Perfect Forward Security of SPAKE2

Michel Abdalla[1,2] and Manuel Barbosa[3]

[1] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
michel.abdalla@ens.fr
[2] INRIA, Paris, France
[3] FCUP and INESC TEC, Porto, Portugal
mbb@fc.up.pt

Abstract. SPAKE2 is a balanced password-authenticated key exchange (PAKE) protocol, proposed by Abdalla and Pointcheval at CTRSA 2005. Due to its simplicity and efficiency, SPAKE2 is one of the balanced PAKE candidates currently under consideration for standardization by the CFRG, together with SPEKE, CPace, and J-PAKE. In this paper, we show that SPAKE2 achieves perfect forward security in the random-oracle model under the Gap Diffie-Hellman assumption. Unlike prior results, which either did not consider forward security or only proved a weak form of it, our results guarantee the security of the derived keys even for sessions that were created with the active involvement of the attacker, as long as the parties involved in the protocol are not corrupted when these sessions take place. Finally, our proofs also demonstrate that SPAKE2 is flexible with respect to the generation of its global parameters M and N. This includes the cases where M is a uniform group element and $M = N$ or the case where M and N are chosen as the output of a random oracle.

1 Introduction . . . 1
2 Preliminaries . . . 2
3 SPAKE2 . . . 3
4 Security Model . . . 3
5 Assumptions . . . 4
6 Perfect Forward Security Proof . . . 6
7 Security with Key Confirmation . . . 12
8 Implications . . . 18
Acknowledgments . . . 18
A Games and adversaries for the proof of Theorems 6.1 and 6.2 . . . 20
B Games and adversary for the proof of Theorem 7.1 . . . 25

1 Introduction

Password-authenticated key exchange (PAKE) allows users to establish session keys among themselves with the help of a short secret, known as a password, which can be drawn from a small set of possible values. Passwords can be represented in shorter human-readable formats and distributed/stored using a wider range of mechanisms, which are important requirements in many applications. One important use-case is when secrets must be memorized by humans.

Since the seminal work by Bellovin and Merritt [BM92], several PAKE protocols have appeared in the literature, achieving different levels of security (such as indistinguishability-based security or universal composability) under a variety of assumptions.

Recently, the Crypto Forum Research Group (CFRG)[4], which is an IRTF (Internet Research Task Force) research group focused on the discussion and review of uses of cryptographic mechanisms, started a PAKE selection process with the goal of providing recommendations for password-based authenticated key establishment in IETF protocols. Originally, 4 candidates were under consideration by the CRFG in the balanced PAKE category: SPEKE [Jab97, Mac01, HS14], SPAKE2 [AP05, LK19], J-PAKE [HR10, ABM15], and CPace [HL18, HL19]. Currently, only SPAKE2 and CPace are still under consideration in the balanced PAKE category.

Security analyses for SPAKE2. In the original security analysis [AP05], Abdalla and Pointcheval proved that SPAKE2 achieves implicit authentication in the indistinguishability-based model by Bellare, Pointcheval, and Rogaway [BPR00]. Their security analysis, however, did not take corruption queries into account, which are needed for proving forward security. To address this shortcoming, Becerra, Ostrev, and Skrobot [BOS18] recently provided a proof of weak forward security for SPAKE. Unlike perfect forward security, weak forward security only guarantees the security of sessions created without an active intervention by the attacker as long as the involved parties remain uncorrupted during the execution of these sessions [KOY03, Kra05].

In addition to proving weak forward security in [BOS18], the authors also provide a proof of perfect forward security for a variant of SPAKE2 which includes explicit authentication and in which one of the flows is not encrypted with the password. The latter result, however, seems to be a particular case of the scheme proven secure by Abdalla et al. in [ABC+06].

Our contributions. In this work, we provide further security analyses for SPAKE2 [AP05, LK19]. More precisely, we demonstrate that SPAKE2 achieves perfect forward security even without key confirmation. Our proof of security is based on the Gap Diffie-Hellman assumption [OP01] in the random-oracle model [BR93]. Note that this does *not* contradict the impossibility result about the perfect forward security of 2-message key-exchange protocols in in the HMQV [Kra05] paper since the SPAKE2 protocol assumes the pre-existence of secure shared state between parties which is the password itself. We also include an analysis of the protocol when key confirmation is added, which yields a better bound.

Finally, our proofs also demonstrate that SPAKE2 is flexible with respect to the generation of its global parameters M and N. This includes the cases where M is a uniform group element and $M = N$ or the case where M and N are chosen as the output of a random oracle.

Difference to previous version. The proof of security of SPAKE2 without key confirmation has been updated to refine the analysis of a bad event that was not fully covered by the initial version. To deal with this, we added an extra game hop, for which we give two alternative justifications. In the first one, which yields a tight reduction to GapSqDH, we assume an algebraic adversary. In the second one, we get a looser reduction to GCDH via the splitting lemma of [PS00, AP05]. We also added the analysis of SPAKE2 with key confirmation.

