw}$
    $z' := ((s_1 \cdot g^{-1}_{b_1}) \star x^{\mathsf{U}}_1, \ldots, (s_\ell \cdot g^{-1}_{b_\ell}) \star x^{\mathsf{U}}_\ell)$
    if $z = z'$
      $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z)\}$
  $K \xleftarrow{\$} \mathcal{K}$
  $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] := (\mathsf{S}, (s_1, \ldots, s_\ell), K)$
else
  $\pi^t_{\mathsf{S}}.\mathsf{fr} := \mathbf{false}$
  $(b_1, \ldots, b_\ell) := \mathsf{pw}_{\mathsf{US}}$
  $z := ((s_1 \cdot g^{-1}_{b_1}) \star x^{\mathsf{U}}_1, \ldots, (s_\ell \cdot g^{-1}_{b_\ell}) \star x^{\mathsf{U}}_\ell)$
  $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$
$\pi^t_{\mathsf{S}} := ((s_1, \ldots, s_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
return $(\mathsf{S}, x^{\mathsf{S}})$

SENDTERMINIT(U, t, S, x^S)
if $\pi^t_{\mathsf{U}} \neq ((u_1, \ldots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \bot), \bot, \bot)$
  return $\bot$
if $\exists \mathsf{P} \in \mathcal{U}, t'$ s. t. $\pi^{t'}_{\mathsf{P}}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
  return $\bot$
if $\exists t'$ s. t. $\pi^{t'}_{\mathsf{S}}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
and $\pi^{t'}_{\mathsf{S}}.\mathsf{fr} = \mathbf{true}$
  $\pi^t_{\mathsf{U}}.\mathsf{fr} := \mathbf{true}$
  $(\mathsf{S}, (s_1, \ldots, s_\ell), K) := T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}]$
else if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
  $\pi^t_{\mathsf{U}}.\mathsf{fr} := \mathbf{true}$
  $\forall \mathsf{pw}, z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z) \in T$
    $(b_1, \ldots, b_\ell) := \mathsf{pw}$
    $z' := ((u_1 \cdot g^{-1}_{b_1}) \star x^{\mathsf{S}}_1, \ldots, (u_\ell \cdot g^{-1}_{b_\ell}) \star x^{\mathsf{S}}_\ell)$
    if $z = z'$
      $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z)\}$
  $K \xleftarrow{\$} \mathcal{K}$
  $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] := (\mathsf{U}, (u_1, \ldots, u_\ell), K)$
else
  $\pi^t_{\mathsf{U}}.\mathsf{fr} := \mathbf{false}$
  $(b_1, \ldots, b_\ell) := \mathsf{pw}_{\mathsf{US}}$
  $z := ((u_1 \cdot g^{-1}_{b_1}) \star x^{\mathsf{S}}_1, \ldots, (u_\ell \cdot g^{-1}_{b_\ell}) \star x^{\mathsf{S}}_\ell)$
  $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z)$
$\pi^t_{\mathsf{U}} := ((u_1, \ldots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
return true
```

Fig. 15. Game G_6 for the proof of Theorem 3. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$EXECUTE, SENDINIT, SENDRESP, SENDTERMINIT, REVEAL, CORRUPT, TEST, H$\}$. Oracles REVEAL and TEST are defined as in game G_4 in Figure 12. Oracle EXECUTE is defined as in Figure 14. Differences to game G_5 are highlighted in blue.

If the instance is a server instance, $\mathcal{B}_2$ outputs $(y, y_0, y_1) = (x^{\mathsf{U}}_i, s^{-1}_i \star z_i, s^{-1}_i \star z'_i)$. This concludes the analysis of $\mathbf{bad}_{\mathsf{pw}}$.

Next, we analyze event $\mathbf{bad}_{\text{guess}}$. As $\mathbf{bad}_{\text{guess}}$ happens only if $\mathbf{bad}_{\mathsf{pw}}$ does not happen, there is at most one entry for each instance in T_{bad} and the size of T_{bad} is at most q_s. As all entries were added before

```
$\mathcal{B}_2^{\text{GA-DDH}_{x_0}(x,\cdot,\cdot),\text{GA-DDH}_{x_1}(x,\cdot,\cdot)}(x, x_0, x_1)$
$(\mathcal{C}, T, T_s) := (\varnothing, \varnothing, \varnothing)$
$\beta \xleftarrow{\$} \{0, 1\}$
$\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
for $(\mathsf{U}, \mathsf{S}) \in \mathcal{U} \times \mathcal{S} \setminus \mathcal{C}$
    $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
if $\exists \mathsf{pw}, \mathsf{pw}', (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, z, z')$
 s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z) \in T_{\text{bad}}$
 and $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}', z') \in T_{\text{bad}}$
    $(b_1, ..., b_\ell) := \mathsf{pw}$
    $(b'_1, ..., b'_\ell) := \mathsf{pw}'$
    Find first index $i$ such that $b_i \neq b'_i$
    W.l.o.g. let $b_i = 0,\ b'_i = 1$
    if $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] = (\mathsf{U}, (u_1, ..., u_\ell), K)$
        Stop with $(x_i^{\mathsf{S}}, u_i^{-1} \star z_i, u_i^{-1} \star z'_i)$
    if $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] = (\mathsf{S}, (s_1, ..., s_\ell), K)$
        Stop with $(x_i^{\mathsf{U}}, s_i^{-1} \star z_i, s_i^{-1} \star z'_i)$
```

```
SendInit$(\mathsf{U}, t, \mathsf{S})$
if $\pi_{\mathsf{U}}^t \neq \perp$ return $\perp$
$(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star x, ..., u_\ell \star x)$
$\pi_{\mathsf{U}}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \perp), \perp, \perp)$
return $(\mathsf{U}, x^{\mathsf{U}})$
```

```
$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z)$
if $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z] = K \neq \perp$
    return $K$
if $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}) \in T_s$
    $(b_1, ..., b_\ell) := \mathsf{pw}$
    if $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] = (\mathsf{U}, (u_1, ..., u_\ell), K)$
        if $\text{GA-DDH}_{x_{b_i}}(x, x_i^{\mathsf{S}}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
            if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
                $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z)\}$
            if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$ and $\mathsf{pw} = \mathsf{pw}_{\mathsf{US}}$
                return $K$
    if $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] = (\mathsf{S}, (s_1, ..., s_\ell), K)$
        if $\text{GA-DDH}_{x_{b_i}}(x, x_i^{\mathsf{U}}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
            if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
                $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z)\}$
            if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$ and $\mathsf{pw} = \mathsf{pw}_{\mathsf{US}}$
                return $K$
if $\exists (u_1, ..., u_\ell)$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, (u_1, ..., u_\ell)) \in T$
    $(b_1, ..., b_\ell) := \mathsf{pw}$
    if $\text{GA-DDH}_{x_{b_i}}(x, x_i^{\mathsf{S}}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
        return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, (u_1, ..., u_\ell)]$
else if $\exists (s_1, ..., s_\ell)$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, (s_1, ..., s_\ell)) \in T$
    $(b_1, ..., b_\ell) := \mathsf{pw}$
    if $\text{GA-DDH}_{x_{b_i}}(x, x_i^{\mathsf{U}}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
        return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, (s_1, ..., s_\ell)]$
$T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z] \xleftarrow{\$} \mathcal{K}$
return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z]$
```

```
SendResp$(\mathsf{S}, t, \mathsf{U}, x^{\mathsf{U}})$
if $\pi_{\mathsf{S}}^t \neq \perp$ return $\perp$
$(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star x, ..., s_\ell \star x)$
if $\exists P \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$ return $\perp$
if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
    $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ true
    $\forall \mathsf{pw}, z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z) \in T$
        $(b_1, ..., b_\ell) := \mathsf{pw}$
        if $\text{GA-DDH}_{x_{b_i}}(x, x_i^{\mathsf{U}}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
            $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z)\}$
    $K \xleftarrow{\$} \mathcal{K}$
    $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] := (S, (s_1, ..., s_\ell), K)$
else
    $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ false
    $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
    if $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
    and $\text{GA-DDH}_{x_{b_i}}(x, x_i^{\mathsf{U}}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
        $K := T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z]$
    else
        $K \xleftarrow{\$} \mathcal{K}$
        $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, (s_1, ..., s_\ell)] := K$
$\pi_{\mathsf{S}}^t := ((s_1, ..., {}_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
return $(\mathsf{S}, x^{\mathsf{S}})$
```

```
SendTermInit$(\mathsf{U}, t, \mathsf{S}, x^{\mathsf{S}})$
if $\pi_{\mathsf{U}}^t \neq ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \perp), \perp, \perp)$ return $\perp$
if $\exists \mathsf{P} \in \mathcal{U}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$ return $\perp$
if $\exists t'$ s. t. $\pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$ and $\pi_{\mathsf{S}}^{t'}.\mathsf{fr} =$ true
    $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ true
    $(\mathsf{S}, (s_1, ..., s_\ell), K) := T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}]$
else if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
    $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ true
    $\forall \mathsf{pw}, z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z) \in T$
        $(b_1, ..., b_\ell) := \mathsf{pw}$
        if $\text{GA-DDH}_{x_{b_i}}(x, x_i^{\mathsf{S}}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
            $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z)\}$
    $K \xleftarrow{\$} \mathcal{K}$
    $T_s[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] := (\mathsf{U}, (u_1, ..., u_\ell), K)$
else
    $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ false
    $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
    if $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
    and $\text{GA-DDH}_{x_{b_i}}(x, x_i^{\mathsf{S}}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
        $K := T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z]$
    else
        $K \xleftarrow{\$} \mathcal{K}$
        $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, (u_1, ..., u_\ell)] := K$
$\pi_{\mathsf{U}}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \mathbf{true})$
return true
```

Fig. 16. Adversary $\mathcal{B}_2$ against $\mathsf{Sim\text{-}GA\text{-}StCDH}$ for the proof of Theorem 3. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$Execute, SendInit, SendResp, SendTermInit, Reveal, Corrupt, Test, $\mathsf{H}\}$. Oracles Execute, Reveal, Corrupt and Test are defined as in G_6. Lines written in blue show how $\mathcal{B}_2$ simulates the game.

the corresponding password was sampled, we can bound the probability by

$$\Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\text{guess}}] \leq \frac{q_s}{|\mathcal{PW}|} \ .$$

Finally, if none of the bad events happens in G_6, all session keys output by Test are uniformly random and the adversary can only guess β. Hence, $\Pr[\mathsf{G}_6 \Rightarrow 1] = \frac{1}{2}$ and collecting the probabilities yields the bound in Theorem 3. □

D Security of $\mathsf{X\text{-}GA\text{-}PAKE}_{\ell,N}$ and $\mathsf{X\text{-}GA\text{-}PAKE}^{\mathsf{t}}_{\ell}$

Protocols $\mathsf{X\text{-}GA\text{-}PAKE}_{\ell,N}$ and $\mathsf{X\text{-}GA\text{-}PAKE}^{\mathsf{t}}_{\ell}$ are the two variants of $\mathsf{X\text{-}GA\text{-}PAKE}_{\ell}$ as introduced in Section 8. In Appendices D.1 and D.2, we now provide a security analysis for the two protocols.

D.1 $\mathsf{X\text{-}GA\text{-}PAKE}_{\ell,N}$

The main result of this part is the following theorem.

Theorem 4 (Security of $\mathsf{X\text{-}GA\text{-}PAKE}_{\ell,N}$). *For any adversary $\mathcal{A}$ against $\mathsf{X\text{-}GA\text{-}PAKE}_{\ell,N}$ that issues at most q_e execute queries and q_s send queries and where H is modeled as a random oracle, there exist adversary $\mathcal{B}_1$ against $\mathsf{GA\text{-}StCDH}$ and $\mathcal{B}_2$ against $\mathsf{SqInv\text{-}GA\text{-}StCDH}$ such that*

$$\mathsf{Adv}_{\mathsf{X\text{-}GA\text{-}PAKE}_{\ell,N}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{GA\text{-}StCDH}}_{\mathsf{EGA}}(\mathcal{B}_1) + 2 \cdot \mathsf{Adv}^{\mathsf{SqInv\text{-}GA\text{-}StCDH}}_{\mathsf{EGA}}(\mathcal{B}_2) + \frac{q_s}{|\mathcal{PW}|} + \frac{(q_s+q_e)^2}{|\mathcal{G}|^M} \ .$$

The proof of Theorem 4 is very similar to the proof of Theorem 1. Therefore we do not give a full proof for the security of $\mathsf{X\text{-}GA\text{-}PAKE}_{\ell,N}$, but shortly explain the difference between the two proofs.

The security assumptions underlying the proof of Theorem 1 need to be slightly adapted. In particular, the problem $\mathsf{DSim\text{-}GA\text{-}StCDH}$ is replaced by its variant $\mathsf{2^N DSim\text{-}GA\text{-}StCDH}$. However the $\mathsf{2^N DSim\text{-}GA\text{-}StCDH}$ problem can be reduced to $\mathsf{DSim\text{-}GA\text{-}StCDH}$ (Lemma 3).

Definition 13 ($\mathsf{2^N DSim\text{-}GA\text{-}StCDH}$). *On input $(x_0 = g_0 \star \tilde{x}, ..., x_{2^N-1} = g_{2^N-1} \star \tilde{x}, w_0 = h_0 \star \tilde{x}, w_1 = h_1 \star \tilde{x}) \in \mathcal{X}^{2^N+2}$, the $\mathsf{2^N DSim\text{-}GA\text{-}StCDH}$ problem requires to find $i \neq j \in [0, 2^N-1]$ and a tuple $(y, y_0, y_1, y_2, y_3) \in \mathcal{X}^5$ such that*

$$(y_0, y_1, y_2, y_3) = (g_i^{-1} \cdot h_0 \star y,\ g_i^{-1} \cdot h_1 \star y,\ g_j^{-1} \cdot h_0 \star y,\ g_j^{-1} \cdot h_1 \star y).$$

For a group action $\mathsf{XXX} \in \{\mathsf{EGA}, \mathsf{REGA}, \mathsf{EGAT}, \mathsf{REGAT}\}$, we define the advantage function of an adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{2^N DSim\text{-}GA\text{-}StCDH}}_{\mathsf{XXX}}(\mathcal{A}) := \Pr\left[\begin{array}{c|c} i \neq j \in [0, 2^N-1] & \\ y_0 = \mathsf{GA\text{-}CDH}_{x_i}(w_0, y) & (g_0, ..., g_{2^N-1}, h_0, h_1) \xleftarrow{\$} \mathcal{G}^{2^N+2} \\ y_1 = \mathsf{GA\text{-}CDH}_{x_i}(w_1, y) & (x_0, ..., x_{2^N-1}) = (g_0 \star \tilde{x}, ..., g_{2^N-1} \star \tilde{x}) \\ y_2 = \mathsf{GA\text{-}CDH}_{x_j}(w_0, y) & (w_0, w_1) = (h_0 \star \tilde{x}, h_1 \star \tilde{x}) \\ y_3 = \mathsf{GA\text{-}CDH}_{x_j}(w_1, y) & (i, j, y, y_0, y_1, y_2, y_3) \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, ..., x_{2^N-1}, w_0, w_1) \end{array}\right],$$

where $\mathrm{O} = \{\mathrm{GA\text{-}DDH}_{x_i}(w_j, \cdot, \cdot)\}_{i \in [0, 2^N-1], j \in \{0,1\}}$.

Lemma 3. *For any adversary $\mathcal{A}$ against $\mathsf{2^N DSim\text{-}GA\text{-}StCDH}$, there exists adversary $\mathcal{B}$ against $\mathsf{DSim\text{-}GA\text{-}StCDH}$ such that*

$$\mathsf{Adv}^{\mathsf{2^N DSim\text{-}GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{\mathsf{DSim\text{-}GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{B}) \ .$$

Proof. We construct adversary $\mathcal{B}$ as follows. On input $(x_0, x_1, w_0, w_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x}, h_0 \star \tilde{x}, h_1 \star \tilde{x})$, $\mathcal{B}$ chooses a random bit $b_i \xleftarrow{\$} \{0,1\}$, a random group element $\alpha_i \xleftarrow{\$} \mathcal{G}$ and computes $x'_i = \alpha_i \star x_{b_i}$ for each $i \in \{0, ..., 2^N-1\}$. Then it runs $\mathcal{A}$ on input $(x'_0, ..., x'_{2^N-1}, w_0, w_1)$. Finally, $\mathcal{A}$ outputs two indices (i, j) with $i \neq j$ and (y, y_0, y_1, y_2, y_3). If $b_i = b_j$ which happens with probability 1/2, then $\mathcal{B}$ aborts. Otherwise, assume that $b_i = 0$ and $b_j = 1$. Then $x'_i = (\alpha_i g_0) \star \tilde{x}$ and $x'_j = (\alpha_j g_1) \star \tilde{x}$, hence

$$y_0 = (\alpha_i^{-1} g_0^{-1} h_0) \star y, \qquad y_2 = (\alpha_j^{-1} g_1^{-1} h_0) \star y,$$
$$y_1 = (\alpha_i^{-1} g_0^{-1} h_1) \star y, \qquad y_3 = (\alpha_j^{-1} g_1^{-1} h_1) \star y.$$

It follows that $\mathcal{B}$ can compute the solution by simply multiplying by α_i and α_j respectively. If $b_j = 0$ and $b_i = 1$, the output of $\mathcal{B}$ must be swapped.

During the experiment, $\mathcal{A}$ also has access to decision oracles $\mathrm{GA\text{-}DDH}_{x'_i}(w_j, \cdot, \cdot)$ for $i \in \{0, ..., 2^N-1\}$ and $j \in \{0,1\}$. These can be easily simulated using $\mathcal{B}$'s decision oracles. On a query $\mathrm{GA\text{-}DDH}_{x'_i}(w_j, z_1, z_2)$, $\mathcal{B}$ queries its own oracle on $\mathrm{GA\text{-}DDH}_{x_{b_i}}(w_j, z_1, \alpha_i \star z_2)$ and forwards the output to $\mathcal{A}$. □

D.2 $\mathsf{X\text{-}GA\text{-}PAKE}^{\mathsf{t}}_{\ell}$

Here we discuss the security of $\mathsf{X\text{-}GA\text{-}PAKE}^{\mathsf{t}}_{\ell}$, the twisted version of $\mathsf{X\text{-}GA\text{-}PAKE}_{\ell}$, in more detail. For the most part, one can just replace x_1 by x_0^t everywhere in the proof of Theorem 1. The only significant difference occurs in the analysis of the event $\mathbf{bad}_{\mathsf{pw}}$. In particular this event does not allow to construct an adversary against DSim-GA-StCDH. Instead, we need to consider the following alteration of DSim-GA-StCDH for the security analysis.

Definition 14 (Twisted Double Simultaneous GA-StCDH (TDSim-GA-StCDH)). *On input $(x_0 = g_0 \star \tilde{x}, w_0 = h_0 \star \tilde{x}, w_1 = h_1 \star \tilde{x}) \in \mathcal{X}^3$, the TDSim-GA-StCDH requires to find a tuple $(y, y_0, y_1, y_2, y_3) \in \mathcal{X}^5$ such that*

$$(y_0, y_1, y_2, y_3) = (g_0^{-1} \cdot h_0 \star y,\ g_0^{-1} \cdot h_1 \star y,\ g_0 \cdot h_0 \star y,\ g_0 \cdot h_1 \star y).$$

For a group action XXX $\in$ {EGA, REGA, EGAT, REGAT}, we define the advantage function of an adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{TDSim\text{-}GA\text{-}StCDH}}_{\mathsf{XXX}}(\mathcal{A}) := \Pr\left[\begin{array}{l|c} y_0 = \mathsf{GA\text{-}CDH}_{x_0}(w_0, y) & (g_0, h_0, h_1) \xleftarrow{\$} \mathcal{G}^4 \\ y_1 = \mathsf{GA\text{-}CDH}_{x_0}(w_1, y) & x_0 = g_0 \star \tilde{x} \\ y_2 = \mathsf{GA\text{-}CDH}_{x_0^t}(w_0, y) & (w_0, w_1) = (h_0 \star \tilde{x}, h_1 \star \tilde{x}) \\ y_3 = \mathsf{GA\text{-}CDH}_{x_0^t}(w_1, y) & (y, y_0, y_1, y_2, y_3) \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, w_0, w_1) \end{array}\right],$$

where $\mathrm{O} = \{\text{GA-DDH}_{x_0}(w_j, \cdot, \cdot)\}_{j \in \{0,1\}}$.

The proof of Lemma 1 can be adapted to the TDSim-GA-StCDH problem in a straight forward way, showing that

$$\mathsf{Adv}^{\mathsf{TDSim\text{-}GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{SqInv\text{-}GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{B}) \ .$$

Consequently the statement of Theorem 1 continues to hold true for $\mathsf{X\text{-}GA\text{-}PAKE}^{\mathsf{t}}_{\ell}$.

E Security of $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell}$ and its Variants

The first part of this section is dedicated to the proof of Theorem 2. The second part discusses the different variants of $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell}$ and analyzes their security.

E.1 Proof of Theorem 2

For the convenience of the reader, we repeat the statement of the theorem.

Theorem 2 (Security of $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell}$). *For any adversary $\mathcal{A}$ against $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell}$ that issues at most q_e execute queries, q_s send queries and at most q_{G} and q_{H} queries to random oracles G and H, there exist an adversary $\mathcal{B}_1$ against GA-StCDH and an adversary $\mathcal{B}_2$ against GA-GapCDH such that*

$$\mathsf{Adv}_{\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell}}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{B}_1) + q_s\ell \cdot \sqrt{\mathsf{Adv}^{\mathsf{GA\text{-}GapCDH}}_{\mathsf{EGAT}}(\mathcal{B}_2)} + \frac{(q_s + q_e)^2}{|\mathcal{G}|^{\ell}} + \frac{q_{\mathsf{G}} q_s}{|\mathcal{G}|^{\ell}}$$
$$+ \frac{2 \cdot (q_{\mathsf{G}} + q_s + q_e)^2}{2^{\lambda}} + \frac{q_s}{|\mathcal{PW}|} \ ,$$

where λ is the output length of G in bits.

Before proving this theorem, we will introduce a new (interactive) computational assumption which is tailored to the protocol where the interactive part of the assumption reflects the commitment in that protocol. We will show that this assumption is implied by GA-GapCDH.

Definition 15 (Interactive Simultaneous GA-StCDH (ISim-GA-StCDH)). *On input $(x_0, x_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x}) \in \mathcal{X}^2$, the adversary first chooses and commits to some $y \in \mathcal{X}$. After receiving the challenge*

$x = g \star \tilde{x} \in \mathcal{X}$, the ISim-GA-StCDH *problem requires to compute $y_0 = gg_0^{-1} \star y$, $y_1 = gg_1^{-1} \star y$. For a group action* $\mathsf{XXX} \in \{\mathsf{EGA}, \mathsf{REGA}, \mathsf{EGAT}, \mathsf{REGAT}\}$*, we define the advantage function of an adversary $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathsf{XXX}}^{\mathsf{ISim\text{-}GA\text{-}StCDH}}(\mathcal{A}) := \Pr\left[\begin{array}{l|c} & (g_0, g_1) \xleftarrow{\$} \mathcal{G}^2 \\ & (x_0, x_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x}) \\ y_0 = \mathsf{GA\text{-}CDH}_{x_0}(x, y) & y \leftarrow \mathcal{A}^{\mathrm{O}_1}(x_0, x_1) \\ y_1 = \mathsf{GA\text{-}CDH}_{x_1}(x, y) & g \xleftarrow{\$} \mathcal{G} \\ & x = g \star \tilde{x} \\ & (y_0, y_1) \leftarrow \mathcal{A}^{\mathrm{O}_1, \mathrm{O}_2}(x) \end{array}\right] ,$$

where $\mathrm{O}_1 = \{\mathrm{GA\text{-}DDH}_{x_j}(\tilde{x}, \cdot, \cdot)\}_{j \in \{0,1\}}$ *and* $\mathrm{O}_2 = \{\mathrm{GA\text{-}DDH}_{x_j}(x, \cdot, \cdot)\}_{j \in \{0,1\}}$.

Note that ISim-GA-StCDH would be easy without the commitment if a group action allows to compute twists efficiently. In this case an adversary could simply choose $(y, y_0, y_1) = (x^t, x_0^t, x_1^t)$. The commitment prevents this trivial solution and intuitively, it thwarts the offline dictionary attack that was possible on $\mathsf{GA\text{-}PAKE}_\ell$ (Proposition 1).

Lemma 4. *The Group Action Gap Computational Diffie-Hellman Problem* (GA-GapCDH) *problem implies the Interactive Simultaneous* GA-StCDH (ISim-GA-StCDH) *for* EGAT*s, in particular*

$$\mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{ISim\text{-}GA\text{-}StCDH}}(\mathcal{A}) \le \sqrt{\mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{GA\text{-}GapCDH}}(\mathcal{B})} \ .$$

Proof. For the proof we use the reset lemma (see Lemma 2) with $H = \mathcal{X}$. Let $\mathcal{A}$ be an adversary against ISim-GA-StCDH. Consider adversary $\mathcal{B}$ against GA-GapCDH in Figure 17 that takes input (x_0, x_1) as well as x. It also has access to a gap oracle GA-DDH$_*$. First, $\mathcal{B}$ runs $\mathcal{A}$ on (x_0, x_1) to receive a commitment y. Now $\mathcal{B}$ sends x to $\mathcal{A}$ and $\mathcal{A}$ will finally output (y_0, y_1). $\mathcal{B}$ checks if the solution is correct using the decision oracle and if this is the case, it outputs $b = 1$ and $\sigma = (y, y_0, y_1)$. Otherwise it outputs $(0, \epsilon)$. As $\mathcal{B}$ has access to a full gap oracle, it can forward all queries of $\mathcal{A}$.

```
$\mathcal{B}^{\mathrm{GA\text{-}DDH}_*}(x_0, x_1, x)$
00 $y \leftarrow \mathcal{A}^{\mathrm{O}_1}(x_0, x_1)$
01 $(y_0, y_1) \leftarrow \mathcal{A}^{\mathrm{O}_1, \mathrm{O}_2}(x)$
02 if $\mathrm{GA\text{-}DDH}_{x_0}(x, y, y_0) = 1$
   and $\mathrm{GA\text{-}DDH}_{x_1}(x, y, y_1) = 1$
03     return $(1, (y, y_0, y_1))$
04 return $(0, \epsilon)$
```

```
$\mathcal{C}^{\mathrm{GA\text{-}DDH}_*}(x_0, x_1)$
05 Pick random coins $\rho$ for $\mathcal{B}$
06 $a \xleftarrow{\$} \mathcal{G}$; $x := a \star \tilde{x}$
07 $(b, \sigma) \leftarrow \mathcal{B}^{\mathrm{GA\text{-}DDH}_*}(x_0, x_1^t, x; \rho)$
08 if $b = 0$ return $\perp$
09 $(y, y_0, y_1) := \sigma$
10 $\alpha \xleftarrow{\$} \mathcal{G}$; $x' := \alpha \star y_0^t$
11 $(b', \sigma') \leftarrow \mathcal{B}^{\mathrm{GA\text{-}DDH}_*}(x_0, x_1^t, x'; \rho)$
12 if $b = 0$ return $\perp$
13 $(y, y_0', y_1') := \sigma$
14 return $\alpha^{-1} \cdot a \star y_1'$
```

Fig. 17. Adversaries $\mathcal{B}$ and $\mathcal{C}$ against GA-GapCDH for the proof of Lemma 4. Adversary $\mathcal{A}$ has access to decision oracles $\mathrm{O}_1 = \{\mathrm{GA\text{-}DDH}_{x_j}(\tilde{x}, \cdot, \cdot)\}_{j \in \{0,1\}}$ and $\mathrm{O}_2 = \{\mathrm{GA\text{-}DDH}_{x_j}(x, \cdot, \cdot)\}_{j \in \{0,1\}}$, which $\mathcal{B}$ simulates using the gap oracle GA-DDH$_*$.

Let IG be the algorithm that chooses $g_0, g_1 \xleftarrow{\$} \mathcal{G}$ and outputs $(x_0, x_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x})$. Let acc be defined as in Lemma 2, thus

$$\mathrm{acc} \ge \mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{ISim\text{-}GA\text{-}StCDH}}(\mathcal{A}) \ .$$

Let $\mathcal{R}_\mathcal{B}$ be the reset algorithm associated to $\mathcal{B}$ as in Lemma 2 with access to the same decision oracles as $\mathcal{B}$.

We construct an adversary $\mathcal{C}$ against GA-GapCDH (Figure 17), but instead of running the reset algorithm, $\mathcal{C}$ will simulate $\mathcal{R}_\mathcal{B}$ running $\mathcal{B}$ directly.

$\mathcal{C}$ inputs (x_0, x_1) and has access to a gap oracle. First, it chooses random coins ρ for $\mathcal{B}$. It also samples a random element from H by first picking $a \xleftarrow{\$} \mathcal{G}$ and then computing $x = a \star \tilde{x}$. Then it runs $\mathcal{B}$ on $(x_0, x_1^t, x; \rho)$. Note that we use the twist of x_1. $\mathcal{B}$ outputs a bit b and side output σ. If $\mathcal{B}$ was successful, i.e., $b = 1$, then $\mathcal{C}$ parses σ as (y, y_1, y_2). Otherwise it aborts. Now it runs $\mathcal{B}$ a second time, this time on input (x_0, x_1^t, x'), where $x' = \alpha \star y_0^t$ for some $\alpha \xleftarrow{\$} \mathcal{G}$, and the same random coins ρ. Note that x' is also uniformly distributed over $\mathcal{X}$. If $\mathcal{B}$ is successful again, it outputs $(1, (y, y_0', y_1'))$, where y is the same as before since we run $\mathcal{B}$ on the same random coins. Now $\mathcal{C}$ can solve $\mathsf{GA\text{-}GapCDH}$ as follows: Let $y = h \star \tilde{x}$ for some $h \in \mathcal{G}$. Then we have

$$\alpha \star y_0^t = (\alpha \cdot a^{-1} g_0 h^{-1}) \star \tilde{x},$$

hence $y_0' = (\alpha \cdot a^{-1}) \star \tilde{x}$, $y_1' = (\alpha \cdot a^{-1} \cdot g_0 \cdot g_1 \star \tilde{x})$. Note that h cancels out. Using the knowledge of a and α, $\mathcal{C}$ can compute $\alpha^{-1} \cdot a \star y_1' = \mathsf{GA\text{-}CDH}(x_0, x_1)$.

Note that even if $x = x'$, we can solve $\mathsf{GA\text{-}GapCDH}$, so there is no additional term in the bound. □

Note that for the above proof it is indeed necessary that $\mathcal{B}$ (and $\mathcal{C}$) has access to a full gap oracle. If $\mathcal{B}$ only had access to a restricted oracle, it would not be able to simulate the oracles $\text{GA-DDH}_{x_0}(\alpha \star y_0^t, \cdot, \cdot)$ and $\text{GA-DDH}_{x_1^t}(\alpha \star y_0^t, \cdot, \cdot)$ in the second part of the proof.

Now we give the full proof of Theorem 2.

Proof (of Theorem 2). Let $\mathcal{A}$ be an adversary against $\mathsf{Com\text{-}GA\text{-}PAKE}_\ell$. Consider the games in Figure 18.

Game G_0. This is the original game, hence

$$\mathsf{Adv}_{\mathsf{Com\text{-}GA\text{-}PAKE}_\ell}(\mathcal{A}) \leq |\Pr[\mathsf{G}_0 \Rightarrow 1] - 1/2| \ .$$

Game G_1. In game G_1, we raise flag $\mathbf{bad}_{\text{coll}}$ whenever a server instance computes the same trace as any other accepted instance (line 92) or a user instance computes the same trace as any other accepted user instance (line 108). In this case, SendTermInit or SendTermResp return $\perp$. We do the same if a trace that is computed in an Execute query collides with one of a previously accepted instance (line 20). Due to the difference lemma,

$$|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_0 \Rightarrow 1]| \leq \Pr[\mathbf{bad}_{\text{coll}}] \ .$$

Note that when $\mathbf{bad}_{\text{coll}}$ is not raised, each instance is unique and has at most one partner. In order to bound $\mathbf{bad}_{\text{coll}}$, recall that the trace of an oracle π_{P}^t consists of $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}} = (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}), x^{\mathsf{S}} = (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}), \mathsf{com})$, where the user message consists of x^{U} and the server messages consist of $(x^{\mathsf{S}}, \mathsf{com})$. When the game chooses x^{U}, then the probability that it collides with one particular other user instances is $|\mathcal{G}|^{-\ell}$ as all group elements must be the same. On the server side, the commitment is determined by the choice of x^{S} and when the game chooses x^{S} , then the probability that this instance collides with one particular other server instances is $|\mathcal{G}|^{-\ell}$ as well. As there are at most $q_s + q_e$ queries, this yields

$$\Pr[\mathbf{bad}_{\text{coll}}] \leq \binom{q_s + q_e}{2} \cdot \frac{1}{|\mathcal{G}|^\ell} \leq \frac{(q_s + q_e)^2}{|\mathcal{G}|^\ell} \ .$$

Game G_2. In G_2 we raise flag $\mathbf{bad}_{\text{bind}}$ if two different inputs to the random oracle G return the same commitment (line 53). This is to ensure that the adversary cannot open a commitment to a different value, which might depend on previous messages. The number of queries to G is bounded by $q_{\mathsf{G}} + q_s + q_e$, thus we have

$$|\Pr[\mathsf{G}_2 \Rightarrow 1] - \Pr[\mathsf{G}_1 \Rightarrow 1]| \leq \Pr[\mathbf{bad}_{\text{bind}}] \leq \frac{(q_{\mathsf{G}} + q_s + q_e)^2}{2^\lambda} \ .$$

Game G_3. In G_3 we choose the commitment uniformly at random when a session is initiated with a SendInit query (line 65). Then we can choose x^{S} only later in SendTermInit (line 86). However, we have to take into account some subtleties.

First, in the previous games we ensured that traces are unique and also the same commitment is not chosen twice. In this case G would have returned $\perp$ in G_2, so now we take care of this explicitly in SendInit. In particular, we also return $\perp$ whenever there already exists an entry in T_{G} that evaluates to the commitment chosen in SendInit (line 67). We add an entry to T_{G} with a placeholder $\diamond$ as input (line 68) to avoid that the commitment is chosen again. (Essentially, the checks in lines 52 and 66 also consider $x^{\mathsf{S}} = \diamond$.) Also we can now only save the commitment in the trace (line 69) and will adapt the initial check in SendTermInit (line 83).

Second, we have to take care of the case that the adversary has already issued a query to G on x^{S} before these values are chosen in SendTermInit because then we cannot assign com to this input. We

```
GAMES G_0-G_7
00 $(g_0, g_1) \xleftarrow{\$} \mathcal{G}^2$
01 $(x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x})$
02 $(\mathcal{C}, T_{\mathsf{H}}, T_{\mathsf{G}}) := (\varnothing, \varnothing)$
03 $(\mathbf{bad}_{\text{coll}}, \mathbf{bad}_{\text{bind}}, \mathbf{bad}_{\text{hide}}) := (\mathbf{false}, \mathbf{false}, \mathbf{false})$
04 $\beta \xleftarrow{\$} \{0,1\}$
05 for $(\mathsf{U}, \mathsf{S}) \in \mathcal{U} \times \mathcal{S}$
06    $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
07 $\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
08 return $[\![\beta = \beta']\!]$

EXECUTE$(\mathsf{U}, t_0, \mathsf{S}, t_1)$
09 if $\pi_{\mathsf{U}}^{t_0} \neq \perp$ or $\pi_{\mathsf{S}}^{t_1} \neq \perp$ return $\perp$
10 $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$                                   //$\mathsf{G}_0$-$\mathsf{G}_6$
11 $(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
12 $(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
13 $x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star x_{b_1}, ..., u_\ell \star x_{b_\ell})$        //$\mathsf{G}_0$-$\mathsf{G}_6$
14 $x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star x_{b_1}, ..., s_\ell \star x_{b_\ell})$        //$\mathsf{G}_0$-$\mathsf{G}_6$
15 $z := (z_1, ..., z_\ell) := (u_1 \star x_1^{\mathsf{S}}, ..., u_\ell \star x_\ell^{\mathsf{S}})$          //$\mathsf{G}_0$-$\mathsf{G}_6$
16 $x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star \tilde{x}, ..., u_\ell \star \tilde{x})$             //$\mathsf{G}_7$
17 $x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star \tilde{x}, ..., s_\ell \star \tilde{x})$             //$\mathsf{G}_7$
18 $\mathsf{com} := \mathsf{G}(x^{\mathsf{S}})$
19 if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$   //$\mathsf{G}_1$-$\mathsf{G}_7$
20    $\mathbf{bad}_{\text{coll}} := \mathbf{true}$; return $\perp$                            //$\mathsf{G}_1$-$\mathsf{G}_7$
21 $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z)$                 //$\mathsf{G}_0$-$\mathsf{G}_5$
22 $K \xleftarrow{\$} \mathcal{K}$                                                     //$\mathsf{G}_6$-$\mathsf{G}_7$
23 $\pi_{\mathsf{U}}^{t_0} := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), K, \mathbf{true})$
24 $\pi_{\mathsf{S}}^{t_1} := ((s_1, ..., s_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), K, \mathbf{true})$
25 $(\pi_{\mathsf{U}}^{t_0}.\mathsf{fr}, \pi_{\mathsf{S}}^{t_1}.\mathsf{fr}) := (\mathbf{true}, \mathbf{true})$                     //$\mathsf{G}_5$-$\mathsf{G}_7$
26 return $(\mathsf{U}, x^{\mathsf{U}}, \mathsf{S}, (\mathsf{com}, x^{\mathsf{S}}))$

CORRUPT$(\mathsf{U}, \mathsf{S})$
27 if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$ return $\perp$
28 for $\mathsf{P} \in \{\mathsf{U}, \mathsf{S}\}$
29    if $\exists t$ s. t. $\pi_{\mathsf{P}}^t.\text{test} = \mathbf{true}$
      and $\nexists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$
30       return $\perp$
31    $\forall \pi_{\mathsf{P}}^t$ : if $\nexists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$  //$\mathsf{G}_5$-$\mathsf{G}_7$
32       $\pi_{\mathsf{P}}^t.\mathsf{fr} = \mathbf{false}$                                    //$\mathsf{G}_5$-$\mathsf{G}_7$
33 $\mathcal{C} := \mathcal{C} \cup \{(\mathsf{U}, \mathsf{S})\}$
34 return $\mathsf{pw}_{\mathsf{US}}$

TEST$(\mathsf{P}, t)$)
35 if $\mathsf{Fresh}(\pi_{\mathsf{P}}^t) = \mathbf{false}$ return $\perp$                        //$\mathsf{G}_0$-$\mathsf{G}_4$
36 if $\pi_{\mathsf{P}}^t.\mathsf{fr} = \mathbf{false}$ return $\perp$                         //$\mathsf{G}_5$-$\mathsf{G}_7$
37 $K_0^* := \text{REVEAL}(\mathsf{P}, t)$
38 if $K_0^* = \perp$ return $\perp$
39 $K_1^* \xleftarrow{\$} \mathcal{K}$
40 $\pi_{\mathsf{P}}^t.\text{test} := \mathbf{true}$
41 return $K_\beta^*$

REVEAL$(\mathsf{P}, t)$
42 if $\pi_{\mathsf{P}}^t.\text{acc} \neq \mathbf{true}$ or $\pi_{\mathsf{P}}^t.\text{test} = \mathbf{true}$
43    return $\perp$
44 if $\exists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$
   and $\pi_{\mathsf{P}'}^{t'}.\text{test} = \mathbf{true}$
45    return $\perp$
46 $\forall(\mathsf{P}', t')$ s. t. $\pi_{\mathsf{P}'}^{t'}.\mathsf{tr} = \pi_{\mathsf{P}}^t.\mathsf{tr}$                       //$\mathsf{G}_5$-$\mathsf{G}_7$
47    $\pi_{\mathsf{P}'}^{t'}.\mathsf{fr} := \mathbf{false}$                                  //$\mathsf{G}_5$-$\mathsf{G}_7$
48 return $\pi_{\mathsf{P}}^t.K$

$\mathsf{G}(x^{\mathsf{S}})$
49 if $T_{\mathsf{G}}[x^{\mathsf{S}}] = \mathsf{com} \neq \perp$
50    return $\mathsf{com}$
51 $T_{\mathsf{G}}[x^{\mathsf{S}}] \xleftarrow{\$} \{0,1\}^\lambda$
52 if $\exists x^{\mathsf{S}'}$ s. t. $T_{\mathsf{G}}[x^{\mathsf{S}'}] = T_{\mathsf{G}}[x^{\mathsf{S}}]$                      //$\mathsf{G}_2$-$\mathsf{G}_7$
53    $\mathbf{bad}_{\text{bind}} := \mathbf{true}$; return $\perp$                            //$\mathsf{G}_2$-$\mathsf{G}_7$
54 return $T_{\mathsf{G}}[x^{\mathsf{S}}]$

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z)$
55 if $T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z] = K \neq \perp$
56    return $K$
57 $T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, Z] \xleftarrow{\$} \mathcal{K}$
58 return $T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z]$

SENDINIT$(\mathsf{S}, t, \mathsf{U})$
59 if $\pi_{\mathsf{S}}^t \neq \perp$ return $\perp$
60 $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
61 $(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$                                         //$\mathsf{G}_0$-$\mathsf{G}_2$
62 $x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star x_{b_1}, ..., s_\ell \star x_{b_\ell})$          //$\mathsf{G}_0$-$\mathsf{G}_2$
63 $\mathsf{com} := \mathsf{G}(x^{\mathsf{S}})$                                               //$\mathsf{G}_0$-$\mathsf{G}_2$
64 $\pi_{\mathsf{S}}^t := ((s_1, ..., s_\ell), (\mathsf{U}, \mathsf{S}, \perp, x^{\mathsf{S}}, \mathsf{com}), \perp, \perp)$            //$\mathsf{G}_0$-$\mathsf{G}_2$
65 $\mathsf{com} \xleftarrow{\$} \{0,1\}^\lambda$                                              //$\mathsf{G}_3$-$\mathsf{G}_7$
66 if $\exists x^{\mathsf{S}}$ s. t. $T_{\mathsf{G}}[x^{\mathsf{S}}] = \mathsf{com}$                                //$\mathsf{G}_3$-$\mathsf{G}_7$
67    return $\perp$                                                 //$\mathsf{G}_3$-$\mathsf{G}_7$
68 $T_{\mathsf{G}}[\diamond] := \mathsf{com}$                                               //$\mathsf{G}_3$-$\mathsf{G}_7$
69 $\pi_{\mathsf{S}}^t := (\perp, (\mathsf{U}, \mathsf{S}, \perp, \perp, \mathsf{com}), \perp, \perp)$                        //$\mathsf{G}_3$-$\mathsf{G}_7$
70 $\pi_{\mathsf{S}}^t.\mathsf{fr} := \mathbf{false}$                                          //$\mathsf{G}_5$-$\mathsf{G}_7$
71 return $(\mathsf{S}, \mathsf{com})$

SENDRESP$(\mathsf{U}, t, \mathsf{S}, \mathsf{com})$
72 if $\pi_{\mathsf{U}}^t \neq \perp$ return $\perp$
73 if $\nexists x^{\mathsf{S}}$ s. t. $T_{\mathsf{G}}[x^{\mathsf{S}}] = \mathsf{com}$                                //$\mathsf{G}_4$-$\mathsf{G}_7$
74    $\pi_{\mathsf{U}}^t.\text{acc} := \mathbf{false}$                                    //$\mathsf{G}_4$-$\mathsf{G}_7$
75 $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
76 $(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
77 $x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star x_{b_1}, ..., u_\ell \star x_{b_\ell})$
78 $\pi_{\mathsf{U}}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \perp, \mathsf{com}), \perp, \perp)$
79 $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{false}$                                          //$\mathsf{G}_5$-$\mathsf{G}_7$
80 return $(\mathsf{U}, x^{\mathsf{U}})$

SENDTERMINIT$(\mathsf{S}, t, \mathsf{U}, x^{\mathsf{U}})$
81 if $\pi_{\mathsf{S}}^t \neq ((s_1, ..., s_\ell), (\mathsf{U}, \mathsf{S}, \perp, x^{\mathsf{S}}, \mathsf{com}), \perp, \perp)$         //$\mathsf{G}_0$-$\mathsf{G}_2$
82    return $\perp$                                                 //$\mathsf{G}_0$-$\mathsf{G}_2$
83 if $\pi_{\mathsf{S}}^t \neq (\perp, (\mathsf{U}, \mathsf{S}, \perp, \perp, \mathsf{com}), \perp, \perp)$                     //$\mathsf{G}_3$-$\mathsf{G}_7$
84    return $\perp$                                                 //$\mathsf{G}_3$-$\mathsf{G}_7$
85 $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
86 $(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$                                         //$\mathsf{G}_3$-$\mathsf{G}_7$
87 $x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star x_{b_1}, ..., s_\ell \star x_{b_\ell})$          //$\mathsf{G}_3$-$\mathsf{G}_7$
88 if $T_{\mathsf{G}}[x^{\mathsf{S}}] \neq \perp$                                            //$\mathsf{G}_3$-$\mathsf{G}_7$
89    $\mathbf{bad}_{\text{hide}} := \mathbf{true}$; return $\perp$                        //$\mathsf{G}_3$-$\mathsf{G}_7$
90 Replace $\diamond$ in $T_{\mathsf{G}}[\diamond] := \mathsf{com}$ with $x^{\mathsf{S}}$                     //$\mathsf{G}_3$-$\mathsf{G}_7$
91 if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$  //$\mathsf{G}_1$-$\mathsf{G}_7$
92    $\mathbf{bad}_{\text{coll}} := \mathbf{true}$                                       //$\mathsf{G}_1$-$\mathsf{G}_7$
93    return $\perp$                                                 //$\mathsf{G}_1$-$\mathsf{G}_7$
94 if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$                                               //$\mathsf{G}_5$-$\mathsf{G}_7$
95    $\pi_{\mathsf{S}}^t.\mathsf{fr} := \mathbf{true}$                                        //$\mathsf{G}_5$-$\mathsf{G}_7$
96 else                                                           //$\mathsf{G}_5$-$\mathsf{G}_7$
97    $\pi_{\mathsf{S}}^t.\mathsf{fr} := \mathbf{false}$                                       //$\mathsf{G}_5$-$\mathsf{G}_7$
98 $z := (z_1, ..., z_\ell) := (s_1 \star x_1^{\mathsf{U}}, ..., s_\ell \star x_\ell^{\mathsf{U}})$
99 $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z)$
100 $\pi_{\mathsf{S}}^t := ((s_1, ..., s_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), K, \mathbf{true})$
101 return $(\mathsf{S}, x^{\mathsf{S}})$

SENDTERMRESP$(\mathsf{U}, t, \mathsf{S}, x^{\mathsf{S}})$
102 if $\pi_{\mathsf{U}}^t \neq ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \perp, \mathsf{com}), \perp, \perp)$
103    return $\perp$
104 if $\mathsf{G}(x^{\mathsf{S}}) \neq \mathsf{com}$
105    $\pi_{\mathsf{U}}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), \perp, \mathbf{false})$
106    return $\perp$
107 if $\exists \mathsf{P} \in \mathcal{U}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$          //$\mathsf{G}_1$-$\mathsf{G}_7$
108    $\mathbf{bad}_{\text{coll}} := \mathbf{true}$; return $\perp$                            //$\mathsf{G}_1$-$\mathsf{G}_7$
109 if $\exists t'$ s. t. $\pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
    and $\pi_{\mathsf{S}}^{t'}.\mathsf{fr} = \mathbf{true}$                                      //$\mathsf{G}_5$-$\mathsf{G}_7$
110    $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{true}$                                        //$\mathsf{G}_5$-$\mathsf{G}_7$
111 else if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$                                           //$\mathsf{G}_5$-$\mathsf{G}_7$
112    $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{true}$                                        //$\mathsf{G}_5$-$\mathsf{G}_7$
113 else                                                          //$\mathsf{G}_5$-$\mathsf{G}_7$
114    $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{false}$                                       //$\mathsf{G}_5$-$\mathsf{G}_7$
115 $z := (z_1, ..., z_\ell) := (u_1 \star x_1^{\mathsf{S}}, ..., u_\ell \star x_\ell^{\mathsf{S}})$
116 $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z)$
117 $\pi_{\mathsf{U}}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), K, \mathbf{true})$
118 return true
```

Fig. 18. Games G_0-G_7 for the proof of Theorem 2. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$EXECUTE, SENDINIT, SENDRESP, SENDTERMINIT, REVEAL, CORRUPT, TEST, $\mathsf{G}, \mathsf{H}\}$.

cover this in event $\mathbf{bad}_{\text{hide}}$ (line 89). If $\mathbf{bad}_{\text{hide}}$ does not happen, we can overwrite the entry with the placeholder in T_{G} with the correct input (line 90).

Note that $\mathcal{A}$ can only distinguish these changes if $\mathbf{bad}_{\text{hide}}$ occurs. The probability that $\mathbf{bad}_{\text{hide}}$ occurs for one particular instance can be bounded by $\frac{q_{\mathsf{G}}}{|\mathcal{G}|^\ell}$ since x^{S} is fresh. For at most q_s instances, this yields

$$|\Pr[\mathsf{G}_3 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1]| \leq \Pr[\mathbf{bad}_{\text{hide}}] \leq \frac{q_{\mathsf{G}} q_s}{|\mathcal{G}|^\ell} \ .$$

Game G_4. Whenever $\mathcal{A}$ sends a commitment com in G_4 which was not computed using G, i.e., there does not exist an entry x^{S} in T_{G} such that $T_{\mathsf{G}}[x^{\mathsf{S}}] = \mathsf{com}$, we expect that this instance will reject the key and we immediately set the acc flag to **false** (line 74). Although SendResp will still output the first message, as soon as SendTermResp is queried, the game will return $\perp$ as the initial check will fail (line 102).

This change is only observable if $\mathcal{A}$ finds a correct input to G after it has sent the commitment. Then the instance will accept the key in G_3, but not in G_4. As there are at most $q_{\mathsf{G}} + q_s + q_e$ queries to G and q_s instances, we can upper bound the difference by

$$|\Pr[\mathsf{G}_4 \Rightarrow 1] - \Pr[\mathsf{G}_3 \Rightarrow 1]| \leq \frac{(q_{\mathsf{G}} + q_s + q_e) q_s}{2^\lambda} \ .$$

Game G_5. In game G_5, we make the freshness explicit. To each oracle π_{P}^t, we assign an additional variable $\pi_{\mathsf{P}}^t.\mathsf{fr}$ which is updated during the game. In particular, all instances used in execute queries are marked as fresh (line 25).

A server instance is fresh if the password was not corrupted yet (line 95). Otherwise, it is not fresh (line 97). A user instance is fresh if it has a fresh partner instance (line 110) or the password was not corrupted yet (line 112). Otherwise, it is not fresh (line 114). If $\mathcal{A}$ issues a Corrupt query later, the freshness variable will also be updated (line 32). When the session key of an instance is revealed, this instance and its potential partner instance are marked as not fresh (line 47). On a query to test, the game then only checks the freshness variable (line 36).

These are only a conceptual changes, hence

$$\Pr[\mathsf{G}_5 \Rightarrow 1] = \Pr[\mathsf{G}_4 \Rightarrow 1] \ .$$

Game G_6. In game G_6, we choose random keys for all instances queried to Execute (line 22). We construct adversary $\mathcal{B}_1$ against $\mathsf{GA\text{-}StCDH}$ in Figure 19 and show that

$$|\Pr[\mathsf{G}_6 \Rightarrow 1] - \Pr[\mathsf{G}_5 \Rightarrow 1]| \leq \mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{GA\text{-}StCDH}}(\mathcal{B}_1) \ .$$

Adversary $\mathcal{B}_1$ inputs a $\mathsf{GA\text{-}StCDH}$ challenge $(x, y) = (g \star \tilde{x}, h \star \tilde{x})$ and has access to a decision oracle $\text{GA-DDH}(x, \cdot, \cdot)$. First, it generates the crs elements (x_0, x_1) as in game G_6 and then runs adversary $\mathcal{A}$. Queries to Execute are simulated as follows: For $i \in [\ell]$ it chooses random group elements u_i and s_i for user and server instances, but instead of using (x_0, x_1) to compute the set elements, $\mathcal{B}_1$ uses x for the user instance (line 22) and y for the server instance (line 23), independent of the password bits b_i. We can rewrite this as

$$x_i^{\mathsf{U}} = u_i \star x = (u_i \cdot g) \star \tilde{x} = (u_i \cdot g \cdot g_{b_i} \cdot g_{b_i}^{-1}) \star \tilde{x} = \underbrace{(u_i \cdot g \cdot g_{b_i}^{-1})}_{u_i'} \star x_{b_i} \ ,$$

where u_i' is the group element that the user actually needs to compute the session key. In the same way, $s_i' = s_i \cdot h \cdot g_{b_i}^{-1}$. Note that z_i is implicitly set to

$$z_i = (u_i' \cdot s_i') \star x_{b_i} = u_i \cdot g \cdot s_i \cdot h \cdot g_{b_i}^{-1} \star \tilde{x} \ . \qquad (2)$$

We want to choose a random session key now, but before that we check if there has been a query to the random oracle H that matches the session key (lines 27-30). We iterate over the entries in T_{H}, where U,

```
$\underline{\mathcal{B}_1^{\text{GA-DDH}(x,\cdot,\cdot)}(x,y)}$
00 $(g_0, g_1) \xleftarrow{\$} \mathcal{G}^2$
01 $(x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x})$
02 $(\mathcal{C}, T_{\mathsf{H}}, T_{\mathsf{G}}) := (\varnothing, \varnothing)$
03 $\beta \xleftarrow{\$} \{0,1\}$
04 for $(\mathsf{U},\mathsf{S}) \in \mathcal{U} \times \mathcal{S}$
05    $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
06 $\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
07 Stop.