2 Preliminaries

Notation. We write $x \leftarrow y$ for the action of assigning the value y to the variable x. We write $x_1, \ldots, x_n \leftarrow_{\$} \mathsf{X}$ for sampling $x_1, \ldots, x_n$ from a finite set X uniformly at random. If A is a probabilistic algorithm we write $y_1, \ldots, y_n \leftarrow_{\$} A(x_1, \ldots, x_n)$ for the action of running A on inputs $x_1, \ldots, x_n$ with independently chosen coins, and assigning the result to $y_1, \ldots, y_n$. ppt as usual abbreviates probabilistic polynomial-time. We use notation $T[x]$ to denote access to a dictionary/table T at index x, and $\{\,\}$ to denote the empty table. We abuse notation and use T to also represent the set of assigned indices in table T.

Games. We use the code-based game-playing language [BR04]. Each game has an Initialize and a Finalize procedure. It also has specifications of procedures to respond to an adversary's various

[4] https://irtf.org/cfrg

queries. A game is run with an adversary $\mathcal{A}$ as follows. First INITIALIZE runs and its outputs are passed to $\mathcal{A}$. Then $\mathcal{A}$ runs and its oracle queries are answered by the procedures of the game. When $\mathcal{A}$ terminates, its output is passed to FINALIZE which returns the outcome of the game.

3 SPAKE2

Fig. 1 shows a SPAKE2 protocol execution between a user U and a server S.

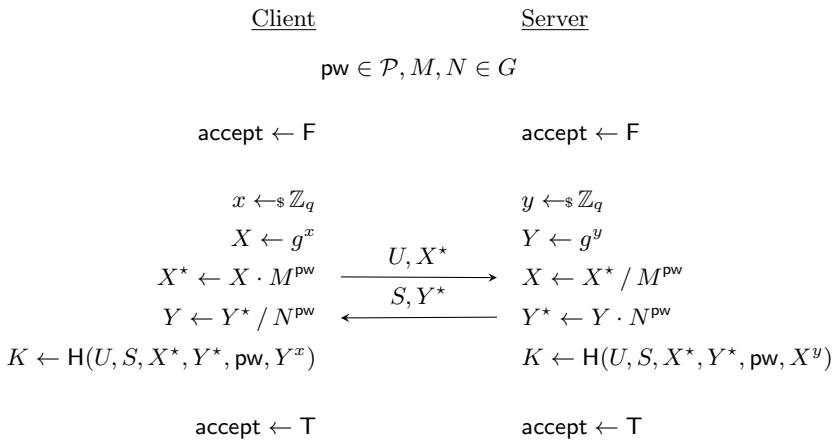

Fig. 1. The SPAKE2 protocol [AP05].

4 Security Model

Our proof of security uses the indistinguishability-based model by Bellare, Pointcheval, and Rogaway [BPR00] and its extension to multiple TEST queries proposed by Abdalla, Fouque, and Pointcheval (AFP) [AFP05, AFP06]. In the latter, all TEST queries are answered with the same challenge bit b.

The security game already instantiated with SPAKE2 can be seen as G_1 in Fig. 3. The name spaces for servers $\mathcal{S}$ and users $\mathcal{U}$ are assumed to be disjoint; oracles reject queries inconsistent with these name spaces.

In the following, we describe more precisely the state of a party instance, the notion of partnering, and the freshness condition used in the proof.

Definition 4.1 (Instance state). *The state of an instance i at party P, denoted π_P^i is a tuple of the form $(\mathsf{e}, \mathsf{tr}, K, \mathsf{ac})$ where*

- e *is the secret exponent (x or y) chosen by the party in that instance*
- tr *is a session trace of the form $(U, S, X^\star, Y^\star)$*
- K *is the accepted session key*
- ac *is a boolean flag that indicates whether the instance accepted ($\mathsf{ac} = \mathsf{T}$) or is expecting a response ($\mathsf{ac} = \mathsf{F}$)*

We will use $\pi_P^i.\mathsf{e}$, $\pi_P^i.\mathsf{tr}$, $\pi_P^i.K$, $\pi_P^i.\mathsf{ac}$ to denote the individual components of the state.

Definition 4.2 (Partnering). *A server instance π_S^i and a user instance π_U^j are partnered iff*

$$\pi_S^i.\mathsf{ac} = \mathsf{T} \wedge \pi_U^j.\mathsf{ac} = \mathsf{T} \wedge \pi_S^i.\mathsf{tr} = \pi_U^j.\mathsf{tr}$$

Two user instances are never partnered. Two server instances are never partnered.

This is a definition in the style of matching-conversations. Note that partnering together with correctness implies that the the associated keys are the same.

Definition 4.3 (Freshness). *Instance i at party P is* fresh, *written* $\mathsf{Fresh}(P, i)$ *if and only if all the following conditions hold:*

1. *the instance accepted;*
2. *the instance was not queried to* Test *or* Reveal *before;*
3. *At least one of the following conditions holds:*
   (a) *The instance accepted during a query to* Execute*;*
   (b) *There exists more than one partner instance;*
   (c) *No partner instance exists and* $\textsc{Corrupt}(U, S)$ *was not called prior to acceptance;*
   (d) *A unique fresh partner instance exists.*

5 Assumptions

The perfect forward security of SPAKE2 is based on the difficulty of solving the Computational Diffie-Hellman (CDH) problem by attackers that have access to a Decisional Diffie-Hellman (DDH) oracle. This is usually known as the Gap Diffie-Hellman problem (GCDH) [OP01].