$\underline{\mathsf{H}(\mathsf{U},\mathsf{S},x^{\mathsf{U}},x^{\mathsf{S}},\mathsf{com},\mathsf{pw},z)}$
08 if $\exists(u_1,...,u_\ell,s_1,...,s_\ell)$ s.t.
   $(\mathsf{U},\mathsf{S},x^{\mathsf{U}},x^{\mathsf{S}},\mathsf{com},\mathsf{pw},u_1,...,u_\ell,s_1,...,s_\ell) \in T_{\mathrm{e}}$
09    $(b_1,...,b_\ell) := \mathsf{pw}$
10    for $i \in [\ell]$
11       if $\text{GA-DDH}(x, x_i^{\mathsf{S}}, (u_i^{-1} \cdot g_{b_i}) \star z_i) = 1$
12          Stop with $(u_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_i$
13 if $T_{\mathsf{H}}[\mathsf{U},\mathsf{S},x^{\mathsf{U}},x^{\mathsf{S}},\mathsf{com},\mathsf{pw},z] = K \neq \perp$
14    return $K$
15 $T_{\mathsf{H}}[\mathsf{U},\mathsf{S},x^{\mathsf{U}},x^{\mathsf{S}},\mathsf{com},\mathsf{pw},Z] \xleftarrow{\$} \mathcal{K}$
16 return $T_{\mathsf{H}}[\mathsf{U},\mathsf{S},x^{\mathsf{U}},x^{\mathsf{S}},\mathsf{com},\mathsf{pw},z]$

$\underline{\textsc{Execute}(\mathsf{U},t_0,\mathsf{S},t_1)}$
17 if $\pi_{\mathsf{U}}^{t_0} \neq \perp$ or $\pi_{\mathsf{S}}^{t_1} \neq \perp$
18    return $\perp$
19 $(b_1,...,b_\ell) := \mathsf{pw}_{\mathsf{US}}$
20 $(u_1,...,u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
21 $(s_1,...,s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
22 $x^{\mathsf{U}} := (x_1^{\mathsf{U}},...,x_\ell^{\mathsf{U}}) := (u_1 \star x,...,u_\ell \star x)$
23 $x^{\mathsf{S}} := (x_1^{\mathsf{S}},...,x_\ell^{\mathsf{S}}) := (s_1 \star y,...,s_\ell \star y)$
24 $\mathsf{com} := \mathsf{G}(x^{\mathsf{S}})$
25 if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s.t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U},\mathsf{S},x^{\mathsf{U}},x^{\mathsf{S}},\mathsf{com})$
26    return $\perp$
27 $\forall z$ s.t. $(\mathsf{U},\mathsf{S},x^{\mathsf{U}},x^{\mathsf{S}},\mathsf{com},\mathsf{pw}_{\mathsf{US}},z) \in T_{\mathsf{H}}$
28    for $i \in [\ell]$
29       if $\text{GA-DDH}(x, x_i^{\mathsf{S}}, (u_i^{-1} \cdot g_{b_i}) \star z_i) = 1$
30          Stop with $(u_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_i$
31 $T_{\mathrm{e}} := T_{\mathrm{e}} \cup \{\mathsf{U},\mathsf{S},x^{\mathsf{U}},x^{\mathsf{S}},\mathsf{com},\mathsf{pw}_{\mathsf{US}},u_1,...,u_\ell,s_1,...,s_\ell\}$
32 $K \xleftarrow{\$} \mathcal{K}$
33 $\pi_{\mathsf{U}}^{t_0} := ((u_1,...,u_\ell),(\mathsf{U},\mathsf{S},x^{\mathsf{U}},x^{\mathsf{S}},\mathsf{com}),K,\mathbf{true})$
34 $\pi_{\mathsf{S}}^{t_1} := ((s_1,...,s_\ell),(\mathsf{U},\mathsf{S},x^{\mathsf{U}},x^{\mathsf{S}},\mathsf{com}),K,\mathbf{true})$
35 $(\pi_{\mathsf{U}}^{t_0}.\mathsf{fr}, \pi_{\mathsf{S}}^{t_1}.\mathsf{fr}) := (\mathbf{true},\mathbf{true})$
36 return $(\mathsf{U},x^{\mathsf{U}},\mathsf{S},(\mathsf{com},x^{\mathsf{S}}))$
```

Fig. 19. Adversary $\mathcal{B}_1$ against GA-StCDH for the proof of Theorem 2. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$Execute, SendInit, SendResp, SendTermInit, Reveal, Corrupt, Test, G, H$\}$. Oracles SendInit, SendResp, SendTermInit, Reveal, Corrupt, Test and G are defined as in G_5. Lines written in blue show how $\mathcal{B}_1$ simulates the game.

S, x^{U}, x^{S} and $\mathsf{pw}_{\mathsf{US}}$ match, and check if one of the entries in z is correct. More precisely, for all $i \in [\ell]$, we check whether

$$\begin{aligned}\text{GA-CDH}(x, x_i^{\mathsf{S}}) = (u_i^{-1} \cdot g_{b_i}) \star z_i \ &\Leftrightarrow\ \text{GA-CDH}(x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = g_{b_i} \star z_i \\ &\Leftrightarrow\ \text{GA-CDH}_{x_{b_i}}(x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = z_i\end{aligned}$$

using the decision oracle $\text{GA-DDH}(x,\cdot,\cdot)$.

If one z_i is correct, $\mathcal{B}_1$ aborts and outputs the solution $(u_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_i = (g \cdot h) \star \tilde{x}$ (cf. Equation (2)).

Otherwise, we store the values u_i and s_i in list T_{e} together with the trace and the password (line 31) and choose a session key uniformly at random. We need list T_{e} to identify relevant queries to H. In particular, if the trace and password appear in a query, we retrieve the values u_i and s_i to check whether the provided z_i are correct. We do this in the same way as described above using the decision oracle (lines 08-12). If the oracle returns 1 for any i, $\mathcal{B}_1$ aborts and outputs $(u_i^{-1} \cdot s_i^{-1} \cdot g_{b_i}) \star z_i$.

Game G_7. In game G_7, we remove the password from execute queries. In particular, we do not compute x^{U} and x^{S} to the basis x_{b_i}, but simply use $\tilde{x}$ (lines 16, 17). Note that the values have the same distribution as in the previous game. Also, the group elements u and s are not used to derive the key. Hence, this change is not observable by $\mathcal{A}$ and

$$\Pr[\mathsf{G}_7 \Rightarrow 1] = \Pr[\mathsf{G}_6 \Rightarrow 1] \ .$$

Game G_8. G_8 is given in Figure 20. In this game we want to replace the session keys by random for all fresh instances in oracles SendTermInit and SendTermResp (lines 74, 96). Therefore, we introduce an additional independent random oracle T_{s} which maps only the trace of an instance to a key (lines 75, 97). We keep partner instances consistent, i.e., in case the adversary queries SendTermResp for a user instance and there exists a fresh partner instance, then we look in T_{s} for the corresponding key and assign it to this instance as well (line 91). For all instances that are not fresh, we simply compute the correct key using random oracle H (lines 78-79, 100-101). If a session is fresh and there is an inconsistency between T_{H} and T_{s}, we raise flag **bad**. This happens in the following cases:

– a server instance is about to compute the session key, the password was not corrupted, but there already exists an entry in T_{H} with the correct password and z (lines 72-73).
– a user instance is about to compute the session key, there exists no partner instance and the password was not corrupted, but there already exists an entry in T_{H} with the correct password and z (lines 94-95).

```
GAMES $\mathsf{G}_8$
00 $(g_0, g_1) \xleftarrow{\$} \mathcal{G}^2$
01 $(x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x})$
02 $(\mathcal{C}, T_\mathsf{H}, T_\mathsf{G}) := (\varnothing, \varnothing)$
03 $\beta \xleftarrow{\$} \{0,1\}$
04 for $(\mathsf{U},\mathsf{S}) \in \mathcal{U} \times \mathcal{S}$
05     $\mathsf{pw}_\mathsf{US} \xleftarrow{\$} \mathcal{PW}$
06 $\beta' \leftarrow \mathcal{A}^\mathrm{O}(x_0, x_1)$
07 return $[\![\beta = \beta']\!]$

EXECUTE$(\mathsf{U}, t_0, \mathsf{S}, t_1)$
08 if $\pi_\mathsf{U}^{t_0} \neq \bot$ or $\pi_\mathsf{S}^{t_1} \neq \bot$
09     return $\bot$
10 $(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
11 $(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
12 $x^\mathsf{U} := (x_1^\mathsf{U}, ..., x_\ell^\mathsf{U}) := (u_1 \star \tilde{x}, ..., u_\ell \star \tilde{x})$
13 $x^\mathsf{S} := (x_1^\mathsf{S}, ..., x_\ell^\mathsf{S}) := (s_1 \star \tilde{x}, ..., s_\ell \star \tilde{x})$
14 $\mathsf{com} := \mathsf{G}(x^\mathsf{S})$
15 if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s.t. $\pi_\mathsf{P}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com})$
16     return $\bot$
17 $K \xleftarrow{\$} \mathcal{K}$
18 $\pi_\mathsf{U}^{t_0} := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}), K, \mathbf{true})$
19 $\pi_\mathsf{S}^{t_1} := ((s_1, ..., s_\ell), (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}), K, \mathbf{true})$
20 $(\pi_\mathsf{U}^{t_0}.\mathsf{fr}, \pi_\mathsf{S}^{t_1}.\mathsf{fr}) := (\mathbf{true}, \mathbf{true})$
21 return $(\mathsf{U}, x^\mathsf{U}, \mathsf{S}, (\mathsf{com}, x^\mathsf{S}))$

SENDINIT$(\mathsf{S}, t, \mathsf{U})$
22 if $\pi_\mathsf{S}^t \neq \bot$ return $\bot$
23 $(b_1, ..., b_\ell) := \mathsf{pw}_\mathsf{US}$
24 $\mathsf{com} \xleftarrow{\$} \{0,1\}^\lambda$
25 if $\exists x^\mathsf{S}$ s.t. $T_\mathsf{G}[x^\mathsf{S}] = \mathsf{com}$
26     return $\bot$
27 $T_\mathsf{G}[\diamond] := \mathsf{com}$
28 $\pi_\mathsf{S}^t := (\bot, (\mathsf{U}, \mathsf{S}, \bot, \bot, \mathsf{com}), \bot, \bot)$
29 $\pi_\mathsf{S}^t.\mathsf{fr} := \mathbf{false}$
30 return $(\mathsf{S}, \mathsf{com})$

$\mathsf{G}(x^\mathsf{S})$
31 if $T_\mathsf{G}[x^\mathsf{S}] = \mathsf{com} \neq \bot$
32     return com
33 $T_\mathsf{G}[x^\mathsf{S}] \xleftarrow{\$} \{0,1\}^\lambda$
34 if $\exists x^{\mathsf{S}'}$ s.t. $T_\mathsf{G}[x^{\mathsf{S}'}] = T_\mathsf{G}[x^\mathsf{S}]$
35     return $\bot$
36 return $T_\mathsf{G}[x^\mathsf{S}]$

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z)$
37 if $T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z] = K \neq \bot$
38     return $K$
39 if $(\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}) \in T_\mathsf{s}$ and $\mathsf{pw} = \mathsf{pw}_\mathsf{US}$
40     if $T_\mathsf{s}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}] = (\mathsf{U}, u_1, ..., u_\ell, K)$
41         $z' := (z_1', ..., z_\ell') := (u_1 \star x_1^\mathsf{S}, ..., u_\ell \star x_\ell^\mathsf{S})$
42     if $T_\mathsf{s}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}] = (\mathsf{S}, s_1, ..., s_\ell, K)$
43         $z' := (z_1', ..., z_\ell') := (s_1 \star x_1^\mathsf{U}, ..., s_\ell \star x_\ell^\mathsf{U})$
44     if $z = z'$
45         if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$
46             return $K$
47         if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
48             bad := true
49 $T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, Z] \xleftarrow{\$} \mathcal{K}$
50 return $T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z]$
```

```
SENDRESP$(\mathsf{U}, t, \mathsf{S}, \mathsf{com})$
51 if $\pi_\mathsf{U}^t \neq \bot$ return $\bot$
52 if $\nexists x^\mathsf{S}$ s.t. $T_\mathsf{G}[x^\mathsf{S}] = \mathsf{com}$
53     $\pi_\mathsf{U}^t.\mathsf{acc} := \mathbf{false}$
54 $(b_1, ..., b_\ell) := \mathsf{pw}_\mathsf{US}$
55 $(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
56 $x^\mathsf{U} := (x_1^\mathsf{U}, ..., x_\ell^\mathsf{U}) := (u_1 \star x_{b_1}, ..., u_\ell \star x_{b_\ell})$
57 $\pi_\mathsf{U}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, \bot, \mathsf{com}), \bot, \bot)$
58 $\pi_\mathsf{U}^t.\mathsf{fr} := \mathbf{false}$
59 return $(\mathsf{U}, x^\mathsf{U})$

SENDTERMINIT$(\mathsf{S}, t, \mathsf{U}, x^\mathsf{U})$
60 if $\pi_\mathsf{S}^t \neq (\bot, (\mathsf{U}, \mathsf{S}, \bot, \bot, \mathsf{com}), \bot, \bot)$
61     return $\bot$
62 $(b_1, ..., b_\ell) := \mathsf{pw}_\mathsf{US}$
63 $(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
64 $x^\mathsf{S} := (x_1^\mathsf{S}, ..., x_\ell^\mathsf{S}) := (s_1 \star x_{b_1}, ..., s_\ell \star x_{b_\ell})$
65 if $T_\mathsf{G}[x^\mathsf{S}] \neq \bot$
66     return $\bot$
67 Replace $\diamond$ in $T_\mathsf{G}[\diamond] := \mathsf{com}$ with $x^\mathsf{S}$
68 if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s.t. $\pi_\mathsf{P}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com})$
69     return $\bot$
70 if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
71     $\pi_\mathsf{S}^t.\mathsf{fr} := \mathbf{true}$
72     if $\exists z$ s.t. $(\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}_\mathsf{US}, z) \in T_\mathsf{H}$
       and $z_i = s_i \star x_i^\mathsf{U}$ $\forall i \in [\ell]$
73         bad := true
74     $K \xleftarrow{\$} \mathcal{K}$
75     $T_\mathsf{s}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}] := (\mathsf{S}, (s_1, ..., s_\ell), K)$
76 else
77     $\pi_\mathsf{S}^t.\mathsf{fr} := \mathbf{false}$
78     $z := (z_1, ..., z_\ell) := (s_1 \star x_1^\mathsf{U}, ..., s_\ell \star x_\ell^\mathsf{U})$
79     $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}_\mathsf{US}, z)$
80 $\pi_\mathsf{S}^t := ((s_1, ..., s_\ell), (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}), K, \mathbf{true})$
81 return $(\mathsf{S}, x^\mathsf{S})$

SENDTERMRESP$(\mathsf{U}, t, \mathsf{S}, x^\mathsf{S})$
82 if $\pi_\mathsf{U}^t \neq ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, \bot, \mathsf{com}), \bot, \bot)$
83     return $\bot$
84 if $\mathsf{G}(x^\mathsf{S}) \neq \mathsf{com}$
85     $\pi_\mathsf{U}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}), \bot, \mathbf{false})$
86     return $\bot$
87 if $\exists \mathsf{P} \in \mathcal{U}, t'$ s.t. $\pi_\mathsf{P}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com})$
88     return $\bot$
89 if $\exists t'$ s.t. $\pi_\mathsf{S}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com})$
   and $\pi_\mathsf{S}^{t'}.\mathsf{fr} = \mathbf{true}$
90     $\pi_\mathsf{U}^t.\mathsf{fr} := \mathbf{true}$
91     $(\mathsf{S}, (s_1, ..., s_\ell), K) := T_\mathsf{s}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}]$
92 else if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
93     $\pi_\mathsf{U}^t.\mathsf{fr} := \mathbf{true}$
94     if $\exists z$ s.t. $(\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}_\mathsf{US}, z) \in T_\mathsf{H}$
       and $z_i = u_i \star x_i^\mathsf{S}$ $\forall i \in [\ell]$
95         bad := true
96     $K \xleftarrow{\$} \mathcal{K}$
97     $T_\mathsf{s}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}] := (\mathsf{U}, (u_1, ..., u_\ell), K)$
98 else
99     $\pi_\mathsf{U}^t.\mathsf{fr} := \mathbf{false}$
100     $z := (z_1, ..., z_\ell) := (u_1 \star x_1^\mathsf{S}, ..., u_\ell \star x_\ell^\mathsf{S})$
101     $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}_\mathsf{US}, z)$
102 $\pi_\mathsf{U}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}), K, \mathbf{true})$
103 return true
```

Fig. 20. Game G_8 for the proof of Theorem 2. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$EXECUTE, SENDINIT, SENDRESP, SENDTERMINIT, REVEAL, CORRUPT, TEST, $\mathsf{G}, \mathsf{H}\}$. Oracles REVEAL, CORRUPT and TEST are defined as in Figure 18. Differences to game G_7 are highlighted in blue.

– the random oracle is queried on some trace that appears in T_s together with the correct password and z (lines 39-48). At this point, we also check if the password was corrupted in the meantime and if this is the case and the adversary issues the correct query, we simply output the key stored in T_s (line 46) as this instance cannot be tested. This case corresponds to perfect forward secrecy which we cover in Appendix F.3.

Note that when **bad** is not raised, there is no difference between G_7 and G_8. Hence,

$$|\Pr[\mathsf{G}_8 \Rightarrow 1] - \Pr[\mathsf{G}_7 \Rightarrow 1]| \leq \Pr[\mathsf{G}_8 \Rightarrow \mathbf{bad}] \ .$$

Game G_9. G_9 is given in Figure 21. In this game we remove the password from send queries and generate passwords as late as possible, that is either when the adversary issues a corrupt query (line 29) or after it has stopped with output β' (line 07). In SendResp and SendTermInit we still choose group elements u_i and s_i uniformly at random, but now compute x_i^{U} and x_i^{S} using the origin element (lines 56 and 63). Thus, depending on which password is chosen afterwards, we implicitly set

$$x_i^{\mathsf{U}} = u_i \cdot \tilde{x} = (u_i \cdot g_0^{-1}) \star x_0 = (u_i \cdot g_1^{-1}) \star x_1$$

and analogously for x_i^{S}. For all instances that are not fresh, we have to compute the real session key using $z_i = (s_i \cdot g_{b_i}^{-1}) \star x_i^{\mathsf{U}}$ (line 81) or $z_i = (u_i \cdot g_{b_i}^{-1}) \star x_i^{\mathsf{S}}$ (line 107). Note that the password is already defined for these instances.

Recall that event **bad** in game G_8 is raised whenever there is an inconsistency in the random oracle queries and the keys of fresh instances. In this game, we split event **bad** into two different events:

– $\mathbf{bad}_{\mathsf{pw}}$ captures the event that there exists more than one valid entry in T_{H} for the same trace of a fresh instance, but different passwords.
– $\mathbf{bad}_{\text{guess}}$ happens only if $\mathbf{bad}_{\mathsf{pw}}$ does not happen and captures the event that there exists a valid entry in T_{H} for the trace of a fresh instance and the correct password, where the password was not corrupted when the query to H was made.

To identify the different events, we introduce a new set T_{bad}. For all fresh instances in SendTermInit and SendTermResp, we now iterate over all entries in T_{H} that contain the corresponding trace. We check if the given password and z are valid for this trace by computing the real values z' in the same way as for non-fresh instances. If $z = z'$, we add this entry to the set T_{bad} (lines 71-75, 97-101). We essentially do the same when the random oracle H is queried on a trace that appears in T_s. Here, the adversary specifies the password and we check if z is valid for that password using the u_i stored in T_s for user instances and s_i for server instances. If z is valid and the instance is still fresh, we add the query to T_{bad} (lines 39-49). In case the password was corrupted in the meantime, we output the key stored in T_s as introduced in the previous game. After the adversary terminates, we check T_{bad} whether event $\mathbf{bad}_{\mathsf{pw}}$ (line 09) or event $\mathbf{bad}_{\text{guess}}$ (line 12) occurred. We will bound these events below. First note that whenever **bad** is raised in G_8, then either flag $\mathbf{bad}_{\text{guess}}$ or $\mathbf{bad}_{\mathsf{pw}}$ is raised in G_9, thus

$$\Pr[\mathsf{G}_8 \Rightarrow \mathbf{bad}] \leq \Pr[\mathsf{G}_9 \Rightarrow \mathbf{bad}_{\mathsf{pw}}] + \Pr[\mathsf{G}_9 \Rightarrow \mathbf{bad}_{\text{guess}}] \ .$$

Finally, we bound the probabilities of the two events. We start with $\mathbf{bad}_{\mathsf{pw}}$. In Figure 22, we construct adversary $\mathcal{B}_2$ against $\mathsf{ISim\text{-}GA\text{-}StCDH}$ that simulates G_9. We show that when $\mathbf{bad}_{\mathsf{pw}}$ occurs, then $\mathcal{B}_2$ can solve $\mathsf{ISim\text{-}GA\text{-}StCDH}$. In the proof we need to guess the instance and the password bit for which $\mathbf{bad}_{\mathsf{pw}}$ happens. Hence,

$$\Pr[\mathsf{G}_9 \Rightarrow \mathbf{bad}_{\mathsf{pw}}] \leq q_s \ell \cdot \mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{ISim\text{-}GA\text{-}StCDH}}(\mathcal{B}_2) \ .$$

Recall that in the $\mathsf{ISim\text{-}GA\text{-}StCDH}$ problem, $\mathcal{B}_2$ must commit on some $y \in \mathcal{X}$ to receive the challenge $x = g \star \tilde{x}$ for $g \xleftarrow{\$} \mathcal{G}$. Thus, adversary $\mathcal{B}_2$ will first guess a send query τ^* and a password bit i^* for which it will solve the problem. On a high level, the simulation of G_9 for adversary $\mathcal{A}$ works as follows: on the τ^*-th send query, where $\mathcal{A}$ sends x^{P}, $\mathcal{B}_2$ will output the i^*-th set element $x_{i^*}^{\mathsf{P}}$ as a commitment. It then embeds the challenge x in the i^*'s set element which will be output to $\mathcal{A}$. If in the end, $\mathcal{A}$ has issued two valid queries to H for that trace, where the passwords differ in the i^*'s bit, $\mathcal{B}_2$ can solve the $\mathsf{ISim\text{-}GA\text{-}StCDH}$ problem.

Let us now describe $\mathcal{B}_2$ in more detail. Adversary $\mathcal{B}_2$ inputs (x_0, x_1), where $x_0 = g_0 \star \tilde{x}$ and $x_1 = g_1 \star \tilde{x}$ for group elements $g_0, g_1 \in \mathcal{G}$ chosen uniformly at random. Adversary $\mathcal{B}_2$ also has access to decision

```
**GAMES** $\mathsf{G}_9$
$(g_0, g_1) \xleftarrow{\$} \mathcal{G}^2$
$(x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x})$
$(\mathcal{C}, T_{\mathsf{H}}, T_{\mathsf{G}}) := (\varnothing, \varnothing)$
$(\mathbf{bad}_{\text{guess}}, \mathbf{bad}_{\mathsf{pw}}) = (\mathbf{false}, \mathbf{false})$
$\beta \xleftarrow{\$} \{0,1\}$
$\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
$\mathbf{for}\ (\mathsf{U},\mathsf{S}) \in \mathcal{U} \times \mathcal{S} \setminus \mathcal{C}$
  $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
$\mathbf{if}\ \exists \mathsf{pw}, \mathsf{pw}', (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, z, z')$
  $\text{s.t. } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z) \in T_{\text{bad}}$
  $\mathbf{and}\ (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}', z') \in T_{\text{bad}}$
  $\mathbf{bad}_{\mathsf{pw}} := \mathbf{true}$
$\mathbf{else}$
  $\mathbf{if}\ \exists \mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, z$
    $\text{s.t. } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z) \in T_{\text{bad}}$
    $\mathbf{bad}_{\text{guess}} := \mathbf{true}$
$\mathbf{return}\ [\![\beta = \beta']\!]$

SendInit$(\mathsf{S}, t, \mathsf{U})$
$\mathbf{if}\ \pi_{\mathsf{S}}^t \neq \bot\ \mathbf{return}\ \bot$
$\mathsf{com} \xleftarrow{\$} \{0,1\}^{\lambda}$
$\mathbf{if}\ \exists x^{\mathsf{S}} \text{ s.t. } T_{\mathsf{G}}[x^{\mathsf{S}}] = \mathsf{com}$
  $\mathbf{return}\ \bot$
$T_{\mathsf{G}}[\diamond] := \mathsf{com}$
$\pi_{\mathsf{S}}^t := (\bot, (\mathsf{U}, \mathsf{S}, \bot, \bot, \mathsf{com}), \bot, \bot)$
$\pi_{\mathsf{S}}^t.\mathsf{fr} := \mathbf{false}$
$\mathbf{return}\ (\mathsf{S}, \mathsf{com})$