The proof is carried out in the Random-Oracle (RO) model and, to prove perfect forward secret, we need to restrict our attention to algebraic adversaries [FKL18]. Note that the GCDH problem has been shown to be equivalent to the standard discrete logarithm problem (DL) in the Algebraic and Generic Group models [FKL18, Los19] so, our proof implies reductions to the DL problem in these idealized models of computation.[5]

Our proof is structured to highlight which problem an attacker would need to solve to break the protocol. On the one hand, we give tight reductions to well-known computational assumptions. On the other hand, it follows sequence of hops that clarifies how the assumptions map to the generation of SPAKE2 global parameters and runtime operation. We show that, as expected, passive behavior by the attacker implies that it needs to solve the standard Gap CDH problem to break the established session. Intuitively attackers have access to a DDH oracle by comparing random oracle outputs to the keys revealed by the game. Active attackers that may try to take advantage of leaked passwords must compute the CDH of the global parameters M and N or at least compute the squared version of this problem $\mathsf{CDH}(M, M)$, with the help of a DDH oracle. We show that the protocol is flexible with respect to the generation procedure, provided that the previous problem is hard to solve. This includes, in particular, the case where M is a uniform group element and $M = N$.

To this end, we formalize a family of assumptions parametrized by a distribution $\mathcal{D}$ that outputs a pair of elements in $\mathbb{Z}_q$, where q is a large prime. The assumptions are stated with respect to a group G of order q and generator g.

[5] There is a subtlety here which further justifies our presentation with gap assumptions. Our proof in the algebraic model uses the gap oracles on values queried by attackers to a random oracle. As random oracles are an abstraction of local computation, it is perhaps unreasonable to ask that the algebraic adversary provides us with representations of those values. If this is not required from the adversary, then our proof does not necessarily imply a reduction to DL for algebraic attackers, and we must rely on the gap assumptions. For this reason, in our proof we only make explicit use of the algebraic nature of the adversary on messages that it explicitly delivers.

Definition 5.1 (Gap Computational Diffie-Hellman ($\mathcal{D}$-GCDH)). *The $\mathcal{D}$-GCDH assumption states that, for any ppt adversary $\mathcal{A}$, the following probability is small.*

$$\mathsf{Adv}^{\mathsf{GCDH}}_{\mathcal{A}} := \Pr\left[Z = g^{xy} \ : \ Z \leftarrow_{\$} \mathcal{A}^{\mathsf{DDH}(\cdot,\cdot,\cdot)}(g^x, g^y);\ x \leftarrow_{\$} \mathbb{Z}_q;\ y \leftarrow_{\$} \mathcal{D}(x) \right]$$

The standard GCDH assumption is a particular case when $\mathcal{D}$ is the uniform distribution over $\mathbb{Z}_q$. We denote this by $\mathcal{U}$-GCDH. The Gap Squared Diffie-Hellman (GSqDH) assumption is another particular case, when we restrict the previous case to $\mathcal{D}$ outputting $x = y$. For simplicity, in our analysis, we will restrict our attention to global parameter sampling operations where we exclude the possibility that $x = 0$ or $y = 0$. This means that the bound for our proof includes an extra statistical when we remove this restriction (this will be of the form $(q_s + q_e)/q$ for standard GCDH). We will also rely on the weaker Gap Discrete Logarithm assumption.

Definition 5.2 (Gap Discrete Logarithm (GDL)). *The GDL assumption states that, for any ppt adversary $\mathcal{A}$, the following probability is small.*

$$\mathsf{Adv}^{\mathsf{GDL}}_{\mathcal{A}} := \Pr\left[x' = x \ : \ x' \leftarrow_{\$} \mathcal{A}^{\mathsf{DDH}(\cdot,\cdot,\cdot)}(g^x);\ x \leftarrow_{\$} \mathbb{Z}_q \right]$$

Finally, for the version of our result where we do not rely on the assumption that the adversary is algebraic, we require the following assumption introduced in [AP05], which we extend to the gap version.

Definition 5.3 (Gap Discrete Logarithm Password-based Chosen-basis Computational Diffie-Hellman assumption (GPCCDH)). *The $\mathcal{D}$-GPCCDH assumption states that, for any ppt adversary $\mathcal{A}$, the following probability is small.*

$$\mathsf{Adv}^{\mathcal{D}\text{-}\mathsf{GPCCDH}}_{\mathcal{A}} := \Pr\left[C = \mathsf{CDH}(A_1/B_1^{\mathsf{pw}}, A_2/B_2^{\mathsf{pw}}) \;\middle|\; \begin{array}{l} b_1 \leftarrow_{\$} \mathbb{Z}_q; b_2 \leftarrow_{\$} \mathcal{D}(b_1) \\ A_1 \leftarrow_{\$} G; B_1 \leftarrow g^{b_1}; B_2 \leftarrow g^{b_2} \\ (A_2, \mathsf{st}) \leftarrow \mathcal{A}_2^{\mathsf{GPCCDH}(\cdot,\cdot,\cdot)}(B_1, B_2, A_1) \\ \mathsf{pw} \leftarrow_{\$} \mathcal{P} \\ C \leftarrow \mathcal{A}_2^{\mathsf{GPCCDH}(\cdot,\cdot,\cdot)}(\mathsf{pw}, \mathsf{st}) \end{array} \right]$$

It was shown in [AP05] that the assumption above without the DDH oracle is implied by the standard CDH problem. The proof extends naturally to gap versions of the problems, so we have the following.

Lemma 5.4 ([AP05], Lemma 3.4). *For all GPCCDH adversaries $\mathcal{A}$ there exists an adversary $\mathcal{B}$ such that:*

$$\frac{2}{|\mathcal{P}|} \geq \mathsf{Adv}^{\mathcal{D}\text{-}\mathsf{GPCCDH}}_{\mathcal{A}} \geq \frac{1}{|\mathcal{P}|} + \epsilon \Longrightarrow \mathsf{Adv}^{\mathcal{D}\text{-}\mathsf{GCDH}}_{\mathcal{B}} \geq \frac{|\mathcal{P}|}{128} \cdot \epsilon^3 \,.$$

Here we present the lemma for the version where $1/|\mathcal{P}|$ is large when compared to ϵ, i.e., we are dealing with passwords with significantly lower entropy than the security parameter.

In the proof of SPAKE2 with key confirmation, we will assume that the Key Derivation Function (KDF) is a Pseudorandom-Function (PRF). This assumption is implied by the standard notions of security for KDFs [Kra10].

Definition 5.5 (Pseudorandom Function). *A Key Derivation Function KDF with key space $\mathcal{K}$, domain $\mathcal{D}$ and range $\mathcal{R}$ is PRF-secure if, for any ppt adversary $\mathcal{A}$, the following advantage term is small.*

$$\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A}} := \left| \Pr\left[b = 1 \ : \ b \leftarrow_{\$} \mathcal{A}^{\mathsf{KDF}(K,\cdot)}();\ K \leftarrow_{\$} \mathcal{K} \right] - \Pr\left[b = 1 \ : \ b \leftarrow_{\$} \mathcal{A}^{F(\cdot)}();\ F \leftarrow_{\$} \mathcal{F} \right] \right|$$

Here $\mathcal{F}$ is the set of all functions with domain $\mathcal{D}$ and range $\mathcal{R}$.

In our proof, we actually only require a PRF that is pseudorandom for one input per key, and we will denote this weaker requirement by $\mathsf{1PRF}$.

Finally, in the proof of SPAKE2 with key confirmation, we require the standard unforgeability property for a one-time MAC.

Definition 5.6 (One-time MAC security). *A MAC scheme with key space $\mathcal{K}$ and domain $\mathcal{D}$ is one-time unforgeable if, for any ppt adversary $\mathcal{A}$, the following advantage term is small.*

$$\mathsf{Adv}^{\mathsf{1UF}}_{\mathcal{A}} := \Pr\left[\, t = \mathsf{MAC}(K, m) \;:\; (m, t) \leftarrow_{\$} \mathcal{A}(\,); K \leftarrow_{\$} \mathcal{K} \,\right]$$

6 Perfect Forward Security Proof

Theorem 6.1. *SPAKE2 is tightly PFS-secure in the random-oracle model under the Gap Squared Diffie-Hellman assumption. More precisely, for every algebraic attacker $\mathcal{A}$ against SPAKE2, there exist attackers $\mathcal{B}_1$ and $\mathcal{B}_3$ against the Gap Diffie-Hellman problem, attacker $\mathcal{B}_2$ against the Gap Discrete Logarithm problem, and attacker $\mathcal{B}_4$ against the Gap Squared Diffie-Hellman problem such that*

$$\mathsf{Adv}^{\mathrm{SPAKE2}}_{\mathcal{A}}(\,) \leq \frac{2q_s}{|\mathcal{P}|} + \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_1}(\,) + \mathsf{Adv}^{\mathsf{GDL}}_{\mathcal{B}_2}(\,) + \mathsf{Adv}^{\mathcal{D}^\star\text{-}\mathsf{GCDH}}_{\mathcal{B}_3}(\,) + \mathsf{Adv}^{\mathsf{GSqDH}}_{\mathcal{B}_4}(\,) + \frac{(q_s + q_e)^2}{2q}$$

Here there is an extra $q_s/|\mathcal{P}|$ term that is a consequence of our game hopping strategy, which has been chosen for clarity by separating the weak forward secrecy property from the strong one that relies on algebraic adversaries. Using a different sequence of hops, one can remove the extra term by aggregating the two bad events that give rise to these terms into a single one.

Theorem 6.2. *SPAKE2 is PFS-secure in the random-oracle model under the Gap Computational Diffie-Hellman assumption. More precisely, for every attacker $\mathcal{A}$ against SPAKE2, there exist attackers $\mathcal{B}_1$ and $\mathcal{B}_3$ against the Gap Diffie-Hellman problem, attacker $\mathcal{B}_2$ against the Gap Discrete Logarithm problem, and attacker $\mathcal{B}_4$ against the Gap GPCCDH Diffie-Hellman problem such that*

$$\mathsf{Adv}^{\mathrm{SPAKE2}}_{\mathcal{A}}(\,) \leq \frac{q_s}{|\mathcal{P}|} + \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_1}(\,) + \mathsf{Adv}^{\mathsf{GDL}}_{\mathcal{B}_2}(\,) + \mathsf{Adv}^{\mathcal{D}^\star\text{-}\mathsf{GCDH}}_{\mathcal{B}_3}(\,) + 2q_s \cdot \mathsf{Adv}^{\mathsf{GPCCDH}}_{\mathcal{B}_5}(\,) + \frac{(q_s + q_e)^2}{2q}$$

Note that in the above bound there exists an extra $2q_s/|\mathcal{P}|$ within the GPCCDH advantage, since we only have guarantees based on GCDH once that advantage term grows above this limit, so this bound is worse than the one in Theorem 6.1.