Corrupt$(\mathsf{U}, \mathsf{S})$
$\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{return}\ \bot$
$\mathbf{for}\ \mathsf{P} \in \{\mathsf{U}, \mathsf{S}\}$
  $\mathbf{if}\ \exists t \text{ s.t. } \pi_{\mathsf{P}}^t.\text{test} = \mathbf{true}$
  $\mathbf{and}\ \nexists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t' \text{ s.t. } \mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$
    $\mathbf{return}\ \bot$
  $\forall \pi_{\mathsf{P}}^t : \mathbf{if}\ \nexists \mathsf{P}' \in \mathcal{U} \cup \mathcal{S}, t' \text{ s.t. } \mathsf{Partner}(\pi_{\mathsf{P}}^t, \pi_{\mathsf{P}'}^{t'}) = 1$
    $\pi_{\mathsf{P}}^t.\mathsf{fr} = \mathbf{false}$
$\mathcal{C} := \mathcal{C} \cup \{(\mathsf{U}, \mathsf{S})\}$
$\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
$\mathbf{return}\ \mathsf{pw}_{\mathsf{US}}$

$\mathsf{G}(x^{\mathsf{S}})$
$\mathbf{if}\ T_{\mathsf{G}}[x^{\mathsf{S}}] = \mathsf{com} \neq \bot$
  $\mathbf{return}\ \mathsf{com}$
$T_{\mathsf{G}}[x^{\mathsf{S}}] \xleftarrow{\$} \{0,1\}^{\lambda}$
$\mathbf{if}\ \exists x^{\mathsf{S}'} \text{ s.t. } T_{\mathsf{G}}[x^{\mathsf{S}'}] = T_{\mathsf{G}}[x^{\mathsf{S}}]$
  $\mathbf{return}\ \bot$
$\mathbf{return}\ T_{\mathsf{G}}[x^{\mathsf{S}}]$

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z)$
$\mathbf{if}\ T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z] = K \neq \bot$
  $\mathbf{return}\ K$
$\mathbf{if}\ (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}) \in T_{\text{s}}$
  $(b_1, \ldots, b_\ell) := \mathsf{pw}$
  $\mathbf{if}\ T_{\text{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}] = (\mathsf{U}, u_1, \ldots, u_\ell, K)$
    $z' := (z'_1, \ldots, z'_\ell) := ((u_1 \cdot g_{b_1}^{-1}) \star x_1^{\mathsf{S}}, \ldots, (u_\ell \cdot g_{b_\ell}^{-1}) \star x_\ell^{\mathsf{S}})$
  $\mathbf{if}\ T_{\text{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}] = (\mathsf{S}, s_1, \ldots, s_\ell, K)$
    $z' := (z'_1, \ldots, z'_\ell) := ((s_1 \cdot g_{b_1}^{-1}) \star x_1^{\mathsf{U}}, \ldots, (s_\ell \cdot g_{b_\ell}^{-1}) \star x_\ell^{\mathsf{U}})$
  $\mathbf{if}\ z = z'$
    $\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{and}\ \mathsf{pw} = \mathsf{pw}_{\mathsf{US}}$
      $\mathbf{return}\ K$
    $\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
      $T_{\text{bad}} := T_{\text{bad}} \cup \{\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z\}$
$T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, Z] \xleftarrow{\$} \mathcal{K}$
$\mathbf{return}\ T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z]$

SendResp$(\mathsf{U}, t, \mathsf{S}, \mathsf{com})$
$\mathbf{if}\ \pi_{\mathsf{U}}^t \neq \bot\ \mathbf{return}\ \bot$
$\mathbf{if}\ \nexists x^{\mathsf{S}} \text{ s.t. } T_{\mathsf{G}}[x^{\mathsf{S}}] = \mathsf{com}$
  $\pi_{\mathsf{U}}^t.\text{acc} := \mathbf{false}$
$(u_1, \ldots, u_\ell) \xleftarrow{\$} \mathcal{G}^{\ell}$
$x^{\mathsf{U}} := (x_1^{\mathsf{U}}, \ldots, x_\ell^{\mathsf{U}}) := (u_1 \star \tilde{x}, \ldots, u_\ell \star \tilde{x})$
$\pi_{\mathsf{U}}^t := ((u_1, \ldots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \bot, \mathsf{com}), \bot, \bot)$
$\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{false}$
$\mathbf{return}\ (\mathsf{U}, x^{\mathsf{U}})$

SendTermInit$(\mathsf{S}, t, \mathsf{U}, x^{\mathsf{U}})$
$\mathbf{if}\ \pi_{\mathsf{S}}^t \neq (\bot, (\mathsf{U}, \mathsf{S}, \bot, \bot, \mathsf{com}), \bot, \bot)$
  $\mathbf{return}\ \bot$
$(s_1, \ldots, s_\ell) \xleftarrow{\$} \mathcal{G}^{\ell}$
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, \ldots, x_\ell^{\mathsf{S}}) := (s_1 \star \tilde{x}, \ldots, s_\ell \star \tilde{x})$
$\mathbf{if}\ T_{\mathsf{G}}[x^{\mathsf{S}}] \neq \bot$
  $\mathbf{return}\ \bot$
Replace $\diamond$ in $T_{\mathsf{G}}[\diamond] := \mathsf{com}$ with $x^{\mathsf{S}}$
$\mathbf{if}\ \exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t' \text{ s.t. } \pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
  $\mathbf{return}\ \bot$
$\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
  $\pi_{\mathsf{S}}^t.\mathsf{fr} := \mathbf{true}$
  $\forall \mathsf{pw}, z \text{ s.t. } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z) \in T_{\mathsf{H}}$
    $(b_1, \ldots, b_\ell) := \mathsf{pw}$
    $z' := ((s_1 \cdot g_{b_1}^{-1}) \star x_1^{\mathsf{U}}, \ldots, (s_\ell \cdot g_{b_\ell}^{-1}) \star x_\ell^{\mathsf{U}})$
    $\mathbf{if}\ z = z'$
      $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z)\}$
  $K \xleftarrow{\$} \mathcal{K}$
  $T_{\text{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}] := (\mathsf{S}, (s_1, \ldots, s_\ell), K)$
$\mathbf{else}$
  $\pi_{\mathsf{S}}^t.\mathsf{fr} := \mathbf{false}$
  $(b_1, \ldots, b_\ell) := \mathsf{pw}_{\mathsf{US}}$
  $z := (z_1, \ldots, z_\ell) := ((s_1 \cdot g_{b_1}^{-1}) \star x_1^{\mathsf{U}}, \ldots, (s_\ell \cdot g_{b_\ell}^{-1}) \star x_\ell^{\mathsf{U}})$
  $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z)$
$\pi_{\mathsf{S}}^t := ((s_1, \ldots, s_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), K, \mathbf{true})$
$\mathbf{return}\ (\mathsf{S}, x^{\mathsf{S}})$

SendTermResp$(\mathsf{U}, t, \mathsf{S}, x^{\mathsf{S}})$
$\mathbf{if}\ \pi_{\mathsf{U}}^t \neq ((u_1, \ldots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \bot, \mathsf{com}), \bot, \bot)$
  $\mathbf{return}\ \bot$
$\mathbf{if}\ \mathsf{G}(x^{\mathsf{S}}) \neq \mathsf{com}$
  $\pi_{\mathsf{U}}^t := ((u_1, \ldots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), \bot, \mathbf{false})$
  $\mathbf{return}\ \bot$
$\mathbf{if}\ \exists \mathsf{P} \in \mathcal{U}, t' \text{ s.t. } \pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
  $\mathbf{return}\ \bot$
$\mathbf{if}\ \exists t' \text{ s.t. } \pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
$\mathbf{and}\ \pi_{\mathsf{S}}^{t'}.\mathsf{fr} = \mathbf{true}$
  $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{true}$
  $(\mathsf{S}, (s_1, \ldots, s_\ell), K) := T_{\text{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}]$
$\mathbf{else\ if}\ (\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
  $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{true}$
  $\forall \mathsf{pw}, z \text{ s.t. } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z) \in T_{\mathsf{H}}$
    $(b_1, \ldots, b_\ell) := \mathsf{pw}$
    $z' := ((u_1 \cdot g_{b_1}^{-1}) \star x_1^{\mathsf{S}}, \ldots, (u_\ell \cdot g_{b_\ell}^{-1}) \star x_\ell^{\mathsf{S}})$
    $\mathbf{if}\ z = z'$
      $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z)\}$
  $K \xleftarrow{\$} \mathcal{K}$
  $T_{\text{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}] := (\mathsf{S}, (s_1, \ldots, s_\ell), K)$
$\mathbf{else}$
  $\pi_{\mathsf{U}}^t.\mathsf{fr} := \mathbf{false}$
  $(b_1, \ldots, b_\ell) := \mathsf{pw}_{\mathsf{US}}$
  $z := (z_1, \ldots, z_\ell) := ((u_1 \cdot g_{b_1}^{-1}) \star x_1^{\mathsf{S}}, \ldots, (u_\ell \cdot g_{b_\ell}^{-1}) \star x_\ell^{\mathsf{S}})$
  $K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z)$
$\pi_{\mathsf{U}}^t := ((u_1, \ldots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), K, \mathbf{true})$
$\mathbf{return\ true}$
```

Fig. 21. Game G_9 for the proof of Theorem 2. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$Execute, SendInit, SendResp, SendTermInit, Reveal, Corrupt, Test, $\mathsf{G}, \mathsf{H}\}$. Oracles Reveal and Test are defined as in game G_7 in Figure 18. Oracle Execute is defined as in Figure 20. Differences to game G_8 are highlighted in blue.

```
$\mathcal{B}_2^{\text{GA-DDH}_{x_0}(\tilde{x},\cdot,\cdot),\text{GA-DDH}_{x_1}(\tilde{x},\cdot,\cdot)}(x_0, x_1)$
00 $(\mathcal{C}, T_\mathsf{H}, T_\mathsf{G}) := (\varnothing, \varnothing)$
01 $\tau^* \xleftarrow{\$} [q_s]$
02 $i^* \xleftarrow{\$} [\ell]$
03 $\text{cnt} := 0$
04 $\beta \xleftarrow{\$} \{0, 1\}$
05 $\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
06 $\mathbf{for}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{U} \times \mathcal{S} \setminus \mathcal{C}$
07    $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
08 $\mathbf{if}\ \exists \mathsf{pw}, \mathsf{pw}', (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, z, z')$
    $\text{s.t.}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z) \in T_{\text{bad}}$
    $\mathbf{and}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}', z') \in T_{\text{bad}}$
09    $\mathbf{if}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}) = \mathsf{tr}^*$
10       $(b_1, ..., b_\ell) := \mathsf{pw}$
11       $(b'_1, ..., b'_\ell) := \mathsf{pw}'$
12       $\mathbf{if}\ b_{i^*} \neq b'_{i^*}$
13          W.l.o.g. $b_{i^*} = 0,\ b'_{i^*} = 1$
14          Stop with $(z_{i^*}, z'_{i^*})$

$\text{SendInit}(\mathsf{S}, t, \mathsf{U})$
15 $\text{cnt} := \text{cnt} + 1$
16 $\mathbf{if}\ \pi_\mathsf{S}^t \neq \perp\ \mathbf{return}\ \perp$
17 $\mathsf{com} \xleftarrow{\$} \{0, 1\}^\lambda$
18 $\mathbf{if}\ \exists x^\mathsf{S}\ \text{s.t.}\ T_\mathsf{G}[x^\mathsf{S}] = \mathsf{com}$
19    $\mathbf{return}\ \perp$
20 $T_\mathsf{G}[\diamond] := \mathsf{com}$
21 $\pi_\mathsf{S}^t := (\perp, (\mathsf{U}, \mathsf{S}, \perp, \perp, \mathsf{com}), \perp, \perp)$
22 $\pi_\mathsf{S}^t.\mathsf{fr} := \mathbf{false}$
23 $\mathbf{return}\ (\mathsf{S}, \mathsf{com})$

$\text{SendResp}(\mathsf{U}, t, \mathsf{S}, \mathsf{com})$
24 $\text{cnt} := \text{cnt} + 1$
25 $\mathbf{if}\ \pi_\mathsf{U}^t \neq \perp\ \mathbf{return}\ \perp$
26 $\mathbf{if}\ \nexists x^\mathsf{S}\ \text{s.t.}\ T_\mathsf{G}[x^\mathsf{S}] = \mathsf{com}$
27    $\pi_\mathsf{U}^t.\mathsf{acc} := \mathbf{false}$
28 $(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
29 $x^\mathsf{U} := (x_1^\mathsf{U}, ..., x_\ell^\mathsf{U}) := (u_1 \star \tilde{x}, ..., u_\ell \star \tilde{x})$
30 $\mathbf{if}\ \text{cnt} = \tau^*\ \mathbf{and}\ \pi_\mathsf{U}^t.\mathsf{acc} \neq \mathbf{false}$
31    find $x^\mathsf{S}$ s.t. $T_\mathsf{G}[x^\mathsf{S}] = \mathsf{com}$
32    $(x_1^\mathsf{S}, ..., x_\ell^\mathsf{S}) := x^\mathsf{S}$
33    Output $y := x_{i^*}^\mathsf{S}$ to receive challenge $x$
34    // From now on $\mathcal{B}_2$ also has access to
         $\text{GA-DDH}_{x_0}(x, \cdot, \cdot)$, $\text{GA-DDH}_{x_1}(x, \cdot, \cdot)$
35    $x_{i^*}^\mathsf{U} := x$
36    $u_{i^*} := \perp$
37    $\mathsf{tr}^* = (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, \perp, \mathsf{com})$
38 $\pi_\mathsf{U}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, \perp, \mathsf{com}), \perp, \perp)$
39 $\pi_\mathsf{U}^t.\mathsf{fr} := \mathbf{false}$
40 $\mathbf{return}\ (\mathsf{U}, x^\mathsf{U})$

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z)$
41 $\mathbf{if}\ T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z] = K \neq \perp$
42    $\mathbf{return}\ K$
43 $(b_1, ..., b_\ell) := \mathsf{pw}$
44 $\mathbf{if}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}) \in T_\text{s}$
45    $\mathbf{if}\ T_\text{s}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}] = (\mathsf{U}, (u_1, ..., u_\ell), K)$
46       $\mathbf{if}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}) = \mathsf{tr}^*$
47          $\mathbf{if}\ \text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^\mathsf{S}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell] \setminus \{i^*\}$
            $\mathbf{and}\ \text{GA-DDH}_{x_{b_{i^*}}}(x, x_{i^*}^\mathsf{S}, z_{i^*}) = 1$
48             $\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
49                $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z)\}$
50             $\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{and}\ \mathsf{pw} = \mathsf{pw}_{\mathsf{US}}$
51                $\mathbf{return}\ K$
52       $\mathbf{else}$
53          $\mathbf{if}\ \text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^\mathsf{S}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
54             $\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
55                $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z)\}$
56             $\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{and}\ \mathsf{pw} = \mathsf{pw}_{\mathsf{US}}$
57                $\mathbf{return}\ K$
58    $\mathbf{if}\ T_\text{s}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}] = (\mathsf{S}, (s_1, ..., s_\ell), K)$
59       $\mathbf{if}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}) = \mathsf{tr}^*$
60          $\mathbf{if}\ \text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^\mathsf{U}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell] \setminus \{i^*\}$
            $\mathbf{and}\ \text{GA-DDH}_{x_{b_{i^*}}}(x, x_{i^*}^\mathsf{U}, z_{i^*}) = 1$
61             $\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
62                $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z)\}$
63             $\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{and}\ \mathsf{pw} = \mathsf{pw}_{\mathsf{US}}$
64                $\mathbf{return}\ K$
65       $\mathbf{else}$
66          $\mathbf{if}\ \text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^\mathsf{U}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
67             $\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
68                $T_{\text{bad}} := T_{\text{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z)\}$
69             $\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{and}\ \mathsf{pw} = \mathsf{pw}_{\mathsf{US}}$
70                $\mathbf{return}\ K$
71 $\mathbf{if}\ \exists (u_1, ..., u_\ell)\ \text{s.t.}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, (u_1, ..., u_\ell)) \in T_\mathsf{H}$
72    $\mathbf{if}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}) = \mathsf{tr}^*$
73       $\mathbf{if}\ \text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^\mathsf{S}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell] \setminus \{i^*\}$
         $\mathbf{and}\ \text{GA-DDH}_{x_{b_{i^*}}}(x, x_{i^*}^\mathsf{S}, z_{i^*}) = 1$
74          $\mathbf{return}\ T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, (u_1, ..., u_\ell)]$
75    $\mathbf{else}$
76       $\mathbf{if}\ \text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^\mathsf{S}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
77          $\mathbf{return}\ T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, (u_1, ..., u_\ell)]$
78 $\mathbf{else\ if}\ \exists (s_1, ..., s_\ell)\ \text{s.t.}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, (s_1, ..., s_\ell)) \in T_\mathsf{H}$
79    $\mathbf{if}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}) = \mathsf{tr}^*$
80       $\mathbf{if}\ \text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^\mathsf{U}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell] \setminus \{i^*\}$
         $\mathbf{and}\ \text{GA-DDH}_{x_{b_{i^*}}}(x, x_{i^*}^\mathsf{U}, z_{i^*}) = 1$
81          $\mathbf{return}\ T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, (s_1, ..., s_\ell)]$
82    $\mathbf{else}$
83       $\mathbf{if}\ \text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^\mathsf{U}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
84          $\mathbf{return}\ T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, (s_1, ..., s_\ell)]$
85 $T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, Z] \xleftarrow{\$} \mathcal{K}$
86 $\mathbf{return}\ T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z]$
```

Fig. 22. Adversary $\mathcal{B}_2$ against $\mathsf{ISim\text{-}GA\text{-}StCDH}$ for the proof of Theorem 2. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$Execute, SendInit, SendResp, SendTermInit, Reveal, Corrupt, Test, $\mathsf{G}, \mathsf{H}\}$. Oracles Execute, Reveal, Corrupt, Test and G are defined as in G_9. Oracles SendTermInit and SendTermResp are defined in Figure 23. Lines written in blue show how $\mathcal{B}_2$ simulates the game.

oracles $\text{GA-DDH}_{x_0}(\tilde{x}, \cdot, \cdot)$, $\text{GA-DDH}_{x_1}(\tilde{x}, \cdot, \cdot)$. It guesses a send query $\tau^* \xleftarrow{\$} [q_s]$ and a password bit $i^* \xleftarrow{\$} [\ell]$ and then initializes a counter cnt (lines 01-03), before it runs $\mathcal{A}$ on (x_0, x_1).

Apart from increasing the counter, queries to SendInit are simulated exactly as in G_9 as we do not choose any set elements here, but later in SendTermInit.

In SendResp, we also increase the counter and then we choose x^U as in G_9. Only if this is the τ^*-th query and the commitment sent by $\mathcal{A}$ was output by G before, then $\mathcal{B}_2$ looks in the list T_G to find the corresponding input x^S and outputs $x_{i^*}^\mathsf{S}$ as commitment y to receive the $\mathsf{ISim\text{-}GA\text{-}StCDH}$ challenge $x = g \star \tilde{x}$ (lines 30-33). It replaces the i^*-th element in x^U with x, implicitly defining $u_{i^*} = g$. In order to

```
SendTermInit$(\mathsf{S}, t, \mathsf{U}, x^{\mathsf{U}})$
00 $\mathrm{cnt} := \mathrm{cnt} + 1$
01 if $\pi_{\mathsf{S}}^t \neq (\bot, (\mathsf{U}, \mathsf{S}, \bot, \bot, \mathsf{com}), \bot, \bot)$
02     return $\bot$
03 $(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
04 $x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star \tilde{x}, ..., s_\ell \star \tilde{x})$
05 if $\mathrm{cnt} = \tau^*$
06     $(x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := x^{\mathsf{U}}$
07     Output $y := x_{i^*}^{\mathsf{U}}$ to receive challenge $x$
08     // From now on $\mathcal{B}_2$ also has access to
         $\mathrm{GA\text{-}DDH}_{x_0}(x, \cdot, \cdot)$, $\mathrm{GA\text{-}DDH}_{x_1}(x, \cdot, \cdot)$
09     $x_{i^*}^{\mathsf{S}} := x$
10     $s_{i^*} := \bot$
11     $\mathsf{tr}^* = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
12 if $T_{\mathsf{G}}[x^{\mathsf{S}}] \neq \bot$
13     return $\bot$
14 Replace $\diamond$ in $T_{\mathsf{G}}[\diamond] := \mathsf{com}$ with $x^{\mathsf{S}}$
15 if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
16     return $\bot$
17 if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
18     $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ true
19     $\forall \mathsf{pw}, z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z) \in T_{\mathsf{H}}$
20         $(b_1, ..., b_\ell) := \mathsf{pw}$
21         if $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}) = \mathsf{tr}^*$
22             if $\mathrm{GA\text{-}DDH}_{x_{b_i}}(\tilde{x}, x_i^{\mathsf{U}}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell] \setminus \{i^*\}$
               and $\mathrm{GA\text{-}DDH}_{x_{b_{i^*}}}(x, x_{i^*}^{\mathsf{U}}, z_{i^*}) = 1$
23                 $T_{\mathrm{bad}} := T_{\mathrm{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z)\}$
24         else
25             if $\mathrm{GA\text{-}DDH}_{x_{b_i}}(\tilde{x}, x_i^{\mathsf{U}}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
26                 $T_{\mathrm{bad}} := T_{\mathrm{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z)\}$
27     $K \xleftarrow{\$} \mathcal{K}$
28     $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}] := (\mathsf{S}, (s_1, ..., s_\ell), K)$
29 else
30     $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ false
31     $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
32     if $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}) = \mathsf{tr}^*$
       and $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z) \in T_{\mathsf{H}}$
       and $\mathrm{GA\text{-}DDH}_{x_{b_i}}(\tilde{x}, x_i^{\mathsf{U}}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell] \setminus \{i^*\}$
       and $\mathrm{GA\text{-}DDH}_{x_{b_{i^*}}}(x, x_{i^*}^{\mathsf{U}}, z_{i^*}) = 1$
33         $K := T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z]$
34     else if $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z) \in T_{\mathsf{H}}$
       and $\mathrm{GA\text{-}DDH}_{x_{b_i}}(\tilde{x}, x_i^{\mathsf{U}}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
35         $K := T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z]$
36     else
37         $K \xleftarrow{\$} \mathcal{K}$
38         $T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, (s_1, ..., s_\ell)] := K$
39 $\pi_{\mathsf{S}}^t := ((s_1, ..., s_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), K, $ true$)$
40 return $(\mathsf{S}, x^{\mathsf{S}})$
```

```
SendTermResp$(\mathsf{U}, t, \mathsf{S}, x^{\mathsf{S}})$
41 $\mathrm{cnt} := \mathrm{cnt} + 1$
42 if $\pi_{\mathsf{U}}^t \neq ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \bot, \mathsf{com}), \bot, \bot)$
43     return $\bot$
44 if $\mathsf{G}(x^{\mathsf{S}}) \neq \mathsf{com}$
45     $\pi_{\mathsf{U}}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), \bot, $ false$)$
46     return $\bot$
47 if $\exists \mathsf{P} \in \mathcal{U}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
48     return $\bot$
49 if $\pi_{\mathsf{U}}^t.\mathsf{tr} = \mathsf{tr}^*$
50     $\mathsf{tr}^* := (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
51 if $\exists t'$ s. t. $\pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
   and $\pi_{\mathsf{S}}^{t'}.\mathsf{fr} =$ true
52     $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ true
53     $(\mathsf{S}, (s_1, ..., s_\ell), K) := T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}]$
54 else if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
55     $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ true
56     $\forall \mathsf{pw}, z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z) \in T_{\mathsf{H}}$
57         $(b_1, ..., b_\ell) := \mathsf{pw}$
58         if $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}) = \mathsf{tr}^*$
59             if $\mathrm{GA\text{-}DDH}_{x_{b_i}}(\tilde{x}, x_i^{\mathsf{S}}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell] \setminus \{i^*\}$
               and $\mathrm{GA\text{-}DDH}_{x_{b_{i^*}}}(x, x_{i^*}^{\mathsf{S}}, z_{i^*}) = 1$
60                 $T_{\mathrm{bad}} := T_{\mathrm{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z)\}$
61         else
62             if $\mathrm{GA\text{-}DDH}_{x_{b_i}}(\tilde{x}, x_i^{\mathsf{S}}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
63                 $T_{\mathrm{bad}} := T_{\mathrm{bad}} \cup \{(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z)\}$
64     $K \xleftarrow{\$} \mathcal{K}$
65     $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}] := (\mathsf{U}, (u_1, ..., u_\ell), K)$
66 else
67     $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ false
68     $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
69     if $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}) = \mathsf{tr}^*$
       and $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z) \in T_{\mathsf{H}}$
       and $\mathrm{GA\text{-}DDH}_{x_{b_i}}(\tilde{x}, x_i^{\mathsf{S}}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell] \setminus \{i^*\}$
       and $\mathrm{GA\text{-}DDH}_{x_{b_{i^*}}}(x, x_{i^*}^{\mathsf{S}}, z_{i^*}) = 1$
70         $K := T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z]$
71     else if $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z) \in T_{\mathsf{H}}$
       and $\mathrm{GA\text{-}DDH}_{x_{b_i}}(\tilde{x}, x_i^{\mathsf{S}}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
72         $K := T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z]$
73     else
74         $K \xleftarrow{\$} \mathcal{K}$
75         $T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, (u_1, ..., u_\ell)] := K$
76 $\pi_{\mathsf{U}}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), K, $ true$)$
77 return true
```

Fig. 23. Oracles SendTermInit and SendTermResp for adversary $\mathcal{B}_2$ in Figure 22.

recognize where the challenge was embedded, $\mathcal{B}_2$ marks the trace of this instance as the target trace tr^* (lines 35-37). From now on, $\mathcal{B}_2$ also has access to decision oracles $\mathrm{GA\text{-}DDH}_{x_0}(x, \cdot, \cdot)$, $\mathrm{GA\text{-}DDH}_{x_1}(x, \cdot, \cdot)$.

Queries to SendTermInit are simulated similarly (see Figure 23). After increasing the counter, $\mathcal{B}_2$ computes x^{S} and if this is the τ^*-th query, it outputs $x_{i^*}^{\mathsf{U}}$ as commitment to receive x. $x_{i^*}^{\mathsf{S}}$ is then set to x, implicitly setting $s_{i^*} = g$. $\mathcal{B}_2$ also marks the trace as tr^* (lines 05-11). Now $\mathcal{B}_2$ also needs to compute a session key. If the instance is fresh, we must check if there already exists an entry in T_{H} that causes an inconsistency. As in G_9, we iterate over all pw, z in T_{H} that contain the trace of this instance (line 19). In particular, we must check whether z_i satisfies

$$z_i = \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{U}}, s_i \star \tilde{x}) \quad \Leftrightarrow \quad \mathsf{GA\text{-}CDH}_{x_{b_i}}(\tilde{x}, x_i^{\mathsf{U}}) = s_i^{-1} \star z_i \ ,$$

which we can do using the GA-DDH oracle and the equation on the right-hand side (lines 24-26). When checking for an inconsistency of tr^*, we need to call the corresponding additional oracle (where the challenge x is fixed) for the i^*-th element (lines 21-23). In this case, we need to check if

$$z_{i^*} = \mathsf{GA\text{-}CDH}_{x_{b_{i^*}}}(x_{i^*}^{\mathsf{U}}, x_{i^*}^{\mathsf{S}}) = \mathsf{GA\text{-}CDH}_{x_{b_{i^*}}}(x_{i^*}^{\mathsf{U}}, x) = \mathsf{GA\text{-}CDH}_{x_{b_{i^*}}}(x, x_{i^*}^{\mathsf{U}}) \ .$$

If all z_i are valid, then we add this entry to T_{bad}.

If the instance is not fresh, i.e., the password is corrupted, then we have to compute the correct key. We check list T_{H} for a valid entry z as explained above and if it exists, we assign this value to the session key (lines 32-35). Here we also must treat tr^* accordingly. Otherwise, we choose a random key and add a special entry to T_{H}, which instead of z contains the secret group elements s_i (lines 37, 38) so that we can patch the random oracle later.

SendTermInit is simulated analogously, using the secret group elements u_i. Here we must first update tr^* if necessary (line 50).

Now we look at the random oracle queries to H (see Figure 22). If the trace is contained in set T_{s} (line 44) which means the corresponding instance was fresh when the send query was issued, we check if z is valid using the GA-DDH oracle. We do this as described above, depending on whether it is a user or server instance (lines 45, 58) and depending on whether it is tr^* (lines 46, 59) or not (lines 52, 65). In case z is valid, we first check if the instance is still fresh (i.e., the password was not corrupted in the meantime) and if this is the case, we add the query to T_{bad} (lines 49, 55, 62, 68). Otherwise, if the password was corrupted and is specified in the query, we return the session key stored in T_{s} (lines 51, 57, 64, 70).

Next, we check if the query matches a special entry in T_{H} that was added in SendTermInit or SendTermResp for a non-fresh instance, which means we have to output the same key that was chosen before. Again, we can use the GA-DDH oracle and differentiate between user and server instances (lines 71, 78) and tr^*(lines 72, 79).

After $\mathcal{A}$ terminates with output β', $\mathcal{B}_2$ chooses the passwords which have not been generated yet in a Corrupt query (line 07). Then we check for event $\mathbf{bad}_{\mathsf{pw}}$ (line 08). If $\mathbf{bad}_{\mathsf{pw}}$ occurred, then there must be at least two entries in T_{bad} for the same trace and different passwords pw and pw' along with values z and z'. We check if this is the case for the target trace tr^* (line 09) and if the two passwords differ in the i^*-th bit (line 12). In this case $\mathcal{B}_2$ will output z_{i^*} and z'_{i^*} for $b_{i^*} = 0$ and $b'_{i^*} = 1$ (line 14) or it must swap the output. Otherwise, $\mathcal{B}_2$ aborts.

Recall that in case tr^* belongs to a user instance, $\mathcal{B}_2$ committed on $y = x^{\mathsf{S}}_{i^*}$ and embedded the challenge x in $x^{\mathsf{U}}_{i^*}$. To solve the ISim-GA-StCDH problem, $\mathcal{B}_2$ needs to compute

$$\begin{aligned} y_0 &= \text{GA-CDH}_{x_0}(x, y) = \text{GA-CDH}_{x_0}(x^{\mathsf{U}}_{i^*}, x^{\mathsf{S}}_{i^*}) = z_{i^*} \text{ if } b_{i^*} = 0, \\ y_1 &= \text{GA-CDH}_{x_1}(x, y) = \text{GA-CDH}_{x_1}(x^{\mathsf{U}}_{i^*}, x^{\mathsf{S}}_{i^*}) = z'_{i^*} \text{ if } b'_{i^*} = 1, \end{aligned}$$

which is exactly what is stored in T_{bad}. If tr^* belongs to a server instance, the analysis works analogously.

Therefore, $\mathcal{B}_2$ is successful whenever its guesses are correct and $\mathcal{A}$ issues two queries for the target trace and both password bits. This concludes the analysis of $\mathbf{bad}_{\mathsf{pw}}$.

Next, we analyze event $\mathbf{bad}_{\mathrm{guess}}$. Recall that $\mathbf{bad}_{\mathrm{guess}}$ happens only if $\mathbf{bad}_{\mathsf{pw}}$ does not happen. Hence, for each instance there is at most one entry in T_{bad} and the size of T_{bad} is at most q_s. As all entries were added before the corresponding password was sampled, the probability is bounded by

$$\Pr[\mathsf{G}_9 \Rightarrow \mathbf{bad}_{\mathrm{guess}}] \leq \frac{q_s}{|\mathcal{PW}|} \enspace .$$

Finally, note that if none of the bad events happens in G_9, all session keys output by Test are uniformly random and the adversary can only guess β. Hence, $\Pr[\mathsf{G}_9 \Rightarrow 1] = \frac{1}{2}$. Collecting the probabilities and applying Lemma 4 yields the bound in Theorem 2. □

E.2 Variants of Com-GA-PAKE$_\ell$

In Section 8 we described optimizations that can be used to reduce the number of group action evaluations in an execution of the protocol. Here, we show that Com-GA-PAKE$_\ell$ remains secure when one or both optimizations is applied.

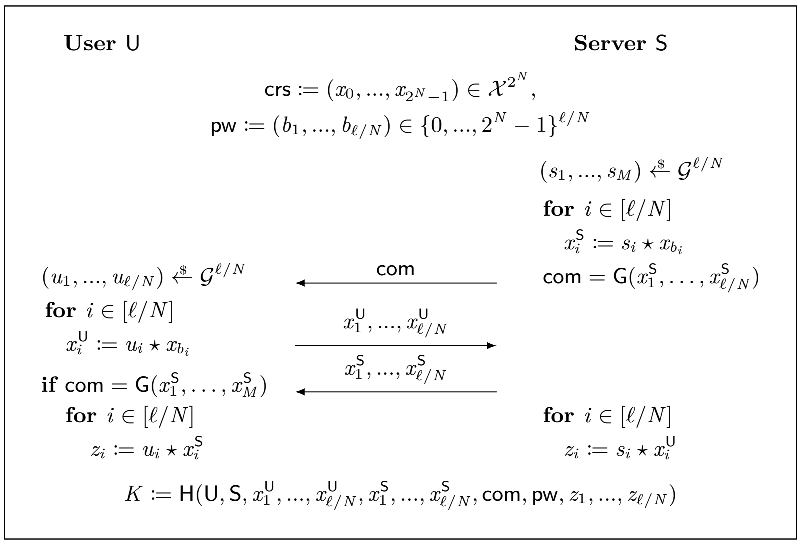
Com-GA-PAKE$_{\ell,N}$. Let N be some positive integer dividing ℓ. We set

$$\mathsf{crs} := (x_0, \dots, x_{2^N-1}) \in \mathcal{X}^{2^N}$$

and divide the password into ℓ/N blocks of length N

$$\mathsf{pw} = (b_1, ..., b_{\ell/N}) \in \{0, ..., 2^N - 1\}^{\ell/N}.$$

The general outline of the protocol does not change. The only difference is that both the server and the user only generate ℓ/N random group elements (instead of ℓ). Hence they only need to perform $2 \cdot \ell/N$ group action evaluations in total. We write $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell,N}$ for this variant of the protocol. The alterations are summarized in Figure 24.



User U
Server S
$\mathsf{crs} := (x_0, ..., x_{2^N-1}) \in \mathcal{X}^{2^N}$,
$\mathsf{pw} := (b_1, ..., b_{\ell/N}) \in \{0, ..., 2^N - 1\}^{\ell/N}$
$(s_1, ..., s_M) \xleftarrow{\$} \mathcal{G}^{\ell/N}$
for $i \in [\ell/N]$
$x_i^{\mathsf{S}} := s_i \star x_{b_i}$
$\mathsf{com} = \mathsf{G}(x_1^{\mathsf{S}}, \ldots, x_{\ell/N}^{\mathsf{S}})$
$\xleftarrow{\mathsf{com}}$
$(u_1, ..., u_{\ell/N}) \xleftarrow{\$} \mathcal{G}^{\ell/N}$
for $i \in [\ell/N]$
$x_i^{\mathsf{U}} := u_i \star x_{b_i}$
$\xrightarrow{x_1^{\mathsf{U}}, ..., x_{\ell/N}^{\mathsf{U}}}$
$\xleftarrow{x_1^{\mathsf{S}}, ..., x_{\ell/N}^{\mathsf{S}}}$
if $\mathsf{com} = \mathsf{G}(x_1^{\mathsf{S}}, \ldots, x_M^{\mathsf{S}})$
for $i \in [\ell/N]$
$z_i := u_i \star x_i^{\mathsf{S}}$
for $i \in [\ell/N]$
$z_i := s_i \star x_i^{\mathsf{U}}$
$K := \mathsf{H}(\mathsf{U}, \mathsf{S}, x_1^{\mathsf{U}}, ..., x_{\ell/N}^{\mathsf{U}}, x_1^{\mathsf{S}}, ..., x_{\ell/N}^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}, z_1, ..., z_{\ell/N})$

Fig. 24. $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell,N}$ with some $N \mid \ell$.

Theorem 5 (Security of $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell,N}$). *For any adversary $\mathcal{A}$ against $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell,N}$ that issues at most q_e queries to oracle* Execute *and q_s queries to oracle* SendInit *and* SendResp*, there exist adversary $\mathcal{B}_1$ against* GA-StCDH*, $\mathcal{B}_2$ against* GA-GapCDH *such that*

$$\mathsf{Adv}_{\mathsf{Com\text{-}GA\text{-}PAKE}_\ell}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{GA\text{-}StCDH}}(\mathcal{B}_1) + \frac{2q_s\ell}{N} \cdot \sqrt{\mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{GA\text{-}GapCDH}}(\mathcal{B}_2)} + \frac{(q_s + q_e)^2}{|\mathcal{G}|^{\ell/N}} + \frac{q_{\mathsf{G}}\, q_s}{|\mathcal{G}|^{\ell/N}} + \frac{2 \cdot (q_{\mathsf{G}} + q_s + q_e)^2}{2^\lambda} + \frac{q_s}{|\mathcal{PW}|} \;\;,$$

where λ is the output length of G *in bits.*

The proof of Theorem 5 is very similar to the proof of Theorem 2. Therefore we do not give a full proof for the security of $\mathsf{Com\text{-}GA\text{-}PAKE}_{\ell,N}$, but shortly explain the difference between the two protocols.

The main difference appears in the analysis of the event $\mathbf{bad}_{\mathsf{pw}}$. Here, it is not possible to construct an adversary against ISim-GA-StCDH as was done in the original proof. Instead, we construct an adversary against 2^{N}ISim-GA-StCDH (Definition 16). This construction is a straight-forward adaption of the original construction, therefore we do not give any details here. But we show that the new assumption can be reduced to ISim-GA-StCDH (Theorem 6).

Definition 16 (2^{N}-Interactive Simultaneous GA-StCDH (2^{N}ISim-GA-StCDH)). *On input $(x_0 = g_0 \star \tilde{x}, ..., x_{2^N-1} = g_{2^N-1} \star \tilde{x}) \in \mathcal{X}^{2^N}$, the adversary first chooses and commits to some $y \in \mathcal{X}$. After receiving the challenge $x = g \star \tilde{x} \in \mathcal{X}$, the* 2^{N}ISim-GA-StCDH *problem requires to compute $y_0 = gg_i^{-1} \star y$ and $y_1 = gg_j^{-1} \star y$ for one pair $i \neq j \in \{0, ..., 2^N - 1\}$. For a group action* $\mathsf{XXX} \in \{\mathsf{EGA}, \mathsf{REGA}, \mathsf{EGAT}, \mathsf{REGAT}\}$, *we define the advantage function of an adversary $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathsf{XXX}}^{\mathsf{2^N ISim\text{-}GA\text{-}StCDH}}(\mathcal{A}) := \Pr\left[\begin{array}{c|c} i \neq j & (g_0, ..., g_{2^N}) \xleftarrow{\$} \mathcal{G}^{2^N} \\ y_0 = \mathsf{GA\text{-}CDH}_{x_i}(x, y) & (x_0, ..., x_{2^N-1}) = (g_0 \star \tilde{x}, ..., g_{2^N-1} \star \tilde{x}) \\ y_1 = \mathsf{GA\text{-}CDH}_{x_j}(x, y) & y \leftarrow \mathcal{A}^{\mathrm{O}_1}(x_0, ..., x_{2^N-1}) \\ & g \xleftarrow{\$} \mathcal{G} \\ & x = g \star \tilde{x} \\ & (y_0, y_1) \leftarrow \mathcal{A}^{\mathrm{O}_1, \mathrm{O}_2}(x) \end{array}\right] ,$$

where $\mathrm{O}_1 = \{\mathrm{GA\text{-}DDH}_{x_k}(\tilde{x}, \cdot, \cdot)\}_k$ *and* $\mathrm{O}_2 = \{\mathrm{GA\text{-}DDH}_{x_k}(x, \cdot, \cdot)\}_k$ *with* $k \in \{0, ..., 2^N - 1\}$.

Theorem 6 **(ISim-GA-StCDH implies $\mathsf{2^N}$ISim-GA-StCDH).** *For any adversary $\mathcal{A}$ against* $\mathsf{2^N}$ISim-GA-StCDH, *there exists adversary $\mathcal{B}$ against* ISim-GA-StCDH *such that*

$$\mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{2^N ISim\text{-}GA\text{-}StCDH}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{ISim\text{-}GA\text{-}StCDH}}(\mathcal{B}) .$$

Proof. We construct adversary $\mathcal{B}$ as follows. On input $(x_0, x_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x})$, for each $i \in \{0, ..., 2^N - 1\}$ $\mathcal{B}$ chooses a random bit $b_i \xleftarrow{\$} \{0, 1\}$, a random group element $h_i \xleftarrow{\$} \mathcal{G}$ and computes $x_i = h_i \star x_{b_i}$. Then it runs $\mathcal{A}$ on input $(x_0, ..., x_{2^N-1})$. When $\mathcal{A}$ commits on y, $\mathcal{B}$ forwards the commitment to receive the challenge x which it gives to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs (i, j) and (y_0, y_1) such that $i \neq j$ and $y_0 = \mathsf{GA\text{-}CDH}_{x_i}(x, y)$, $y_1 = \mathsf{GA\text{-}CDH}_{x_j}(x, y)$. If $b_i = b_j$ which happens with probability 1/2, then $\mathcal{B}$ aborts. Otherwise, note that $x_i = (h_i g_{b_i}) \star \tilde{x}$ and thus $y_0 = (h_i^{-1} g_{b_i}^{-1} g) \star y$, so $\mathcal{B}$ can compute the solution $y_0' = h_i \star y_0 = g_{b_i}^{-1} g \star y$ and equivalently $y_1' = h_j \star y_1 = g_{b_j}^{-1} g \star y$. If $b_j = 0$ and $b_i = 1$, the output of $\mathcal{B}$ must be swapped.

During the experiment, $\mathcal{A}$ also has access to decision oracles $\mathrm{GA\text{-}DDH}_{x_i}(\tilde{x}, \cdot, \cdot)$ and $\mathrm{GA\text{-}DDH}_{x_i}(x, \cdot, \cdot)$ for $i \in \{0, ..., 2^N - 1\}$. These can be easily simulated using $\mathcal{B}$'s decision oracles for x_0 and x_1. On a query $\mathrm{GA\text{-}DDH}_{x_i}(\tilde{x}, z_1, z_2)$, $\mathcal{B}$ queries its own oracle $\mathrm{GA\text{-}DDH}_{x_{b_i}}(\tilde{x}, z_1, h_i \star z_2)$ and fowards the output to $\mathcal{A}$. Analogously, on a query $\mathrm{GA\text{-}DDH}_{x_i}(x, z_1, z_2)$, $\mathcal{B}$ queries $\mathrm{GA\text{-}DDH}_{x_{b_i}}(x, z_1, h_i \star z_2)$. □

Twisted version of Com-GA-PAKE$_\ell$. Here, we analyze the security of Com-GA-PAKE$_\ell^\mathsf{t}$, the twisted version of Com-GA-PAKE$_\ell$, as defined in Section 8.2. In contrast to the situation for X-GA-PAKE$_\ell^\mathsf{t}$, Theorem 2 needs to be changed slightly. Before presenting the new theorem, we need to introduce the the following assumption.

Definition 17 **(Square GA-GapCDH (Sq-GA-GapCDH)).** *On input $g \star \tilde{x} \in \mathcal{X}$, the* Sq-GA-GapCDH *problem requires to compute the set element $g^2 \star \tilde{x}$. To an effective group action* XXX $\in$ {EGA, REGA, EGAT, REGAT}, *we associate the advantage function of an adversary $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathsf{XXX}}^{\mathsf{Sq\text{-}GA\text{-}GapCDH}}(\mathcal{A}) := \Pr[\mathcal{A}^{\mathrm{GA\text{-}DDH}_*}(g \star \tilde{x}) \Rightarrow g^2 \star \tilde{x}] ,$$

where $g \xleftarrow{\$} \mathcal{G}$ and $\mathcal{A}$ has access to a general decision oracle $\mathrm{GA\text{-}DDH}_*$.

Note that in the CSIDH setting, the square group action computational Diffie-Hellman problem coincides with Problem 3 in [27].

Theorem 7 **(Security of Com-GA-PAKE$_\ell^\mathsf{t}$).** *For any adversary $\mathcal{A}$ against* Com-GA-PAKE$_\ell^\mathsf{t}$ *that issues at most q_e execute queries, q_s send queries and at most q_G and q_H queries to random oracles* G *and* H, *there exist an adversary $\mathcal{B}_1$ against* GA-StCDH *and an adversary $\mathcal{B}_2$ against* Sq-GA-GapCDH *such that*

$$\mathsf{Adv}_{\mathsf{Com\text{-}GA\text{-}PAKE}_\ell^\mathsf{t}}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{GA\text{-}StCDH}}(\mathcal{B}_1) + q_s\ell \cdot \sqrt{\mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{Sq\text{-}GA\text{-}GapCDH}}(\mathcal{B}_2)} + \frac{(q_s + q_e)^2}{|\mathcal{G}|^\ell} + \frac{q_\mathsf{G} q_s}{|\mathcal{G}|^\ell} + \frac{2 \cdot (q_\mathsf{G} + q_s + q_e)^2}{2^\lambda} + \frac{q_s}{|\mathcal{PW}|} ,$$

where λ is the output length of G *in bits.*

Again, we do not provide a full proof of the theorem since most parts are completely analogous to the proof of Theorem 2. For the most part, one can just replace x_1 by x_0^t everywhere in the proof. The only significant difference occurs in the analysis of the event $\mathbf{bad}_{\mathsf{pw}}$. In particular the restriction $x_1 = x_0^t$ does not allow to construct an adversary against ISim-GA-StCDH. Instead, we need to consider the following alteration of ISim-GA-StCDH for the security analysis.

Definition 18 (Twisted Interactive Simultaneous GA-StCDH (TISim-GA-StCDH)). *On input $x_0 = g_0 \star \tilde{x} \in \mathcal{X}$, the adversary first chooses and commits to some $y \in \mathcal{X}$. After receiving the challenge $x = g \star \tilde{x} \in \mathcal{X}$, the* (TISim-GA-StCDH) *problem requires to compute $y_0 = gg_0^{-1} \star y, y_1 = gg_0 \star y$. For a group action* XXX $\in$ {EGA, REGA, EGAT, REGAT}*, we define the advantage function of an adversary $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathsf{XXX}}^{\mathsf{TISim\text{-}GA\text{-}StCDH}}(\mathcal{A}) := \Pr\left[\begin{array}{l|c} & g_0 \xleftarrow{\$} \mathcal{G} \\ & x_0 = g_0 \star \tilde{x} \\ y_0 = \mathsf{GA\text{-}CDH}_{x_0}(x, y) & y \leftarrow \mathcal{A}^{\mathrm{O}_1}(x_0) \\ y_1 = \mathsf{GA\text{-}CDH}_{x_0^t}(x, y) & g \xleftarrow{\$} \mathcal{G} \\ & x = g \star \tilde{x} \\ & (y_0, y_1) \leftarrow \mathcal{A}^{\mathrm{O}_1, \mathrm{O}_2}(x) \end{array}\right],$$

where $\mathrm{O}_1 = \{\mathrm{GA\text{-}DDH}_{x_0}(\tilde{x}, \cdot, \cdot), \mathrm{GA\text{-}DDH}_{x_0^t}(\tilde{x}, \cdot, \cdot)\}$ *and* $\mathrm{O}_2 = \{\mathrm{GA\text{-}DDH}_{x_0}(x, \cdot, \cdot), \mathrm{GA\text{-}DDH}_{x_0^t}(x, \cdot, \cdot)\}$.

Recall that for the proof of Com-GA-PAKE$_\ell$, we showed that the ISim-GA-StCDH is implied by GA-GapCDH (Lemma 4). In the same way, one can show that TISim-GA-StCDH is implied by Sq-GA-GapCDH, more precisely

$$\mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{TISim\text{-}GA\text{-}StCDH}}(\mathcal{A}) \leq \sqrt{\mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{Sq\text{-}GA\text{-}GapCDH}}(\mathcal{B})} \ .$$

This explains the dependence on the new security assumption Sq-GA-GapCDH for this version of Com-GA-PAKE$_\ell$.

F PAKE with Perfect Forward Secrecy

When considering perfect forward secrecy, the security experiment is the same as $\mathsf{Exp}_{\mathsf{PAKE}}$ described in Section 4. However, we replace the forth freshness condition by the following:

3.4 No partner exists and Corrupt was not queried prior to acceptance.

As a result, the Corrupt oracle does not need to check anymore whether an instance that has no partner instance has been tested before or not. If it has tested, it must have been fresh at that point which is not changed by a corruption query.

Definition 19 (Security of PAKE with Forward Security). *We define the security experiment as $\mathsf{Exp}_{\mathsf{PAKE}}$ with the additional property in the freshness definition. The advantage of an adversary $\mathcal{A}$ against a password authenticated key exchange protocol* PAKE *in $\mathsf{Exp}_{\mathsf{PAKE}}^{\mathsf{pfs}}$ is defined as*

$$\mathsf{Adv}_{\mathsf{PAKE}}^{\mathsf{pfs}}(\mathcal{A}) := \left|\Pr[\mathsf{Exp}_{\mathsf{PAKE}}^{\mathsf{pfs}} \Rightarrow 1] - \frac{1}{2}\right| \ .$$

F.1 Perfect Forward Secrecy of GA-PAKE$_\ell$

In order to consider perfect forward secrecy, we need an interactive and password-based security assumption. This is similar as in the analysis of SPAKE2 [1] for example.

Definition 20 (Password-based GA-StCDH (Pw-GA-StCDH)). *On input $(x_0, x_1, x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}) = (g_0 \star \tilde{x}, g_1 \star \tilde{x}, p_1 \star \tilde{x}, ..., p_\ell \star \tilde{x})$, the* Pw-GA-StCDH *problem requires the adversary to commit on $(y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}) \in \mathcal{X}^\ell$ and then compute $z_i = \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{P}}, y_i^{\mathsf{P}}) = (g_{b_i}^{-1} \cdot p_i) \star y_i^{\mathsf{P}}\ \forall i \in [\ell]$ after receiving the challenge password*

$\mathsf{pw} = (b_1, ..., b_\ell) \in \{0,1\}^\ell$. *To an effective group action* $\mathsf{XXX} \in \{\mathsf{EGA}, \mathsf{REGA}\}$*, we define the advantage function of* $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathsf{Pw\text{-}GA\text{-}StCDH}}_{\mathsf{XXX}}(\mathcal{A}) := \Pr\left[z_i = \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{P}}, y_i^{\mathsf{P}})\ \forall i \in [\ell] \;\middle|\; \begin{array}{c} (g_0, g_1, p_1, ..., p_\ell) \xleftarrow{\$} \mathcal{G}^{\ell+2} \\ (x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x}) \\ (x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}) := (p_1 \star \tilde{x}, ..., p_\ell \star \tilde{x}) \\ (y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}) \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1, x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}) \\ \mathsf{pw} := (b_1, ..., b_\ell) \xleftarrow{\$} \mathcal{PW} \\ (z_1, ..., z_\ell) \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pw}) \end{array} \right],$$

where $\mathrm{O} = \{\text{GA-DDH}_{x_j}(x_i^{\mathsf{P}}, \cdot, \cdot)\}_{i \in [\ell], j \in \{0,1\}}$.

Lemma 5. *The simultaneous* $\mathsf{GA\text{-}StCDH}$ ($\mathsf{Sim\text{-}GA\text{-}StCDH}$) *implies the password-based* $\mathsf{GA\text{-}StCDH}$ ($\mathsf{Pw\text{-}GA\text{-}StCDH}$), *more precisely*

$$\mathsf{Adv}^{\mathsf{Pw\text{-}GA\text{-}StCDH}}_{\mathsf{EGA}}(\mathcal{A}) \leq \sqrt{\mathsf{Adv}^{\mathsf{Sim\text{-}GA\text{-}StCDH}}_{\mathsf{EGA}}(\mathcal{B})} + \frac{1}{|\mathcal{PW}|}.$$

Proof. Intuitively, the term $1/|\mathcal{PW}|$ comes from the fact that the adversary can trivially solve the problem when guessing the password directly. For the proof we apply the reset lemma (see Lemma 2), where $H = \mathcal{PW}$.

Let $\mathcal{A}$ be an adversary against the $\mathsf{Pw\text{-}GA\text{-}StCDH}$ problem. Now consider adversary $\mathcal{B}$ in Figure 25 that takes as input three set elements (x, x_0, x_1) and a password pw. It also has access to decision oracles. First, $\mathcal{B}$ chooses $(p_1, ...p_\ell) \xleftarrow{\$} \mathcal{G}^\ell$ and computes $x_i^{\mathsf{P}} = p_i \star x$ for all $i \in [\ell]$. Then it runs $\mathcal{A}$ on input $(x_0, x_1, x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}})$ and receives a commitment $(y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}})$. Now $\mathcal{B}$ gives pw to $\mathcal{A}$ and $\mathcal{A}$ will finally output $(z_1, ..., z_\ell)$. $\mathcal{B}$ checks if the solution is correct using the decision oracles and if this is the case, it outputs $I = 1$ and $\sigma = (\mathsf{pw}, p_1, ..., p_\ell, y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z_1, ..., z_\ell)$. Otherwise it outputs $(0, \epsilon)$.