We give a joint proof for both theorems, since they only differ in the proof of the last game hop. The code of all games and adversaries is given in Appendix A.

Proof. We prove the security of SPAKE2 using a sequence of games shown in Figs. 3 to 5. In Table 1 we give a summary of the meaning of bad events in each game.

Game G_1. This is the initial security game, so we have:

$$\mathsf{Adv}^{\mathrm{SPAKE2}}_{\mathcal{A}}(\,) = |\Pr[\,\mathsf{G}_1 \Rightarrow \mathsf{T}\,] - 1/2\,|$$

Game G_2. From game G_1 to game G_2 we introduce a bad flag bad_2. The bad_2 flag is set whenever

- a session is about to be accepted on the server side that collides on $(U, S, X^\star, Y^\star)$ with any previously accepted session;
- a session is about to be accepted on the user side that collides on $(U, S, X^\star, Y^\star)$ with any previously accepted user session.

If bad_2 is set, then the oracle call is ignored and the session is not accepted.

Note that this implies that all accepted sessions on the server (resp. client) side are unique and that no session can have more than one partner. The two games are identical until bad occurs, and this event can be bound by a statistical term. More precisely, for an attacker placing at most q_e queries to Execute and q_s queries to the Send oracles, we have for large q denoting the order of the group:

$$|\Pr[\mathsf{G}_1 \Rightarrow \mathsf{T}] - \Pr[\mathsf{G}_2 \Rightarrow \mathsf{T}]| \leq \frac{(q_e + q_s)^2}{2q}$$

Here, $q_e + q_s$ denotes the maximum number of accepted sessions (we count passively partnered sessions as one here) and this is a birthday bound computed pessimistically for an attacker that makes accepted sessions collide by fixing the user, server and one of the transmitted group elements in all accepted sessions.

Game G_3. In this game we take advantage of the fact that accepted sessions have unique traces (modulo partnering) to make the freshness condition explicit in the game (we extend the state of sessions to keep track of freshness with a new field fr). We also remove the code that refers to bad_2. Since nothing changes, we have

$$\Pr[\mathsf{G}_3 \Rightarrow \mathsf{T}] = \Pr[\mathsf{G}_2 \Rightarrow \mathsf{T}]$$

Game G_4. In this game we make the keys of sessions accepted as a result of calls to Execute independent from the random oracle accessible to the attacker. These sessions obtain keys from a new random oracle T_e that is indexed by the session trace $(U, S, X^\star, Y^\star)$. Note that, due to the guarantee of session uniqueness on the server and client sides, there is no room for ambiguity. We set a bad flag bad_4 if ever the answers given to the attacker could be inconsistent with the previous game: either because a new accepted session in Execute collides with a previous call to H, or because a new call to H collides with a previously accepted session in Execute. The games are identical until bad occurs, so we have:

$$|\Pr[\mathsf{G}_3 \Rightarrow \mathsf{T}] - \Pr[\mathsf{G}_4 \Rightarrow \mathsf{T}]| \leq \Pr[\mathsf{G}_4 \Rightarrow \mathsf{bad}_4]$$

The probability of bad_4 occurring in G_4 can be tightly reduced to the standard Gap CDH problem,which means that:

$$\Pr[\mathsf{G}_4 \Rightarrow \mathsf{bad}_4] \leq \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_1}()$$

The reduction to Gap CDH is given in Lemma 6.4.

Game G_5. In game G_5 we again rearrange code in preparation for the next hop. We remove the code that deals with bad events, and we change Execute so that nothing depends on passwords. We can do this because the random oracle that generates these keys only depend on the trace, which can be sampled independently of passwords. We have

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

Game G_6. In this game we change the way in which we generate keys for Send queries. We treat corrupt sessions, for which the attacker may trivially compute the key, as before. The remaining keys are derived using an independent random oracle T_s. We set a bad flag bad_6 if ever there could be an inconsistency with the main random oracle. As before, this means that:

$$|\Pr[\mathsf{G}_5 \Rightarrow \mathsf{T}] - \Pr[\mathsf{G}_6 \Rightarrow \mathsf{T}]| \leq \Pr[\mathsf{G}_6 \Rightarrow \mathsf{bad}_6]$$

We also observe that, since all fresh sessions that can be tested now have keys derived using random oracles independent from H, accepted keys of fresh sessions are independently distributed from anything else in the game. This means that the attacker has no information on bit b and has therefore 0 advantage in this game. From the previous equations we can derive that:

$$\mathsf{Adv}^{\mathrm{SPAKE2}}_{\mathcal{A}}() \leq \frac{(q_s + q_e)^2}{2q} + \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_1}() + \Pr[\mathsf{G}_6 \Rightarrow \mathsf{bad}_6]$$

To bound the probability of bad_6 we need to continue our sequence of hops.