```
$\mathcal{B}^{\text{GA-DDH}_{x_0}(x,\cdot,\cdot),\text{GA-DDH}_{x_1}(x,\cdot,\cdot)}(x, x_0, x_1, \mathsf{pw})$
$(p_1, ..., p_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$(x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}) := (p_1 \star x, ..., p_\ell \star x)$
$(y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}) \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1, x_1^{\mathsf{P}}, ...x_\ell^{\mathsf{P}})$
$(z_1, ..., z_\ell) \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pw})$
$(b_1, ..., b_\ell) := \mathsf{pw}$
if $\text{GA-DDH}_{x_{b_i}}(x, y_i^{\mathsf{P}}, p_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
    return $(1, (\mathsf{pw}, p_1, ..., p_\ell, y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z_1, ..., z_\ell))$
return $(0, \epsilon)$

$\{\text{GA-DDH}_{x_j}(x_i^{\mathsf{P}}, y', z')\}_{i \in [\ell], j \in \{0,1\}}$
return $\text{GA-DDH}_{x_j}(x, y', p_i^{-1} \star z')$
```

```
$\mathcal{C}^{\text{GA-DDH}_{x_0}(x,\cdot,\cdot),\text{GA-DDH}_{x_1}(x,\cdot,\cdot)}(x, x_0, x_1)$
$(b^*, \sigma, \sigma') \leftarrow \mathcal{R}_{\mathcal{B}}^{\mathrm{O}'}(x, x_0, x_1)$
if $b^* = 0$
    abort and return $\perp$
$(\mathsf{pw}, p_1, ..., p_\ell, y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z_1, ..., z_\ell) := \sigma$
$(\mathsf{pw}', p_1, ..., p_\ell, y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z_1', ..., z_\ell') := \sigma'$
$(b_1, ..., b_\ell) := \mathsf{pw}$
$(b_1', ..., b_\ell') := \mathsf{pw}'$
Find $i$ such that $b_i \neq b_i'$
return $(y_i^{\mathsf{P}}, p_i^{-1} \star z_i, p_i^{-1} \star z_i')$
```

Fig. 25. Adversaries $\mathcal{B}$ and $\mathcal{C}$ against $\mathsf{Sim\text{-}GA\text{-}StCDH}$ for the proof of Lemma 5. $\mathcal{A}$ has access to $\mathrm{O} = \{\text{GA-DDH}_{x_j}(x_i^{\mathsf{P}}, \cdot, \cdot)\}_{i \in [\ell], j \in \{0,1\}}$. Reset algorithm $\mathcal{R}_{\mathcal{B}}$ has access to $\mathrm{O}' = \{\text{GA-DDH}_{x_0}(x, \cdot, \cdot), \text{GA-DDH}_{x_1}(x, \cdot, \cdot)\}$.

If $\mathcal{A}$ issues a query to a decision oracle $\text{GA-DDH}_{x_j}(x_i^{\mathsf{P}}, y', z')$ for some $i \in [\ell]$ and $j \in \{0,1\}$, $\mathcal{B}$ queries its own decision oracle $\text{GA-DDH}_{x_j}(x, y', p_i^{-1} \star z')$ and forwards the output to $\mathcal{A}$.

Let IG be the algorithm that chooses $g, g_0, g_1 \xleftarrow{\$} \mathcal{G}$ and outputs $(x, x_0, x_1) = (g \star \tilde{x}, g_0 \star \tilde{x}, g_1 \star \tilde{x})$. Let acc be defined as in Lemma 2, thus

$$\mathrm{acc} \geq \mathsf{Adv}^{\mathsf{Pw\text{-}GA\text{-}StCDH}}_{\mathsf{EGA}}(\mathcal{A}) \ .$$

Let $\mathcal{R}_{\mathcal{B}}$ be the reset algorithm associated to $\mathcal{B}$ as in Lemma 2 with access to the same decision oracles as $\mathcal{B}$. Then we construct adversary $\mathcal{C}$ against $\mathsf{Sim\text{-}GA\text{-}StCDH}$ as in Figure 25. $\mathcal{C}$ runs the reset algorithm to obtain a bit b^* as well as the two side outputs σ, σ' each consisting of a password, ℓ group elements p_i, ℓ set elements y_i^{P} and ℓ set elements z_i. If $b^* = 0$, it aborts. If the reset algorithm was successful, note that $\mathsf{pw} \neq \mathsf{pw}'$, but $p_i = p_i'$ and $y_i^{\mathsf{P}} = y_i^{\mathsf{P}'}$ for all $i \in [\ell]$ as we run $\mathcal{B}$ on the same random coins. Now $\mathcal{C}$ looks for the first index i where the two passwords differ and outputs the solution $(y, y_0, y_1) = (y_i^{\mathsf{P}}, p_i^{-1} \star z_i, p_i^{-1} \star z_i')$

which solves $\mathsf{Sim\text{-}GA\text{-}StCDH}$ as $z_i = \mathsf{GA\text{-}CDH}_{x_0}(x_i^{\mathsf{P}}, y_i^{\mathsf{P}}) = (g_0^{-1} \cdot p_i) \star x_i^{\mathsf{P}}$ and $z_i' = \mathsf{GA\text{-}CDH}_{x_1}(x_i^{\mathsf{P}}, y_i^{\mathsf{P}}) = (g_1^{-1} \cdot p_i) \star y_i^{\mathsf{P}}$.

Applying Lemma 2, we get the bound stated in Lemma 5. ◻

Theorem 8 (Perfect Forward Secrecy of $\mathsf{GA\text{-}PAKE}_\ell$). *For any adversary $\mathcal{A}$ against $\mathsf{GA\text{-}PAKE}_\ell$ that issues at most q_e execute queries and q_s send queries and where H is modeled as a random oracle, there exist adversary $\mathcal{B}_1$ against $\mathsf{GA\text{-}StCDH}$ and adversaries $\mathcal{B}_2$, $\mathcal{B}_3$ against $\mathsf{Sim\text{-}GA\text{-}StCDH}$ such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{pfs}}_{\mathsf{GA\text{-}PAKE}_\ell}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{GA\text{-}StCDH}}_{\mathsf{EGA}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{Sim\text{-}GA\text{-}StCDH}}_{\mathsf{EGA}}(\mathcal{B}_2) + q_s \cdot \sqrt{\mathsf{Adv}^{\mathsf{Sim\text{-}GA\text{-}StCDH}}_{\mathsf{EGA}}(\mathcal{B}_3)} \\ + \frac{2q_s}{|\mathcal{PW}|} + \frac{(q_s+q_e)^2}{|\mathcal{G}|^\ell} \ .\end{aligned}$$

Proof. The proof follows the one of Theorem 3 very closely. However, we have to consider that an instance is fresh if the password was not corrupted when the instance accepts. We keep games G_0-G_6 from Figures 12, 14 and 15 almost as they are, with the following differences.

In G_2, we do not update the freshness variable in the CORRUPT oracle in case the instance was tested (line 88, Figure 12) because this is now a valid query.

Recall that in G_5, we raise flag **bad** whenever there is an inconsistency between the random oracle list T and the list of keys from send queries T_{s}. Now we also want keys to be random even if the password was corrupted afterwards, hence we raise **bad** in H in this case as well (line 38, Figure 14), instead of outputting the real session key.

This also translates to G_6, where we will then raise flag $\mathbf{bad}_{\mathrm{pfs}}$ at this point (line 39, Figure 15). Thus, we now have

$$\Pr[\mathsf{G}_5 \Rightarrow \mathbf{bad}] \le \Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\mathsf{pw}}] + \Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\mathrm{guess}}] + \Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\mathrm{pfs}}] \ ,$$

where $\mathbf{bad}_{\mathsf{pw}}$ and $\mathbf{bad}_{\mathrm{guess}}$ can be bounded as in the proof of Theorem 3.

It remains to bound $\mathbf{bad}_{\mathrm{pfs}}$. Therefore, we construct an adversary $\mathcal{B}_3$ against $\mathsf{Pw\text{-}GA\text{-}StCDH}$ in Figure 26 and show that

$$\Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\mathrm{pfs}}] \le q_s \cdot \mathsf{Adv}^{\mathsf{Pw\text{-}GA\text{-}StCDH}}_{\mathsf{EGA}}(\mathcal{B}_3) \ .$$

We cannot achieve a tight bound for $\mathbf{bad}_{\mathrm{pfs}}$, as an adversary against $\mathsf{Pw\text{-}GA\text{-}StCDH}$ must commit on ℓ set elements before it receives the challenge password. Thus, adversary $\mathcal{B}_3$ first guesses a send query τ^* which it will use to solve the problem. On a high level, the simulation of G_6 for adversary $\mathcal{A}$ works as follows: on the τ^*-th send query, $\mathcal{B}_3$ will output the ℓ set elements x_i^{P} provided by the assumption and it uses the input to that send query as the commitment y_i^{P}. Then, if $\mathcal{A}$ decides to corrupt the password, it will receive the challenge password and if it then issues the correct query to H, we can solve $\mathsf{Pw\text{-}GA\text{-}StCDH}$, where we use the decision oracle to simulate the other instances.

We will now describe adversary $\mathcal{B}_3$ in more detail. $\mathcal{B}_3$ inputs set elements $(x_0, x_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x})$ and $x_i^{\mathsf{P}} = (p_i \star \tilde{x})$ for $g_0, g_1, p_i \xleftarrow{\$} \mathcal{G}$ and $i \in [\ell]$. It chooses index τ^* uniformly at random from $[q_s]$ and initializes a counter to keep track of the number of send queries issued so far (lines 01, 03). The counter is incremented whenever a send query is made (lines 07, 40, 63). As we do not know whether the τ^*-th send query will be issued for a user or a server instance, $\mathcal{B}_3$ will embed the elements x_i^{P} for all instances, similarly to adversary $\mathcal{B}_2$ in Figure 16. In particular,

$$\begin{aligned} x_i^{\mathsf{U}} &= u_i \star x_i^{\mathsf{P}} = (u_i \cdot p_i \cdot g_0^{-1}) \star x_0 = (u_i \cdot p_i \cdot g_1^{-1}) \star x_1 \\ x_i^{\mathsf{S}} &= s_i \star x_i^{\mathsf{P}} = (s_i \cdot p_i \cdot g_0^{-1}) \star x_0 = (s_i \cdot p_i \cdot g_1^{-1}) \star x_1 \end{aligned}$$

If the password of an instance is not corrupted when the send query is issued, $\mathcal{B}_3$ checks whether this is the τ^*-th query. If this is the case, it marks the trace of this instance as tr^* and outputs x^{U} in SENDRESP or x^{S} in SENDTERMINIT as the commitment y^{P} (lines 48-50, 73-75) to receive the challenge password $\mathsf{pw}_{\mathsf{US}}$. This is the password which $\mathcal{B}_3$ will output when the adversary corrupts this pair (U, S). For all other corrupt queries, it samples a password uniformly at random (line 16). Instances that are not fresh

```
$\mathcal{B}_3^{\{\text{GA-DDH}_{x_0}(x_i^{\mathsf{P}},\cdot,\cdot),\text{GA-DDH}_{x_1}(x_i^{\mathsf{P}},\cdot,\cdot)\}_{i\in[\ell]}}(x_0, x_1, x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}})$
00 $(\mathcal{C}, T, T_{\mathrm{s}}) \coloneqq (\varnothing, \varnothing, \varnothing)$
01 $\tau^* \xleftarrow{\$} [q_s]$
02 $\mathsf{tr}^* \coloneqq \perp$
03 $\mathrm{cnt} \coloneqq 0$
04 $\beta \xleftarrow{\$} \{0, 1\}$
05 $\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
06 Stop.

$\textsc{SendInit}(\mathsf{U}, t, \mathsf{S})$
07 $\mathrm{cnt} \coloneqq \mathrm{cnt} + 1$
08 $\textbf{if } \pi_{\mathsf{U}}^t \neq \perp \textbf{ return } \perp$
09 $(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
10 $x^{\mathsf{U}} \coloneqq (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) \coloneqq (u_1 \star x_1^{\mathsf{P}}, ..., u_\ell \star x_\ell^{\mathsf{P}})$
11 $\pi_{\mathsf{U}}^t \coloneqq ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \perp), \perp, \perp)$
12 $\textbf{return } (\mathsf{U}, x^{\mathsf{U}})$

$\textsc{Corrupt}(\mathsf{U}, \mathsf{S})$
13 $\textbf{if } (\mathsf{U}, \mathsf{S}) \in \mathcal{C} \textbf{ return } \perp$
14 $\mathcal{C} \coloneqq \mathcal{C} \cup \{(\mathsf{U}, \mathsf{S})\}$
15 $\textbf{if } \mathsf{tr}^* \neq (\mathsf{U}, \mathsf{S}, \cdot, \cdot)$
16    $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
17 $\textbf{return } \mathsf{pw}_{\mathsf{US}}$

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z)$
18 $\textbf{if } T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z] = K \neq \perp$
19    $\textbf{return } K$
20 $\textbf{if } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}) \in T_{\mathrm{s}}$
21    $(b_1, ..., b_\ell) \coloneqq \mathsf{pw}$
22    $\textbf{if } T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] = (\mathsf{U}, (u_1, ..., u_\ell), K)$
23       $\textbf{if } \text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{S}}, u_i^{-1} \star z_i) = 1 \ \forall i \in [\ell]$
         $\textbf{and } (\mathsf{U}, \mathsf{S}) \in \mathcal{C} \textbf{ and } \mathsf{pw} \coloneqq \mathsf{pw}_{\mathsf{US}}$
24          $\textbf{if } \mathsf{tr}^* \neq (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}) \textbf{ abort}$
25          Output $(u_1^{-1} \star z_1, ..., u_\ell^{-1} \star z_\ell)$
26    $\textbf{if } T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] = (\mathsf{S}, (s_1, ..., s_\ell), K)$
27       $\textbf{if } \text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{U}}, s_i^{-1} \star z_i) = 1 \ \forall i \in [\ell]$
         $\textbf{and } (\mathsf{U}, \mathsf{S}) \in \mathcal{C} \textbf{ and } \mathsf{pw} \coloneqq \mathsf{pw}_{\mathsf{US}}$
28          $\textbf{if } \mathsf{tr}^* \neq (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}) \textbf{ abort}$
29          Output $(s_1^{-1} \star z_1, ..., s_\ell^{-1} \star z_\ell)$
30 $\textbf{if } \exists (u_1, ..., u_\ell) \text{ s.t. } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, (u_1, ..., u_\ell)) \in T$
31    $(b_1, ..., b_\ell) \coloneqq \mathsf{pw}$
32    $\textbf{if } \text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{S}}, u_i^{-1} \star z_i) = 1 \ \forall i \in [\ell]$
33       $\textbf{return } T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, (u_1, ..., u_\ell)]$
34 $\textbf{else if } \exists (s_1, ..., s_\ell) \text{ s.t. } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, (s_1, ..., s_\ell)) \in T$
35    $(b_1, ..., b_\ell) \coloneqq \mathsf{pw}$
36    $\textbf{if } \text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{U}}, s_i^{-1} \star z_i) = 1 \ \forall i \in [\ell]$
37       $\textbf{return } T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, (s_1, ..., s_\ell)]$
38 $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z] \xleftarrow{\$} \mathcal{K}$
39 $\textbf{return } T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}, z]$

$\textsc{SendResp}(\mathsf{S}, t, \mathsf{U}, x^{\mathsf{U}})$
40 $\mathrm{cnt} \coloneqq \mathrm{cnt} + 1$
41 $\textbf{if } \pi_{\mathsf{S}}^t \neq \perp \textbf{ return } \perp$
42 $(s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
43 $x^{\mathsf{S}} \coloneqq (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) \coloneqq (s_1 \star x_1^{\mathsf{P}}, ..., s_\ell \star x_\ell^{\mathsf{P}})$
44 $\textbf{if } \exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t' \text{ s.t. } \pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
45    $\textbf{return } \perp$
46 $\textbf{if } (\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
47    $\pi_{\mathsf{S}}^t.\mathsf{fr} \coloneqq \textbf{true}$
48    $\textbf{if } \mathrm{cnt} = \tau^*$
49       $\mathsf{tr}^* \coloneqq (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
50       Output $y^{\mathsf{P}} \coloneqq x^{\mathsf{U}}$ to receive $\mathsf{pw}_{\mathsf{US}}$
51    $K \xleftarrow{\$} \mathcal{K}$
52    $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] \coloneqq (\mathsf{S}, (s_1, ..., s_\ell), K)$
53 $\textbf{else}$
54    $\pi_{\mathsf{S}}^t.\mathsf{fr} \coloneqq \textbf{false}$
55    $(b_1, ..., b_\ell) \coloneqq \mathsf{pw}_{\mathsf{US}}$
56    $\textbf{if } \exists z \text{ s.t. } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
      $\textbf{and } \forall i \in [\ell] : \text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{U}}, s_i^{-1} \star z_i) = 1$
57       $K \coloneqq T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z]$
58    $\textbf{else}$
59       $K \xleftarrow{\$} \mathcal{K}$
60       $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, (s_1, ..., s_\ell)] \coloneqq K$
61 $\pi_{\mathsf{S}}^t \coloneqq ((s_1, ..., s_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \textbf{true})$
62 $\textbf{return } (\mathsf{S}, x^{\mathsf{S}})$

$\textsc{SendTermInit}(\mathsf{U}, t, \mathsf{S}, x^{\mathsf{S}})$
63 $\mathrm{cnt} \coloneqq \mathrm{cnt} + 1$
64 $\textbf{if } \pi_{\mathsf{U}}^t \neq ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \perp), \perp, \perp)$
65    $\textbf{return } \perp$
66 $\textbf{if } \exists \mathsf{P} \in \mathcal{U}, t' \text{ s.t. } \pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
67    $\textbf{return } \perp$
68 $\textbf{if } \exists t' \text{ s.t. } \pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
   $\textbf{and } \pi_{\mathsf{S}}^{t'}.\mathsf{fr} = \textbf{true}$
69    $\pi_{\mathsf{U}}^t.\mathsf{fr} \coloneqq \textbf{true}$
70    $(\mathsf{S}, (s_1, ..., s_\ell), K) \coloneqq T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, B]$
71 $\textbf{else if } (\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
72    $\pi_{\mathsf{U}}^t.\mathsf{fr} \coloneqq \textbf{true}$
73    $\textbf{if } \mathrm{cnt} = \tau^*$
74       $\mathsf{tr}^* \coloneqq (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}})$
75       Output $y^{\mathsf{P}} \coloneqq x^{\mathsf{S}}$ to receive $\mathsf{pw}_{\mathsf{US}}$
76    $K \xleftarrow{\$} \mathcal{K}$
77    $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}] \coloneqq (\mathsf{U}, (u_1, ..., u_\ell), K)$
78 $\textbf{else}$
79    $\pi_{\mathsf{U}}^t.\mathsf{fr} \coloneqq \textbf{false}$
80    $(b_1, ..., b_\ell) \coloneqq \mathsf{pw}_{\mathsf{US}}$
81    $\textbf{if } \exists z \text{ s.t. } (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
      $\textbf{and } \forall i \in [\ell] : \text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{S}}, u_i^{-1} \star z_i) = 1$
82       $K \coloneqq T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z]$
83    $\textbf{else}$
84       $K \xleftarrow{\$} \mathcal{K}$
85       $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, (u_1, ..., u_\ell)] \coloneqq K$
86 $\pi_{\mathsf{U}}^t \coloneqq ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}), K, \textbf{true})$
87 $\textbf{return true}$
```

Fig. 26. Adversary $\mathcal{B}_3$ against Pw-GA-StCDH for the proof of Theorem 8. $\mathcal{A}$ has access to oracles $\mathrm{O} \coloneqq \{$Execute, SendInit, SendResp, SendTermInit, Reveal, Corrupt, Test, H$\}$. Oracles Execute, Reveal and Test are defined as in Figure 15. Lines written in blue show how $\mathcal{B}_3$ simulates the game.

will be simulated with the decision oracles as in the simulation of adversary $\mathcal{B}_2$. For server instances (lines 56-57), this means that we check if there already exists an entry in T such that

$$z_i = \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{U}}, x_i^{\mathsf{S}}) = \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{U}}, s_i \star x_i^{\mathsf{P}}) \iff \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{U}}) = s_i^{-1} \star z_i \ .$$

Equivalently, we check if $\mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{S}}) = u_i^{-1} \star z_i$ for user instances (lines 81-82). If there does not exist an entry yet, $\mathcal{B}_3$ adds a special entry to T (lines 60, 85), which contains the secret group elements s_i or u_i so that we can patch the random oracle later (lines 30-37).

Finally, we look at random oracle queries for all fresh instances contained in T_s (lines 20-29). $\mathcal{B}_3$ checks if the provided z is valid using the decision oracles as explained above. Then it checks for event $\mathbf{bad}_{\text{pfs}}$, i.e., if the password was corrupted and it matches the one in the query. If the query now additionally contains the target trace tr^*, we can solve the $\mathsf{Pw\text{-}GA\text{-}StCDH}$ problem by outputting $u_i^{-1} \star z_i$ in case of a user instance or $s_i^{-1} \star z_i$ in case of a server instance. If the trace is not the target trace, $\mathcal{B}_3$ aborts.

This concludes the analysis of $\mathbf{bad}_{\text{pfs}}$. Collecting the bounds and applying Lemma 5 yields the bound in Theorem 8. □

F.2 Perfect Forward Secrecy for $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$

In order to prove forward secrecy of $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$, we need the following interactive problem which is non-tightly implied by the $\mathsf{SqInv\text{-}GA\text{-}StCDH}$ problem.

Definition 21 (Double Password-based $\mathsf{GA\text{-}StCDH}$ ($\mathsf{DPw\text{-}GA\text{-}StCDH}$)). *On input $(x_0, x_1, x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}, \hat{x}_1^{\mathsf{P}}, ..., \hat{x}_\ell^{\mathsf{P}}) = (g_0 \star \tilde{x}, g_1 \star \tilde{x}, p_1 \star \tilde{x}, ..., p_\ell \star \tilde{x}, \hat{p}_1 \star \tilde{x}, ..., \hat{p}_\ell \star \tilde{x})$, the $\mathsf{DPw\text{-}GA\text{-}StCDH}$ problem requires the adversary to commit on $(y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}) \in \mathcal{X}^\ell$ and then compute $z_i = (g_{b_i}^{-1} \cdot p_i) \star y_i^{\mathsf{P}}\ \forall i \in [\ell]$, as well as $\hat{z}_i = (g_{b_i}^{-1} \cdot \hat{p}_i) \star y_i^{\mathsf{P}}\ \forall i \in [\ell]$ after receiving the challenge password $\mathsf{pw} = (b_1, ..., b_\ell) \in \{0,1\}^\ell$. To an effective group action $\mathsf{XXX} \in \{\mathsf{EGA}, \mathsf{REGA}, \mathsf{EGAT}, \mathsf{REGAT}\}$, we define the advantage function of $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathsf{XXX}}^{\mathsf{DPw\text{-}GA\text{-}StCDH}}(\mathcal{A}) := \Pr\left[\begin{array}{l|c} & (g_0, g_1, p_1, ..., p_\ell, \hat{p}_1, ..., \hat{p}_\ell) \xleftarrow{\$} \mathcal{G}^{2\ell+2} \\ & (x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x}) \\ z_i = \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{P}}, y_i^{\mathsf{P}})\ \forall i \in [\ell] & (x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}) := (p_1 \star \tilde{x}, ..., p_\ell \star \tilde{x}) \\ \hat{z}_i = \mathsf{GA\text{-}CDH}_{x_{b_i}}(\hat{x}_i^{\mathsf{P}}, y_i^{\mathsf{P}})\ \forall i \in [\ell] & (\hat{x}_1^{\mathsf{P}}, ..., \hat{x}_\ell^{\mathsf{P}}) := (\hat{p}_1 \star \tilde{x}, ..., \hat{p}_\ell \star \tilde{x}) \\ & (y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}) \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1, x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}, \hat{x}_1^{\mathsf{P}}, ..., \hat{x}_\ell^{\mathsf{P}}) \\ & \mathsf{pw} := (b_1, ..., b_\ell) \xleftarrow{\$} \mathcal{PW} \\ & (z_1, ..., z_\ell, \hat{z}_1, ..., \hat{z}_\ell) \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pw}) \end{array}\right],$$

where $\mathrm{O} = \{\mathrm{GA\text{-}DDH}_{x_j}(x_i^{\mathsf{P}}, \cdot, \cdot), \mathrm{GA\text{-}DDH}_{x_j}(\hat{x}_i^{\mathsf{P}}, \cdot, \cdot)\}_{i \in [\ell], j \in \{0,1\}}$.

Lemma 6. *The square-inverse* $\mathsf{GA\text{-}StCDH}$ ($\mathsf{SqInv\text{-}GA\text{-}StCDH}$) *implies the double password-based* $\mathsf{GA\text{-}StCDH}$ ($\mathsf{DPw\text{-}GA\text{-}StCDH}$), *more precisely*

$$\mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{DPw\text{-}GA\text{-}StCDH}}(\mathcal{A}) \leq \sqrt{\mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{SqInv\text{-}GA\text{-}StCDH}}(\mathcal{B})} + \frac{1}{|\mathcal{PW}|}\ .$$

Proof. Instead of considering the $\mathsf{SqInv\text{-}GA\text{-}StCDH}$ problem, we will show that the $\mathsf{DSim\text{-}GA\text{-}StCDH}$ problem implies the $\mathsf{DPw\text{-}GA\text{-}StCDH}$ problem and then appy Lemma 1. The proof is similar to the one of Lemma 5. We apply the reset lemma (Lemma 2), where $H = \mathcal{PW}$.

Let $\mathcal{A}$ be an adversary against the $\mathsf{DPw\text{-}GA\text{-}StCDH}$ problem. Consider adversary $\mathcal{B}$ in Figure 27 that takes as input four set elements (x_0, x_1, w_0, w_1) and a password pw. It also has access to decision oracles $\mathrm{GA\text{-}DDH}_{x_j}(w_i, \cdot, \cdot)$ with $i, j \in \{0,1\}$. First, $\mathcal{B}$ generates $(p_1, ...p_\ell, \hat{p}_1, ..., \hat{p}_\ell) \xleftarrow{\$} \mathcal{G}^{2\ell}$ and computes $x_i^{\mathsf{P}} = p_i \star w_0$ and $\hat{x}_i^{\mathsf{P}} = \hat{p}_i \star w_1$ for all $i \in [\ell]$. Then it runs $\mathcal{A}$ on input $(x_0, x_1, x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}, \hat{x}_1^{\mathsf{P}}, ..., \hat{x}_\ell^{\mathsf{P}})$ and receives a commitment $(y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}})$. Now $\mathcal{B}$ sends pw to $\mathcal{A}$ and $\mathcal{A}$ will finally output $(z_1, ..., z_\ell, \hat{z}_1, ..., \hat{z}_\ell)$. $\mathcal{B}$ checks if the solution is correct using the decision oracles and if this is the case, it outputs $b = 1$ and $\sigma = (\mathsf{pw}, p_1, ..., p_\ell, \hat{p}_1, ..., \hat{p}_\ell, y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z_1, ..., z_\ell, \hat{z}_1, ..., \hat{z}_\ell)$. Otherwise it outputs $(0, \epsilon)$.

If $\mathcal{A}$ issues a query to a decision oracle $\mathrm{GA\text{-}DDH}_{x_j}(x_i^{\mathsf{P}}, y', z')$ for some $i \in [\ell]$ and $j \in \{0,1\}$, $\mathcal{B}$ queries its own decision oracle $\mathrm{GA\text{-}DDH}_{x_j}(w_0, y', p_i^{-1} \star z')$ and forwards the output to $\mathcal{A}$. Analogously, if $\mathcal{A}$ issues a query to a decision oracle $\mathrm{GA\text{-}DDH}_{x_j}(\hat{x}_i^{\mathsf{P}}, y', z')$, $\mathcal{B}$ queries $\mathrm{GA\text{-}DDH}_{x_j}(w_1, y', \hat{p}_i^{-1} \star z')$.

Let IG be the algorithm that chooses $g_0, g_1, h_0, h_1 \xleftarrow{\$} \mathcal{G}$ and outputs

$$(x_0, x_1, w_0, w_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x}, h_0 \star \tilde{x}, h_1 \star \tilde{x}).$$

Let acc be defined as in Lemma 2, thus

$$\mathrm{acc} \geq \mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{DPw\text{-}GA\text{-}StCDH}}(\mathcal{A})\ .$$

Let $\mathcal{R}_{\mathcal{B}}$ be the forking algorithm associated to $\mathcal{B}$ as in Lemma 2 with access to the same decision oracles as $\mathcal{B}$. Then we construct adversary $\mathcal{C}$ against $\mathsf{DSim\text{-}GA\text{-}StCDH}$ as in Figure 28. $\mathcal{C}$ runs the reset algorithm

```
$\mathcal{B}^{\{\text{GA-DDH}_{x_j}(w_i,\cdot,\cdot)\}_{i,j\in\{0,1\}}}(x_0, x_1, w_0, w_1, \mathsf{pw})$
$(p_1, ..., p_\ell, \hat{p}_1, ..., \hat{p}_\ell) \xleftarrow{\$} \mathcal{G}^{2\ell}$
$(x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}) := (p_1 \star x, ..., p_\ell \star w_0)$
$(\hat{x}_1^{\mathsf{P}}, ..., \hat{x}_\ell^{\mathsf{P}}) := (\hat{p}_1 \star x, ..., \hat{p}_\ell \star w_1)$
$(y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}) \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1, x_1^{\mathsf{P}}, ...x_\ell^{\mathsf{P}}, \hat{x}_1^{\mathsf{P}}, ..., \hat{x}_\ell^{\mathsf{P}})$
$(z_1, ..., z_\ell, \hat{z}_1, ..., \hat{z}_\ell) \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pw})$
$(b_1, ..., b_\ell) := \mathsf{pw}$
if $\text{GA-DDH}_{x_{b_i}}(w_0, y_i^{\mathsf{P}}, p_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
and $\text{GA-DDH}_{x_{b_i}}(w_1, y_i^{\mathsf{P}}, \hat{p}_i^{-1} \star \hat{z}_i) = 1\ \forall i \in [\ell]$
    return $(1, (\mathsf{pw}, p_1, ..., p_\ell, \hat{p}_1, ..., \hat{p}_\ell, y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z_1, ..., z_\ell, \hat{z}_1, ..., \hat{z}_\ell))$
return $(0, \epsilon)$

$\{\text{GA-DDH}_{x_j}(x_i^{\mathsf{P}}, y', z')\}_{i\in[\ell],j\in\{0,1\}}$
return $\text{GA-DDH}_{x_j}(w_0, y', p_i^{-1} \star z')$

$\{\text{GA-DDH}_{x_j}(\hat{x}_i^{\mathsf{P}}, y', z')\}_{i\in[\ell],j\in\{0,1\}}$
return $\text{GA-DDH}_{x_j}(w_1, y', \hat{p}_i^{-1} \star z')$
```

Fig. 27. Adversary $\mathcal{B}$ for the proof of Lemma 6. Adversary $\mathcal{A}$ has access to oracles $\mathrm{O} = \{\text{GA-DDH}_{x_j}(x_i^{\mathsf{P}}, \cdot, \cdot), \text{GA-DDH}_{x_j}(\hat{x}_i^{\mathsf{P}}, \cdot, \cdot)\}_{i\in[\ell],j\in\{0,1\}}$.

```
$\mathcal{C}^{\{\text{GA-DDH}_{x_j}(w_i,\cdot,\cdot)\}_{i,j\in\{0,1\}}}(x_0, x_1, w_0, w_1)$
$(b, \sigma, \sigma') \leftarrow \mathcal{R}_{\mathcal{B}}^{\mathrm{O}}(x_0, x_1, w_0, w_1)$
if $b = 0$
    abort and return $\perp$
$(\mathsf{pw}, p_1, ..., p_\ell, \hat{p}_1, ..., \hat{p}_\ell, y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z_1, ..., z_\ell, \hat{z}_1, ..., \hat{z}_\ell) := \sigma$
$(\mathsf{pw}', p_1, ..., p_\ell, \hat{p}_1, ..., \hat{p}_\ell, y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z_1', ..., z_\ell', \hat{z}_1', ..., \hat{z}_\ell') := \sigma'$
$(b_1, ..., b_\ell) := \mathsf{pw}$
$(b_1', ..., b_\ell') := \mathsf{pw}'$
Find $i$ such that $b_i \neq b_i'$
return $(y_i^{\mathsf{P}}, p_i^{-1} \star z_i, \hat{p}_i^{-1} \star \hat{z}_i, p_i^{-1} \star z_i', \hat{p}_i^{-1} \star \hat{z}_i')$
```

Fig. 28. Adversary $\mathcal{C}$ against $\mathsf{DSim\text{-}GA\text{-}StCDH}$ for the proof of Lemma 6. $\mathcal{R}_{\mathcal{B}}$ has access to oracles $\mathrm{O} = \{\text{GA-DDH}_{x_j}(w_i, \cdot, \cdot)\}_{i,j\in\{0,1\}}$.

to obtain a bit b^* as well as the two side outputs σ, σ'. $\mathcal{C}$ looks for the first index i where pw and pw' differ and outputs the solution $(y, y_0, y_1, y_2, y_3) = (y_i^{\mathsf{P}}, p_i^{-1} \star z_i, \hat{p}_i^{-1} \star \hat{z}_i, p_i^{-1} \star z_i', \hat{p}_i^{-1} \star \hat{z}_i')$.

To see that the latter solves the $\mathsf{DSim\text{-}GA\text{-}StCDH}$ problem, note that $p_i^{-1} \star x_i^{\mathsf{P}} = w_0$. This implies,

$$\mathsf{GA\text{-}CDH}_{x_{b_i}}(w_0, y_i^{\mathsf{P}}) = p_i^{-1} \star \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{P}}, y_i^{\mathsf{P}}) = p_i^{-1} \star z_i.$$

And similarly $\mathsf{GA\text{-}CDH}_{x_{b_i}}(w_1, y_i^{\mathsf{P}}) = \hat{p}_i^{-1} \star \hat{z}_i$.

Applying Lemmata 1 and 2, we get the bound stated in Lemma 6. □

Theorem 9 (Perfect Forward Secrecy of $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$). *For any adversary $\mathcal{A}$ against $\mathsf{X\text{-}GA\text{-}PAKE}_\ell$ that issues at most q_e execute queries and q_s send queries and where H is modeled as a random oracle, there exist adversary $\mathcal{B}_1$ against $\mathsf{GA\text{-}StCDH}$, and adversaries $\mathcal{B}_2$, $\mathcal{B}_3$ against $\mathsf{SqInv\text{-}GA\text{-}StCDH}$ such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{pfs}}_{\mathsf{X\text{-}GA\text{-}PAKE}_\ell}(\mathcal{A}) \leq\ & \mathsf{Adv}^{\mathsf{GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{SqInv\text{-}GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{B}_2) \\ & + q_s \cdot \sqrt{\mathsf{Adv}^{\mathsf{SqInv\text{-}GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{B}_3)} + \frac{2q_s}{|\mathcal{PW}|} + \frac{(q_s + q_e)^2}{|\mathcal{G}|^{2\ell}} \quad .\end{aligned}$$

Proof. The proof follows the one of Theorem 1 very closely and we make the same adaptions as for the proof of Theorem 8. In particular, we also keep games G_0-G_6 from Figures 5, 7 and 8 with the same changes as described in the proof of Theorem 8, until we get

$$\Pr[\mathsf{G}_5 \Rightarrow \mathbf{bad}] \leq \Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\mathsf{pw}}] + \Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\mathrm{guess}}] + \Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\mathrm{pfs}}] \ ,$$

where it remains to bound $\mathbf{bad}_{\mathrm{pfs}}$. For this purpose, we construct an adversary $\mathcal{B}_3$ against $\mathsf{DPw\text{-}GA\text{-}StCDH}$ in Figure 29 and show that

$$\Pr[\mathsf{G}_6 \Rightarrow \mathbf{bad}_{\mathrm{pfs}}] \leq q_s \cdot \mathsf{Adv}^{\mathsf{DPw\text{-}GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{B}_3) \ .$$

As in the proof of Theorem 8, we cannot achieve a tight bound for $\mathbf{bad}_{\mathrm{pfs}}$. The adversary against $\mathsf{DPw\text{-}GA\text{-}StCDH}$ must commit on ℓ set elements before it receives the challenge password. Thus, we

```
$\mathcal{B}_3^{\{\text{GA-DDH}_{x_b}(y,\cdot,\cdot)\}_{b\in\{0,1\},y\in\{x_i^{\mathsf{P}},\hat{x}_i^{\mathsf{P}}\},i\in[\ell]}}(x_0, x_1, x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}, \hat{x}_1^{\mathsf{P}}, ..., \hat{x}_\ell^{\mathsf{P}})$
00 $(\mathcal{C}, T, T_{\mathrm{s}}) := (\varnothing, \varnothing, \varnothing)$
01 $\tau^* \xleftarrow{\$} [q_s]$
02 $\mathsf{tr}^* := \bot$
03 $\mathrm{cnt} := 0$
04 $\beta \xleftarrow{\$} \{0,1\}$
05 $\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
06 Stop.

SendInit$(\mathsf{U}, t, \mathsf{S})$
07 $\mathrm{cnt} := \mathrm{cnt} + 1$
08 if $\pi_{\mathsf{U}}^t \neq \bot$ return $\bot$
09 $u := (u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
10 $\hat{u} := (\hat{u}_1, ..., \hat{u}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
11 $x^{\mathsf{U}} := (x_1^{\mathsf{U}}, ..., x_\ell^{\mathsf{U}}) := (u_1 \star x_1^{\mathsf{P}}, ..., u_\ell \star x_\ell^{\mathsf{P}})$
12 $\hat{x}^{\mathsf{U}} := (\hat{x}_1^{\mathsf{U}}, ..., \hat{x}_\ell^{\mathsf{U}}) := (\hat{u}_1 \star \hat{x}_1^{\mathsf{P}}, ..., \hat{u}_\ell \star \hat{x}_\ell^{\mathsf{P}})$
13 $\pi_{\mathsf{U}}^t := ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \bot, \bot), \bot, \bot)$
14 return $(\mathsf{U}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}})$

Corrupt$(\mathsf{U}, \mathsf{S})$
15 if $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$ return $\bot$
16 $\mathcal{C} := \mathcal{C} \cup \{(\mathsf{U}, \mathsf{S})\}$
17 if $\mathsf{tr}^* \neq (\mathsf{U}, \mathsf{S}, \cdot, \cdot)$
18     $\mathsf{pw}_{\mathsf{US}} \xleftarrow{\$} \mathcal{PW}$
19 return $\mathsf{pw}_{\mathsf{US}}$

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z)$
20 if $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z] = K \neq \bot$
21     return $K$
22 if $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}) \in T_{\mathrm{s}}$
23     $(b_1, ..., b_\ell) := \mathsf{pw}$
24     if $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] = (\mathsf{U}, (u, \hat{u}), K)$
25         if $\text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{S}}, u_i^{-1} \star z_{i,1}) = 1 \ \forall i \in [\ell]$
           and $\text{GA-DDH}_{x_{b_i}}(\hat{x}_i^{\mathsf{P}}, x_i^{\mathsf{S}}, \hat{u}_i^{-1} \star z_{i,2}) = 1 \ \forall i \in [\ell]$
           and $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$ and $\mathsf{pw} := \mathsf{pw}_{\mathsf{US}}$
26             if $\mathsf{tr}^* \neq (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$ abort
27             Output $(u_1^{-1} \star z_{1,1}, ..., u_\ell^{-1} \star z_{\ell,1}, \hat{u}_1^{-1} \star z_{1,2}, ..., \hat{u}_\ell^{-1} \star z_{\ell,2})$
28     if $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] = (\mathsf{S}, (s, \hat{s}), K)$
29         if $\text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{U}}, s_i^{-1} \star z_{i,1}) = 1 \ \forall i \in [\ell]$
           and $\text{GA-DDH}_{x_{b_i}}(\hat{x}_i^{\mathsf{P}}, x_i^{\mathsf{U}}, \hat{s}_i^{-1} \star z_{i,3}) = 1 \ \forall i \in [\ell]$
           and $(\mathsf{U}, \mathsf{S}) \in \mathcal{C}$ and $\mathsf{pw} := \mathsf{pw}_{\mathsf{US}}$
30             if $\mathsf{tr}^* \neq (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$ abort
31             Output $(s_1^{-1} \star z_{1,1}, ..., s_\ell^{-1} \star z_{\ell,1}, \hat{s}_1^{-1} \star z_{1,3}, ..., \hat{s}_\ell^{-1} \star z_{\ell,3})$
32 if $\exists (u, \hat{u})$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, (u, \hat{u})) \in T$
33     $(b_1, ..., b_\ell) := \mathsf{pw}$
34     if $\text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{S}}, u_i^{-1} \star z_{i,1}) = 1 \ \forall i \in [\ell]$
       and $\text{GA-DDH}_{x_{b_i}}(\hat{x}_i^{\mathsf{P}}, x_i^{\mathsf{S}}, \hat{u}_i^{-1} \star z_{i,2}) = 1 \ \forall i \in [\ell]$
35         return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, (u, \hat{u})]$
36 else if $\exists (s, \hat{s})$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, (s, \hat{s})) \in T$
37     $(b_1, ..., b_\ell) := \mathsf{pw}$
38     if $\text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{U}}, s_i^{-1} \star z_{i,1}) = 1 \ \forall i \in [\ell]$
       and $\text{GA-DDH}_{x_{b_i}}(\hat{x}_i^{\mathsf{P}}, x_i^{\mathsf{U}}, \hat{s}_i^{-1} \star z_{i,3}) = 1 \ \forall i \in [\ell]$
39         return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, (s, \hat{s})]$
40 $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z] \xleftarrow{\$} \mathcal{K}$
41 return $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}, z]$

SendResp$(\mathsf{S}, t, \mathsf{U}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}})$
42 $\mathrm{cnt} := \mathrm{cnt} + 1$
43 if $\pi_{\mathsf{S}}^t \neq \bot$ return $\bot$
44 $s := (s_1, ..., s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
45 $\hat{s} := (\hat{s}_1, ..., \hat{s}_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
46 $x^{\mathsf{S}} := (x_1^{\mathsf{S}}, ..., x_\ell^{\mathsf{S}}) := (s_1 \star x_1^{\mathsf{P}}, ..., s_\ell \star x_\ell^{\mathsf{P}})$
47 $\hat{x}^{\mathsf{S}} := (\hat{x}_1^{\mathsf{S}}, ..., \hat{x}_\ell^{\mathsf{S}}) := (\hat{s}_1 \star \hat{x}_1^{\mathsf{P}}, ..., \hat{s}_\ell \star \hat{x}_\ell^{\mathsf{P}})$
48 if $\exists P \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
49     return $\bot$
50 if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
51     $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ true
52     if $\mathrm{cnt} = \tau^*$
53         $\mathsf{tr}^* := (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
54         Output $y^{\mathsf{P}} := x^{\mathsf{U}}$ to receive $\mathsf{pw}_{\mathsf{US}}$
55     $K \xleftarrow{\$} \mathcal{K}$
56     $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] := (\mathsf{S}, (s, \hat{s}), K)$
57 else
58     $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ false
59     $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
60     if $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
       and $\forall i \in [\ell] : \text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{U}}, s_i^{-1} \star z_{i,1}) = 1$
       and $\forall i \in [\ell] : \text{GA-DDH}_{x_{b_i}}(\hat{x}_i^{\mathsf{P}}, x_i^{\mathsf{U}}, \hat{s}_i^{-1} \star z_{i,3}) = 1$
61         $K := T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z]$
62     else
63         $K \xleftarrow{\$} \mathcal{K}$
64         $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, (s, \hat{s})] := K$
65 $\pi_{\mathsf{S}}^t := ((s, \hat{s}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
66 return $(\mathsf{S}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$

SendTermInit$(\mathsf{U}, t, \mathsf{S}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
67 $\mathrm{cnt} := \mathrm{cnt} + 1$
68 if $\pi_{\mathsf{U}}^t \neq ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, \bot, \bot), \bot, \bot)$
69     return $\bot$
70 if $\exists \mathsf{P} \in \mathcal{U}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
71     return $\bot$
72 if $\exists t'$ s. t. $\pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
   and $\pi_{\mathsf{S}}^{t'}.\mathsf{fr} =$ true
73     $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ true
74     $(\mathsf{S}, (s, \hat{s}), K) := T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}]$
75 else if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
76     $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ true
77     if $\mathrm{cnt} = \tau^*$
78         $\mathsf{tr}^* := (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}})$
79         Output $y^{\mathsf{P}} := x^{\mathsf{S}}$ to receive $\mathsf{pw}_{\mathsf{US}}$
80     $K \xleftarrow{\$} \mathcal{K}$
81     $T_{\mathrm{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}] := (\mathsf{U}, (u, \hat{u}), K)$
82 else
83     $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ false
84     $(b_1, ..., b_\ell) := \mathsf{pw}_{\mathsf{US}}$
85     if $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z) \in T$
       and $\forall i \in [\ell] : \text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{S}}, u_i^{-1} \star z_{i,1}) = 1$
       and $\forall i \in [\ell] : \text{GA-DDH}_{x_{b_i}}(\hat{x}_i^{\mathsf{P}}, x_i^{\mathsf{S}}, \hat{u}_i^{-1} \star z_{i,2}) = 1$
86         $K := T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, z]$
87     else
88         $K \xleftarrow{\$} \mathcal{K}$
89         $T[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}, \mathsf{pw}_{\mathsf{US}}, (u, \hat{u})] := K$
90 $\pi_{\mathsf{U}}^t := ((u, \hat{u}), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \hat{x}^{\mathsf{U}}, x^{\mathsf{S}}, \hat{x}^{\mathsf{S}}), K, \mathbf{true})$
91 return true
```

Fig. 29. Adversary $\mathcal{B}_3$ against DPw-GA-StCDH for the proof of Theorem 9. $\mathcal{A}$ has access to oracles $\mathrm{O} :=$ {Execute, SendInit, SendResp, SendTermInit, Reveal, Corrupt, Test, H}. Oracles Execute, Reveal and Test are defined as in Figure 8. Lines written in blue show how $\mathcal{B}_3$ simulates the game.

apply the same guessing and simulation strategy. Due to the similarities to Theorem 8, we will only briefly describe $\mathcal{B}_3$.

It inputs set elements $(x_0, x_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x})$ and $x_i^{\mathsf{P}} = (p_i \star \tilde{x})$, $\hat{x}_i^{\mathsf{P}} = (\hat{p}_i \star \tilde{x})$ for $g_0, g_1, p_i, \hat{p}_i \xleftarrow{\$} \mathcal{G}$ and $i \in [\ell]$. It embeds the elements x_i^{P}, $\hat{x}_i^{\mathsf{P}}$ in all instances queried to SendInit and SendResp.

If the password of an instance is not corrupted when one of SendResp or SendTermInit is queried, $\mathcal{B}_3$ checks whether this is the τ^*-th query and in this case outputs x^{U} in SendResp or x^{S} in SendTermInit as the commitment y^{P} to receive the challenge password $\mathsf{pw}_{\mathsf{US}}$. This is the password $\mathcal{B}_3$ will output when the adversary corrupts this pair of user and server. Instances that are not fresh will be simulated with the decision oracles.

Finally, we look at random oracle queries for all fresh instances contained in T_{s}. $\mathcal{B}_3$ checks if the provided z is valid using the decision oracles. Then it checks for event $\mathbf{bad}_{\mathrm{pfs}}$, i.e., if the password was corrupted and matches the one in the query. If the query now additionally contains the target trace, we can solve the DPw-GA-StCDH problem. If the trace is not the target trace, $\mathcal{B}_3$ aborts. This concludes the analysis of $\mathbf{bad}_{\mathrm{pfs}}$. Collecting the bounds and applying Lemma 6 yields the bound in Theorem 9.

□

F.3 Perfect Forward Secrecy of Com-GA-PAKE$_\ell$

In order to prove forward secrecy of Com-GA-PAKE$_\ell$, we need the following interactive problem which is non-tightly implied by the GA-GapCDH problem.

Definition 22 (Interactive Password-based GA-StCDH (IPw-GA-StCDH)). *On input $(x_0, x_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x})$, the IPw-GA-StCDH problem requires the adversary to commit on $(y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}) \in \mathcal{X}^\ell$. Then it receives $(x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}) = (p_1 \star \tilde{x}, ..., p_\ell \star \tilde{x})$ and the challenge password $\mathsf{pw} = (b_1, ..., b_\ell) \in \{0,1\}^\ell$ and must compute $z_i = (g_{b_i}^{-1} \cdot p_i) \star y_i^{\mathsf{P}}\ \forall i \in [\ell]$. To an effective group action $\mathsf{XXX} \in \{\mathsf{EGA}, \mathsf{REGA}, \mathsf{EGAT}, \mathsf{REGAT}\}$, we associate the advantage function of $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathsf{XXX}}^{\mathsf{IPw\text{-}GA\text{-}StCDH}}(\mathcal{A}) := \Pr\left[z_i = \mathsf{GA\text{-}CDH}_{x_{b_i}}(x_i^{\mathsf{P}}, y_i^{\mathsf{P}})\ \forall i \in [\ell] \;\middle|\; \begin{array}{c} (g_0, g_1, p_1, ..., p_\ell) \xleftarrow{\$} \mathcal{G}^{\ell+2} \\ (x_0, x_1) := (g_0 \star \tilde{x}, g_1 \star \tilde{x}) \\ (y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}) \leftarrow \mathcal{A}^{\mathrm{O}_1}(x_0, x_1) \\ (x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}) := (p_1 \star \tilde{x}, ..., p_\ell \star \tilde{x}) \\ \mathsf{pw} := (b_1, ..., b_\ell) \xleftarrow{\$} \mathcal{PW} \\ (z_1, ..., z_\ell) \leftarrow \mathcal{A}^{\mathrm{O}_1,\mathrm{O}_2}(x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}, \mathsf{pw}) \end{array} \right],$$

where $\mathrm{O}_1 = \{\mathrm{GA\text{-}DDH}_{x_j}(\tilde{x}, \cdot, \cdot)\}_{j \in \{0,1\}}$ and $\mathrm{O}_2 = \{\mathrm{GA\text{-}DDH}_{x_j}(x_i^{\mathsf{P}}, \cdot, \cdot)\}_{i \in [\ell], j \in \{0,1\}}$.

Lemma 7. *The group action gap computational Diffie-Hellman problem (GA-GapCDH) implies the interactive password-based GA-StCDH (IPw-GA-StCDH), more precisely*

$$\mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{IPw\text{-}GA\text{-}StCDH}}(\mathcal{A}) \leq \sqrt{\mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{GA\text{-}GapCDH}}(\mathcal{B})} + \frac{1}{|\mathcal{PW}|}\ .$$

Instead of proving that the ISim-GA-StCDH problem implies the password-based version IPw-GA-StCDH, we directly use the GA-GapCDH problem to achieve a better bound. The proof uses techniques that are also used in the proof of Lemma 4.

Proof. We use the reset lemma (see Lemma 2) with $H = \mathcal{X}^\ell \times \mathcal{PW}$. Let $\mathcal{A}$ be an adversary against IPw-GA-StCDH. Consider adversary $\mathcal{B}$ against GA-GapCDH in Figure 30 that takes input (x_0, x_1) as well as $(x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}, \mathsf{pw})$. It also has access to a gap oracle GA-DDH$_*$. First, $\mathcal{B}$ runs $\mathcal{A}$ on (x_0, x_1) to receive a commitment $(y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}})$. Now $\mathcal{B}$ sends $(x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}, \mathsf{pw})$ to $\mathcal{A}$ and $\mathcal{A}$ will finally output $(z_1, ..., z_\ell)$. $\mathcal{B}$ checks if the solution is correct using the decision oracle and if this is the case, it outputs $b = 1$ and $\sigma = (y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z_1, ..., z_\ell)$ as side output. Otherwise it outputs $(0, \epsilon)$. As $\mathcal{B}$ has access to a full gap oracle, it can forward all queries of $\mathcal{A}$.

Let IG be the algorithm that chooses $g_0, g_1 \xleftarrow{\$} \mathcal{G}$ and outputs $(x_0, x_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x})$. Let acc be defined as in Lemma 2, thus

$$\mathrm{acc} \geq \mathsf{Adv}_{\mathsf{EGAT}}^{\mathsf{IPw\text{-}GA\text{-}StCDH}}(\mathcal{A})\ .$$

Let $\mathcal{R}_{\mathcal{B}}$ be the reset algorithm associated to $\mathcal{B}$ as in Lemma 2 with access to the same decision oracles as $\mathcal{B}$.