GAME G_7. We modify the way in which $X^\star$ and $Y^\star$ are generated, removing the component that depends on the password. The distribution of the $X^\star$ and $Y^\star$ values does not change with respect to the previous game, but we need to adapt the way in which the Diffie-Hellman tuples are computed, to make sure we obtain the same values as in the previous game. After we do this change, passwords associated with fresh sessions are still (and only) used to check for the bad event bad_6. We add a flag bad_7 to account for the unlikely case that a password occurring in the game (either queried by the adversary or sampled by the challenger) collides with the secret exponents used in $X^\star$ and $Y^\star$ sampled by the challenger. If this happens, we reset bad_6 at the end of the game. The probability of this event occurring can be easily reduced to the GDL problem, as shown in Lemma 6.5. We therefore have:

$$\Pr[\,\mathsf{G}_6 \Rightarrow \mathsf{bad}_6\,] \leq \Pr[\,\mathsf{G}_7 \Rightarrow \mathsf{bad}_6\,] + \mathsf{Adv}^{\mathsf{GDL}}_{\mathcal{B}_2}(\,)$$

GAME G_8. We now change the checking of bad_6 to allow delaying all password samplings as late as possible in the game: upon corruption or at the end of the game. This means we stop verifying password equality when checking for bad_6 in the SEND oracles (there is no impact in the random oracle simulation because the attacker provides the password explicitly there). Without additional changes we might not be able to tightly relate the probability of bad_6 in this new game to its probability in the previous game, since we have relaxed the check that detects it. We therefore need to add additional checks that allow us to continue the proof. There are two important cases to distinguish:

- the attacker causes bad_6 for a password that has not been generated yet (this includes all occurrences in the SEND oracles and some in the random oracle simulation).
- the attacker causes bad_6 by querying the random oracle on the correct password *after* the password has been corrupted (intuitively it does not need to guess the password in this case);

For the second case we set a bad flag bad^2_8 and no longer set bad_6.

For the other cases we add the potentially problematic random oracle entries to a list T_{bad}, so that we can delay the analysis of potential inconsistencies until the end of the game. At the end of the game we split the checking of bad_6 into two cases. We set a bad flag bad^1_8 if in the T_{bad} log of problematic H calls there are two entries consistent with a single entry in T_s, i.e., with different passwords; in this case we no longer set bad_6. Finally, we check for bad_6 simply by going through the list of problematic entries and checking whether there is a password match.

The combined actions of the three bad flags in this game account for all possible cases in which bad_6 can occur in game G_7, so we can write:

$$\Pr[\,\mathsf{G}_6 \Rightarrow \mathsf{bad}_7\,] \leq \Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}_6\,] + \Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}^1_8\,] + \Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}^2_8\,]$$

The probability of bad^1_8 occurring in G_8 can be tightly reduced to Gap CDH, where we use here the generalized version for any parameter distribution $\mathcal{D}$ where the CDH problem is hard to compute with the help of a DDH oracle;[6] the reduction is given in Lemma 6.6.

We complete the analysis of this game by further observing that the probability of bad_6 happening is easy to bound. Indeed, the size of the log in which bad_6 is checked has size at most q_s (since there can be no duplicates for the same trace) and all entries are added to this set before the corresponding password is sampled. We therefore have

$$\Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}_6\,] \leq \frac{q_s}{|\mathcal{P}|}$$

Putting these results together, we obtain:

$$\begin{aligned}\Pr[\,\mathsf{G}_7 \Rightarrow \mathsf{bad}_6\,] &\leq \Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}_6\,] + \Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}^1_8\,] + \Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}^2_8\,] \\ &\leq \tfrac{q_s}{|\mathcal{P}|} + \mathsf{Adv}^{\mathcal{D}^\star\text{-}\mathsf{GCDH}}_{\mathcal{B}_3}(\,) + \Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}^2_8\,]\end{aligned}$$

[6] Note that this does not weaken our result, as one can take $\mathcal{D}$ to be the uniform distribution. On the other hand, it makes it clear that different parameter generation procures can be used and our proof stil applies.

To complete the proof we need to bound the probability of bad_8^2 occurring in G_8, and so we continue our sequence of hops.

GAME G_9. For clarity of presentation, in game G_9 we perform a cleanup, removing all code unrelated to bad_8^2. We also add the possibility that the attacker passes an additional parameter alg when it delivers a message. This additional parameter is ignored for now, so clearly

$$\Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}_8^2\,] = \Pr[\,\mathsf{G}_9 \Rightarrow \mathsf{bad}_8^2\,]$$

To complete the proof for Theorem 6.2 we give in Lemma 6.8 a reduction of the above probability to GPCCDH, which yields the desired bound:

$$\mathsf{Adv}^{\mathrm{SPAKE2}}_{\mathcal{A}}() \leq \frac{q_s}{|\mathcal{P}|} + \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_1}() + \mathsf{Adv}^{\mathsf{GDL}}_{\mathcal{B}_2}() + \mathsf{Adv}^{\mathcal{D}^\star\text{-}\mathsf{GCDH}}_{\mathcal{B}_3}() + 2q_s \cdot \mathsf{Adv}^{\mathsf{GPCCDH}}_{\mathcal{B}_5}() + \frac{(q_s+q_e)^2}{2q}$$

We proceed with a final game hop to complete the proof for Theorem 6.1.

GAME G_{10}. This is our final game. Here we restrict our attention to algebraic adversaries. This means that, when performing an active attack, the adversary provides an additional input alg to the SEND oracles that gives a representation of the group element it is producing in terms of the other group elements it observed in the game. These elements include g, M, N and all the honestly generated $X^\star = M^x$ and $Y^\star = N^y$. We note that this means that any group element produced by $\mathcal{A}$ can be rewritten as a representation of the form $\mathsf{alg} = [(g^a), (M^b), (N^c)]$.

We now add a bad flag bad_{10} that excludes some password guessing events related to bad_8^2. When these happen, we no longer set bad_8^2, whereas this would have been set in the previous game. This means that:

$$\Pr[\,\mathsf{G}_9 \Rightarrow \mathsf{bad}_8^2\,] \leq \Pr[\,\mathsf{G}_{10} \Rightarrow \mathsf{bad}_8^2\,] + \Pr[\,\mathsf{G}_{10} \Rightarrow \mathsf{bad}_{10}\,]$$

Event bad_{10} is detected whenever a password is sampled at the moment of corruption, and it hits an already fixed representation of a group element produced by the attacker. The offending values are $b+\lambda(c-\mathsf{pw})$ for group elements delivered at SENDTERM and $b-\mathsf{pw}+\lambda c$ for group elements delivered at SENDRESP. If this event is detected, then we never set bad_8^2, so in this game the analysis of bad_8^2 can proceed under the assumption that bad_{10} did not occur. The probability of this event is easy to bound as

$$\Pr[\mathsf{G}_{10} \Rightarrow \mathsf{bad}_{10}] \leq \frac{q_s}{|\mathcal{P}|}$$

The remaining cases where we set bad_8^2 can be reduced to the Gap Squared Diffie-Hellman problem (GSqDH), as shown in Lemma 6.7. This completes the proof and yields the bound.

$$\mathsf{Adv}^{\mathrm{SPAKE2}}_{\mathcal{A}}() \leq \frac{2q_s}{|\mathcal{P}|} + \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_1}() + \mathsf{Adv}^{\mathsf{GDL}}_{\mathcal{B}_2}() + \mathsf{Adv}^{\mathcal{D}^\star\text{-}\mathsf{GCDH}}_{\mathcal{B}_3}() + \mathsf{Adv}^{\mathsf{GSqDH}}_{\mathcal{B}_4}() + \frac{(q_s+q_e)^2}{2q}$$

A different sequence of games permits removing a $q_s/|\mathcal{P}|$ term, yielding the expected bound for PAKE. □

Corollary 6.3. *SPAKE2 is weak PFS-secure in the random-oracle model under the Gap Computational Diffie-Hellman assumption. More precisely, for every attacker $\mathcal{A}$ against SPAKE2, there exist attackers $\mathcal{B}_1$ and $\mathcal{B}_3$ against the Gap Diffie-Hellman problem and attacker $\mathcal{B}_2$ against the Gap Discrete Logarithm problem such that*

$$\mathsf{Adv}^{\mathrm{SPAKE2}}_{\mathcal{A}}() \leq \frac{q_s}{|\mathcal{P}|} + \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_1}() + \mathsf{Adv}^{\mathsf{GDL}}_{\mathcal{B}_2}() + \mathsf{Adv}^{\mathcal{D}^\star\text{-}\mathsf{GCDH}}_{\mathcal{B}_3}() + \frac{(q_s+q_e)^2}{2q}$$

Proof. In the weak forward secrecy model, corruption of password $\mathsf{pw}_{\mathrm{US}}$ makes all fresh session where the adversary was active immediately unfresh and, furthermore, corruption is not allowed if such a

session was already tested. It is easy to see that a simple adaptation of the previous proof can be done for this setting, where the probability of event bad_8^2 occurring is 0 and the advantage of the adversary is still 0: for such cases where the event would occur we simply return $T_s[\mathsf{tr}]$ as the answer to the problematic H query. □

We now prove the auxiliary lemmas supporting the proofs of Theorems 6.1 and 6.2.

Lemma 6.4. *For every attacker $\mathcal{A}$, there exists an attacker $\mathcal{B}_1$, such that*

$$\Pr[\,\mathsf{G}_4 \Rightarrow \mathsf{bad}_4\,] \leq \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_1}(\,)$$

Proof. Let us consider the GCDH attacker $\mathcal{B}_1$ in Fig. 6. It gets a generalized challenge of the form $(X_1, \ldots, X_{q_e}, Y_1, \ldots, Y_{q_e})$ and finds $\mathsf{CDH}(X_i, Y_j)$, for some $1 \leq i, j \leq q_e$. This problem is tightly equivalent to $\mathcal{U}$-GCDH by random self-reducibility. The attacker embeds X_i and Y_j in traces for protocol executions that result from calls to Execute. For all calls to Send oracles, the attacker runs everything as in G_4. The DDH oracle is used to check for the bad event, in which case a solution to the GCDH problem was found. It uses the DDH oracle whenever the rules of the game require it, and it can also use it to check if bad_4 occurred, in which case it can solve the Gap CDH challenge.[7] □

Lemma 6.5. *For every attacker $\mathcal{A}$, there exists an attacker $\mathcal{B}_2$, such that*

$$\Pr[\,\mathsf{G}_7 \Rightarrow \mathsf{bad}_7\,] \leq \mathsf{Adv}^{\mathsf{GDL}}_{\mathcal{B}_2}(\,)$$

Proof. Let us consider a GDL attacker $\mathcal{B}_2$ in Fig. 6. It gets a challenge of the form A and finds $\mathsf{DL}(A)$. The attacker uses A in Send queries to compute $X^\star = A \cdot g^{\Delta}$ and $Y^\star = A \cdot g^{\Delta}$. Note that the distribution of these values is correct. The checking of CDH tuples, which is needed for correctly maintaining random-oracle consistency, is carried out using the DDH oracle.