```
$\mathcal{B}^{\text{GA-DDH}_*}(x_0, x_1, (x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}, \mathsf{pw}))$
$(y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}) \leftarrow \mathcal{A}^{\text{O}_1}(x_0, x_1)$
$(z_1, ..., z_\ell) \leftarrow \mathcal{A}^{\text{O}_1,\text{O}_2}(x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}, \mathsf{pw})$
$(b_1, ..., b_\ell) := \mathsf{pw}$
if $\text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, y_i^{\mathsf{P}}, z_i) = 1\ \forall i \in [\ell]$
    return $(1, (y_1^{\mathsf{P}}, ...y_\ell^{\mathsf{P}}, z_1, ..., z_\ell))$
return $(0, \epsilon)$
```

```
$\mathcal{C}^{\text{GA-DDH}_*}(x_0, x_1)$
Pick random coins $\rho$ for $\mathcal{B}$
$(p_1, ..., p_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{P}} := (x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}) := (p_1 \star \tilde{x}, ..., p_\ell \star \tilde{x})$
$\mathsf{pw} := (b_1, ..., b_\ell) \xleftarrow{\$} \mathcal{PW}$
$(b, \sigma) \leftarrow \mathcal{B}^{\text{GA-DDH}_*}(x_0, x_1^t, (x^{\mathsf{P}}, \mathsf{pw}); \rho)$
if $b = 0$ return $\perp$
$(y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z_1, ..., z_\ell) := \sigma$
$(\alpha_1, ..., \alpha_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{P}'} := (x_1^{\mathsf{P}'}, ..., x_\ell^{\mathsf{P}'}) := (\alpha_1 \star z_1^t, ..., \alpha_\ell \star z_\ell^t)$
$\mathsf{pw}' := (b'_1, ..., b'_\ell) \xleftarrow{\$} \mathcal{PW}$
$(b', \sigma') \leftarrow \mathcal{B}^{\text{GA-DDH}_*}(x_0, x_1^t, (x^{\mathsf{P}'}, \mathsf{pw}'); \rho)$
if $b = 0$ return $\perp$
$(y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z'_1, ..., z'_\ell) := \sigma'$
Find $i$ such that $b_i \neq b'_i$
if $b_i = 0$ and $b'_i = 1$
    return $\alpha_i^{-1} \cdot p_i \star z'_i$
else
    return $(\alpha_i^{-1} \cdot p_i \star z'_i)^t$
```

Fig. 30. Adversaries $\mathcal{B}$ and $\mathcal{C}$ against GA-GapCDH for the proof of Lemma 7. Adversary $\mathcal{A}$ has access to decision oracles $\text{O}_1 = \{\text{GA-DDH}_{x_j}(\tilde{x}, \cdot, \cdot)\}_{j \in \{0,1\}}$ and $\text{O}_2 = \{\text{GA-DDH}_{x_j}(x_i^{\mathsf{P}}, \cdot, \cdot)\}_{i \in [\ell], j \in \{0,1\}}$, which $\mathcal{B}$ simulates using the gap oracle GA-DDH_*.

We construct an adversary $\mathcal{C}$ against GA-GapCDH (Figure 30), but instead of running the reset algorithm, $\mathcal{C}$ will simulate $\mathcal{R}_{\mathcal{B}}$ running $\mathcal{B}$ directly.

$\mathcal{C}$ inputs (x_0, x_1) and has access to a gap oracle. First, it chooses random coins ρ for $\mathcal{B}$. It also samples a random element from H by first picking $p_i \xleftarrow{\$} \mathcal{G}$ for $i \in [\ell]$ and a password pw and then computing $x_i^{\mathsf{P}} = p_i \star \tilde{x}$. Then it runs $\mathcal{B}$ on $(x_0, x_1^t, (x_1^{\mathsf{P}}, ..., x_\ell^{\mathsf{P}}, \mathsf{pw})$ using random coins ρ. Note that we use the twist of x_1. $\mathcal{B}$ outputs a bit b and side output σ. If $\mathcal{B}$ was successful, i.e., $b = 1$, then $\mathcal{C}$ parses σ as $(y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z_1, ..., z_\ell)$. Otherwise it aborts. Now it runs $\mathcal{B}$ a second time, this time on input $(x_0, x_1^t, (x_1^{\mathsf{P}'}, ..., x_\ell^{\mathsf{P}'}, \mathsf{pw}')$, where $x_i^{\mathsf{P}'} = \alpha_i \star z_i^t$ for $\alpha_i \xleftarrow{\$} \mathcal{G}$ and $i \in [\ell]$ and pw' is a fresh random password, using the same random coins ρ. Note that all x_i^{P} are also uniformly distributed over $\mathcal{X}$. If $\mathcal{B}$ is successful again, it outputs $(1, (y_1^{\mathsf{P}}, ..., y_\ell^{\mathsf{P}}, z'_1, ..., z'_\ell))$, where the first ℓ set elements are the same as in the first run of $\mathcal{B}$ since we run $\mathcal{B}$ on the same random coins. Now $\mathcal{C}$ can solve GA-GapCDH as follows: Let $y_i^{\mathsf{P}} = h_i \star \tilde{x}$ for some $h_i \in \mathcal{G}$. Then we have

$$\alpha_i \star z_i^t = \begin{cases} \alpha_i \cdot g_0 \cdot p_i^{-1} \cdot h_i^{-1} \star \tilde{x} & \text{if } b_i = 0 \\ \alpha_i \cdot g_1^{-1} \cdot p_i^{-1} \cdot h_i^{-1} \star \tilde{x} & \text{if } b_i = 1 \end{cases}$$

and for z'_i it holds that

$$z'_i = \begin{cases} g_0^{-1} \cdot \alpha_i \cdot g_0 \cdot p_i^{-1} \star \tilde{x} & \text{if } b_i = 0, b'_i = 0 \\ g_1 \cdot \alpha_i \cdot g_0 \cdot p_i^{-1} \star \tilde{x} & \text{if } b_i = 0, b'_i = 1 \\ g_0^{-1} \cdot \alpha_i \cdot g_1^{-1} \cdot p_i^{-1} \star \tilde{x} & \text{if } b_i = 1, b'_i = 0 \\ g_1 \cdot \alpha_i \cdot g_1^{-1} \cdot p_i^{-1} \star \tilde{x} & \text{if } b_i = 1, b'_i = 1 \end{cases}$$

where h_i cancels out in all cases. If $\mathsf{pw} \neq \mathsf{pw}'$, then they must differ in at least one bit. Let i be the first index such that $b_i \neq b'_i$. Using the knowledge of p_i and α_i, $\mathcal{C}$ outputs $\alpha^{-1} \cdot p_i \star z'_i = \mathsf{GA\text{-}CDH}(x_0, x_1)$ in case $b_i = 0$ and $b'_i = 1$. Otherwise if $b_i = 1$ and $b'_i = 0$, it outputs $(\alpha_i^{-1} p_i \star z'_i)^t = \mathsf{GA\text{-}CDH}(x_0, x_1)$. □

Theorem 10 (Perfect Forward Secrecy of Com-GA-PAKE$_\ell$). *For any adversary $\mathcal{A}$ against* Com-GA-PAKE$_\ell$ *that issues at most q_e execute queries, q_s send queries and at most q_{G} and q_{H} queries to random oracles* G *and* H, *there exist adversary $\mathcal{B}_1$ against* GA-StCDH *and adversaries $\mathcal{B}_2$, $\mathcal{B}_3$ against* GA-GapCDH *such that*

$$\mathsf{Adv}^{\mathsf{pfs}}_{\mathsf{Com\text{-}GA\text{-}PAKE}_\ell}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{B}_1) + q_s \ell \cdot \sqrt{\mathsf{Adv}^{\mathsf{GA\text{-}GapCDH}}_{\mathsf{EGAT}}(\mathcal{B}_2)} + q_s \cdot \sqrt{\mathsf{Adv}^{\mathsf{GA\text{-}GapCDH}}_{\mathsf{EGAT}}(\mathcal{B}_3)} + \frac{(q_s + q_e)^2}{|\mathcal{G}|^\ell} + \frac{q_{\mathsf{G}} q_s}{|\mathcal{G}|^\ell} + \frac{2 \cdot (q_{\mathsf{G}} + q_s + q_e)^2}{2^\lambda} + \frac{q_s}{|\mathcal{PW}|} \ ,$$

where λ is the output length of G *in bits.*

Proof. The proof follows the one of Theorem 2 very closely and we make the same adaptions as for the proof of Theorem 8. In particular, we also keep games G_0-G_9 from Figures 18, 20 and 21 with the same changes as described in the proof of Theorem 8, until we get

$$\Pr[\mathsf{G}_8 \Rightarrow \mathbf{bad}] \leq \Pr[\mathsf{G}_9 \Rightarrow \mathbf{bad}_{\mathsf{pw}}] + \Pr[\mathsf{G}_9 \Rightarrow \mathbf{bad}_{\text{guess}}] + \Pr[\mathsf{G}_9 \Rightarrow \mathbf{bad}_{\text{pfs}}] \ ,$$

where it remains to bound $\mathbf{bad}_{\text{pfs}}$. For this purpose, we construct an adversary $\mathcal{B}_3$ against $\mathsf{IPw\text{-}GA\text{-}StCDH}$ in Figure 31 and show that

$$\Pr[\mathsf{G}_9 \Rightarrow \mathbf{bad}_{\text{pfs}}] \leq q_s \cdot \mathsf{Adv}^{\mathsf{IPw\text{-}GA\text{-}StCDH}}_{\mathsf{EGAT}}(\mathcal{B}_3) \ .$$

```
$\mathcal{B}_3^{\text{GA-DDH}_{x_0}(\tilde{x},\cdot,\cdot),\text{GA-DDH}_{x_1}(\tilde{x},\cdot,\cdot)}(x_0, x_1)$
$(\mathcal{C}, T_\mathsf{G}, T_\mathsf{H}, T_s) := (\varnothing, \varnothing, \varnothing, \varnothing)$
$\tau^* \xleftarrow{\$} [q_s]$
$\mathsf{tr}^* := \perp$
$\mathsf{cnt} := 0$
$\beta \xleftarrow{\$} \{0,1\}$
$\beta' \leftarrow \mathcal{A}^{\mathrm{O}}(x_0, x_1)$
Stop.

$\textsc{SendInit}(\mathsf{S}, t, \mathsf{U})$
$\mathsf{cnt} := \mathsf{cnt} + 1$
$\mathbf{if}\ \pi_\mathsf{S}^t \neq \perp\ \mathbf{return}\ \perp$
$\mathsf{com} \xleftarrow{\$} \{0,1\}^\lambda$
$\mathbf{if}\ \exists x^\mathsf{S}\ \text{s.t.}\ T_\mathsf{G}[x^\mathsf{S}] = \mathsf{com}$
    $\mathbf{return}\ \perp$
$T_\mathsf{G}[\diamond] := \mathsf{com}$
$\pi_\mathsf{S}^t := (\perp, (\mathsf{U}, \mathsf{S}, \perp, \perp, \mathsf{com}), \perp, \perp)$
$\pi_\mathsf{S}^t.\mathsf{fr} := \mathbf{false}$
$\mathbf{return}\ (\mathsf{S}, \mathsf{com})$

$\textsc{SendResp}(\mathsf{U}, t, \mathsf{S}, \mathsf{com})$
$\mathsf{cnt} := \mathsf{cnt} + 1$
$\mathbf{if}\ \pi_\mathsf{U}^t \neq \perp\ \mathbf{return}\ \perp$
$\mathbf{if}\ \nexists x^\mathsf{S}\ \text{s.t.}\ T_\mathsf{G}[x^\mathsf{S}] = \mathsf{com}$
    $\pi_\mathsf{U}^t.\mathrm{acc} := \mathbf{false}$
$(u_1, ..., u_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^\mathsf{U} := (x_1^\mathsf{U}, ..., x_\ell^\mathsf{U}) := (u_1 \star \tilde{x}, ..., u_\ell \star \tilde{x})$
$\mathbf{if}\ \mathsf{cnt} = \tau^*\ \mathbf{and}\ \pi_\mathsf{U}^t.\mathrm{acc} \neq \mathbf{false}$
    find $(x^\mathsf{S})$ s.t. $T_\mathsf{G}[x^\mathsf{S}] = \mathsf{com}$
    Output $y^\mathsf{P} := x^\mathsf{S}$ to receive $x^\mathsf{P}, \mathsf{pw}_\mathsf{US}$
    // From now on $\mathcal{B}_3$ also has access to
       $\text{GA-DDH}_{x_0}(x_i^\mathsf{P}, \cdot, \cdot), \text{GA-DDH}_{x_1}(x_i^\mathsf{P}, \cdot, \cdot)\ \forall i \in [\ell]$
    $x^\mathsf{U} := x^\mathsf{P}$
    $(u_1, ..., u_\ell) := \perp$
    $\mathsf{tr}^* = (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, \perp, \mathsf{com})$
$\pi_\mathsf{U}^t := ((u_1, ..., u_\ell), (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, \perp, \mathsf{com}), \perp, \perp)$
$\pi_\mathsf{U}^t.\mathsf{fr} := \mathbf{false}$
$\mathbf{return}\ (\mathsf{U}, x^\mathsf{U})$

$\textsc{Corrupt}(\mathsf{U}, \mathsf{S})$
$\mathbf{if}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{return}\ \perp$
$\mathcal{C} := \mathcal{C} \cup \{(\mathsf{U}, \mathsf{S})\}$
$\mathbf{if}\ \mathsf{tr}^* \neq (\mathsf{U}, \mathsf{S}, \cdot, \cdot)$
    $\mathsf{pw}_\mathsf{US} \xleftarrow{\$} \mathcal{PW}$
$\mathbf{return}\ \mathsf{pw}_\mathsf{US}$

$\mathsf{H}(\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z)$
$\mathbf{if}\ T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z] = K \neq \perp$
    $\mathbf{return}\ K$
$(b_1, ..., b_\ell) := \mathsf{pw}$
$\mathbf{if}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}) \in T_\mathrm{s}$
    $\mathbf{if}\ T_\mathrm{s}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}] = (\mathsf{U}, (u_1, ..., u_\ell), K)$
        $\mathbf{if}\ \mathsf{tr}^* = (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com})$
        $\mathbf{and}\ \text{GA-DDH}_{x_{b_i}}(x_i^\mathsf{P}, x_i^\mathsf{S}, z_i) = 1\ \forall i \in [\ell]$
        $\mathbf{and}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{and}\ \mathsf{pw} := \mathsf{pw}_\mathsf{US}$
            Output $(z_1, ..., z_\ell)$
        $\mathbf{if}\ \mathsf{tr}^* = (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com})$
        $\mathbf{and}\ \text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^\mathsf{S}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
        $\mathbf{and}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{and}\ \mathsf{pw} := \mathsf{pw}_\mathsf{US}$
            $\mathbf{abort}$
    $\mathbf{if}\ T_\mathrm{s}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}] = (\mathsf{S}, (s_1, ..., s_\ell), K)$
        $\mathbf{if}\ \mathsf{tr}^* = (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com})$
        $\mathbf{and}\ \text{GA-DDH}_{x_{b_i}}(x_i^\mathsf{P}, x_i^\mathsf{U}, z_i) = 1\ \forall i \in [\ell]$
        $\mathbf{and}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{and}\ \mathsf{pw} := \mathsf{pw}_\mathsf{US}$
            Output $(z_1, ..., z_\ell)$
        $\mathbf{if}\ \text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^\mathsf{U}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
        $\mathbf{and}\ (\mathsf{U}, \mathsf{S}) \in \mathcal{C}\ \mathbf{and}\ \mathsf{pw} := \mathsf{pw}_\mathsf{US}$
            $\mathbf{abort}$
$\mathbf{if}\ \exists (u_1, ..., u_\ell)\ \text{s.t.}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, (u_1, ..., u_\ell)) \in T_\mathsf{H}$
    $\mathbf{if}\ \mathsf{tr}^* = (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com})$
    $\mathbf{and}\ \text{GA-DDH}_{x_{b_i}}(x_i^\mathsf{P}, x_i^\mathsf{S}, z_i) = 1\ \forall i \in [\ell]$
        $\mathbf{return}\ T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, (u_1, ..., u_\ell)]$
    $\mathbf{if}\ \text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^\mathsf{S}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
        $\mathbf{return}\ T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, (u_1, ..., u_\ell)]$
$\mathbf{else\ if}\ \exists (s_1, ..., s_\ell)\ \text{s.t.}\ (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, (s_1, ..., s_\ell)) \in T_\mathsf{H}$
    $\mathbf{if}\ \mathsf{tr}^* = (\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com})$
    $\mathbf{and}\ \text{GA-DDH}_{x_{b_i}}(x_i^\mathsf{P}, x_i^\mathsf{U}, z_i) = 1\ \forall i \in [\ell]$
        $\mathbf{return}\ T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, (s_1, ..., s_\ell)]$
    $\mathbf{if}\ \text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^\mathsf{U}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
        $\mathbf{return}\ T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, (s_1, ..., s_\ell)]$
$T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z] \xleftarrow{\$} \mathcal{K}$
$\mathbf{return}\ T_\mathsf{H}[\mathsf{U}, \mathsf{S}, x^\mathsf{U}, x^\mathsf{S}, \mathsf{com}, \mathsf{pw}, z]$
```

Fig. 31. Adversary $\mathcal{B}_3$ against $\mathsf{IPw\text{-}GA\text{-}StCDH}$ for the proof of Theorem 10. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{$Execute, SendInit, SendResp, SendTermInit, Reveal, Corrupt, Test, $\mathsf{G}, \mathsf{H}\}$. Oracles Execute, Reveal, Test and G are defined as in Figure 21. Oracles SendTermInit and SendTermResp are defined in Figure 32. Lines written in blue show how $\mathcal{B}_3$ simulates the game.

```
$\textsc{SendTermInit}(\mathsf{S}, t, \mathsf{U}, x^{\mathsf{U}})$
$\mathrm{cnt} := \mathrm{cnt} + 1$
if $\pi_{\mathsf{S}}^t \neq (\bot, (\mathsf{U}, \mathsf{S}, \bot, \bot, \mathsf{com}), \bot, \bot)$
    return $\bot$
$(s_1, \ldots, s_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
$x^{\mathsf{S}} := (x_1^{\mathsf{S}}, \ldots, x_\ell^{\mathsf{S}}) := (s_1 \star \tilde{x}, \ldots, s_\ell \star \tilde{x})$
if $\mathrm{cnt} = \tau^*$
    Output $y^{\mathsf{P}} := x^{\mathsf{U}}$ to receive challenge $x^{\mathsf{P}}, \mathsf{pw}_{\mathsf{US}}$
    // From now on $\mathcal{B}_3$ also has access to
       $\text{GA-DDH}_{x_0}(x_i^{\mathsf{P}}, \cdot, \cdot), \text{GA-DDH}_{x_1}(x_i^{\mathsf{P}}, \cdot, \cdot)\ \forall i \in [\ell]$
    $x^{\mathsf{S}} := x^{\mathsf{P}}$
    $(s_1, \ldots, s_\ell) := \bot$
    $\mathsf{tr}^* = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
if $T_{\mathsf{G}}[x^{\mathsf{S}}] \neq \bot$
    return $\bot$
Replace $\diamond$ in $T_{\mathsf{G}}[\diamond] := \mathsf{com}$ with $x^{\mathsf{S}}$
if $\exists \mathsf{P} \in \mathcal{U} \cup \mathcal{S}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
    return $\bot$
if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
    $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ true
    $K \xleftarrow{\$} \mathcal{K}$
    $T_{\mathsf{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}] := (\mathsf{S}, (s_1, \ldots, s_\ell), K)$
else
    $\pi_{\mathsf{S}}^t.\mathsf{fr} :=$ false
    $(b_1, \ldots, b_\ell) := \mathsf{pw}_{\mathsf{US}}$
    if $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}) = \mathsf{tr}^*$
    and $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z) \in T_{\mathsf{H}}$
    and $\text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{U}}, z_i) = 1\ \forall i \in [\ell]$
        $K := T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z]$
    else if $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z) \in T_{\mathsf{H}}$
    and $\text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^{\mathsf{U}}, s_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
        $K := T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z]$
    else
        $K \xleftarrow{\$} \mathcal{K}$
        $T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, (s_1, \ldots, s_\ell)] := K$
$\pi_{\mathsf{S}}^t := ((s_1, \ldots, s_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), K, \textbf{true})$
return $(\mathsf{S}, x^{\mathsf{S}})$
```

```
$\textsc{SendTermResp}(\mathsf{U}, t, \mathsf{S}, x^{\mathsf{S}})$
$\mathrm{cnt} := \mathrm{cnt} + 1$
if $\pi_{\mathsf{U}}^t \neq ((u_1, \ldots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, \bot, \mathsf{com}), \bot, \bot)$
    return $\bot$
if $\mathsf{G}(x^{\mathsf{S}}) \neq \mathsf{com}$
    $\pi_{\mathsf{U}}^t := ((u_1, \ldots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), \bot, \textbf{false})$
    return $\bot$
if $\exists \mathsf{P} \in \mathcal{U}, t'$ s. t. $\pi_{\mathsf{P}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
    return $\bot$
if $\pi_{\mathsf{U}}^t.\mathsf{tr} = \mathsf{tr}^*$
    $\mathsf{tr}^* := (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
if $\exists t'$ s. t. $\pi_{\mathsf{S}}^{t'}.\mathsf{tr} = (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com})$
and $\pi_{\mathsf{S}}^{t'}.\mathsf{fr} =$ true
    $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ true
    $(\mathsf{S}, (s_1, \ldots, s_\ell), K) := T_{\mathsf{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}]$
else if $(\mathsf{U}, \mathsf{S}) \notin \mathcal{C}$
    $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ true
    $K \xleftarrow{\$} \mathcal{K}$
    $T_{\mathsf{s}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}] := (\mathsf{U}, (u_1, \ldots, u_\ell), K)$
else
    $\pi_{\mathsf{U}}^t.\mathsf{fr} :=$ false
    $(b_1, \ldots, b_\ell) := \mathsf{pw}_{\mathsf{US}}$
    if $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}) = \mathsf{tr}^*$
    and $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z) \in T_{\mathsf{H}}$
    and $\text{GA-DDH}_{x_{b_i}}(x_i^{\mathsf{P}}, x_i^{\mathsf{S}}, z_i) = 1\ \forall i \in [\ell]$
        $K := T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z]$
    else if $\exists z$ s. t. $(\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z) \in T_{\mathsf{H}}$
    and $\text{GA-DDH}_{x_{b_i}}(\tilde{x}, x_i^{\mathsf{S}}, u_i^{-1} \star z_i) = 1\ \forall i \in [\ell]$
        $K := T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, z]$
    else
        $K \xleftarrow{\$} \mathcal{K}$
        $T_{\mathsf{H}}[\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}, \mathsf{pw}_{\mathsf{US}}, (u_1, \ldots, u_\ell)] := K$
$\pi_{\mathsf{U}}^t := ((u_1, \ldots, u_\ell), (\mathsf{U}, \mathsf{S}, x^{\mathsf{U}}, x^{\mathsf{S}}, \mathsf{com}), K, \textbf{true})$
return true
```

Fig. 32. Oracles SendTermInit and SendTermResp for adversary $\mathcal{B}_3$ in Figure 31.

We apply the same guessing and simulation strategy. Due to the similarities to Theorem 8 and adversary $\mathcal{B}_2$ in Theorem 2, we will only briefly describe $\mathcal{B}_3$. It inputs set elements $(x_0, x_1) = (g_0 \star \tilde{x}, g_1 \star \tilde{x})$.

In SendResp, we check if this is the τ^*-th query and the commitment sent by $\mathcal{A}$ was output by G before, then $\mathcal{B}_3$ looks in the list T_{G} to find the corresponding input x^{S} and outputs x^{S} as commitment y^{P} to receive the IPw-GA-StCDH challenge x^{P} and $\mathsf{pw}_{\mathsf{US}}$. Queries to SendTermInit are simulated similarly. $\mathcal{B}_2$ computes x^{S} and if this is the τ^*-th query, it outputs x^{U} as commitment y^{P} to receive x^{P} and $\mathsf{pw}_{\mathsf{US}}$. Instances that are not fresh will be simulated with the decision oracles.

Finally, we look at random oracle queries for all fresh instances contained in T_{s}. $\mathcal{B}_3$ checks if the provided z is valid using the decision oracles. Then it checks for event $\mathbf{bad}_{\mathrm{pfs}}$, i.e., if the password was corrupted and it matches the one in the query. If the query now additionally contains the target trace, we can solve the IPw-GA-StCDH problem. If the trace is not the target trace, $\mathcal{B}_3$ aborts. This concludes the analysis of $\mathbf{bad}_{\mathrm{pfs}}$ and applying Lemma 7 yields the bound in Theorem 10. □