If bad_7 occurs, our reduction recovers the discrete logarithm of A offset by a known quantity. Indeed, suppose the attacker guessed $\mathsf{pw} = x$ or $\mathsf{pw} = y$ in game G_7, which means we have $X^\star = g^{m\mathsf{pw}}$ or $Y^\star = g^{n\mathsf{pw}}$. In our reduction we program both $X^\star$ and $Y^\star$ as $g^{a+\Delta}$, where a is the unknown discrete logarithm, and we know all of m, pw and Δ. This means that, if the guess was successful, we can compute the discrete logarithm as $a = m\mathsf{pw} - \Delta$ or $a = n\mathsf{pw} - \Delta$. □

Lemma 6.6. *For every attacker $\mathcal{A}$, there exists an attacker $\mathcal{B}_3$, such that*

$$\Pr[\,\mathsf{G}_7 \Rightarrow \mathsf{bad}_8^1\,] \leq \mathsf{Adv}^{\mathcal{D}^\star\text{-}\mathsf{GCDH}}_{\mathcal{B}_3}(\,)$$

Proof. Let us consider a $\mathcal{D}^\star$-GCDH attacker $\mathcal{B}_3$ in Fig. 6. It gets a challenge of the form (M, N) and finds $\mathsf{CDH}(M, N)$. The attacker uses (M, N) as the variables of the same name in game G_8. The checking of CDH tuples, which is needed for correctly maintaining random-oracle consistency and detecting bad events, is carried out using the DDH oracle.

Once the game terminates, the attacker checks for the occurrence of bad_8^1 and if offending entries exist, it recovers $(X^\star, Y^\star, \mathsf{pw}, Z)$ and $(X^\star, Y^\star, \mathsf{pw}', Z')$ such that $\mathsf{pw} \neq \mathsf{pw}'$ and the following holds for known x or known y, and known pw and pw':

$$\mathsf{CDH}(g^{mx}/M^{\mathsf{pw}}, g^{ny}/N^{\mathsf{pw}}) = Z \;\wedge\; \mathsf{CDH}(g^{mx}/M^{\mathsf{pw}'}, g^{ny}/N^{\mathsf{pw}'}) = Z'$$

Letting $Z = g^z$, $Z' = g^{z'}$, we can rewrite these equations as

$$\begin{cases} m(x - \mathsf{pw}) \cdot n(y - \mathsf{pw}) = z \\ m(x - \mathsf{pw}') \cdot n(y - \mathsf{pw}') = z' \end{cases}$$

[7] Note that it actually suffices to have a restricted DDH oracle in which one of the inputs is fixed as in the Strong Diffie-Hellman (SDH) problem [ABR01], so a tight reduction to SDH is also possible. Furthermore, storing all possible values that cause the bad event and returning one at random gives a reduction to CDH that loses a linear factor in the maximum size of the random-oracle table.

Let us assume that the attacker knows x (the other case is symmetric). We scale the equations and rearrange the formula as follows:

$$\begin{cases} m(x-\mathsf{pw})(x-\mathsf{pw}')\cdot n(y-\mathsf{pw}) = z\cdot(x-\mathsf{pw}') \\ m(x-\mathsf{pw}')(x-\mathsf{pw})\cdot n(y-\mathsf{pw}') = z'\cdot(x-\mathsf{pw}) \end{cases}$$

Subtracting the two equations, we get

$$z(x-\mathsf{pw}') - z'(x-\mathsf{pw}) = mn(x-\mathsf{pw})(x-\mathsf{pw}')(\mathsf{pw}'-\mathsf{pw})$$

And we can derive a formula for the desired CDH:

$$g^{mn} = \left(Z^{x-\mathsf{pw}'}\cdot Z'^{\mathsf{pw}-x}\right)^{\frac{1}{(x-\mathsf{pw})(x-\mathsf{pw}')(\mathsf{pw}'-\mathsf{pw})}}$$

This formula can be computed by the attacker, provided that $x \neq \mathsf{pw}$ and $x \neq \mathsf{pw}'$, which we know to be the case as otherwise the bad event would not have occurred due to the action of bad_7. □

Lemma 6.7. *For every algebraic attacker $\mathcal{A}$, there exists an attacker $\mathcal{B}_4$, such that*

$$\Pr[\,\mathsf{G}_{10} \Rightarrow \mathsf{bad}_8^2\,] \leq \mathsf{Adv}^{\mathsf{GSqDH}}_{\mathcal{B}_4}(\,)$$

Proof. The reduction is given in Fig. 7. In the unlikely event that $M = 1$ there is nothing to do, as the problem is trivial to solve. We can also assume that the conditions checked to activate bad_7 all failed, as otherwise bad_8^2 would have been reset. This implies, in particular, that there are no entries in T_s such that the secret exponent x or y collides with the password $\mathsf{pw}_{\mathrm{US}}$.

Given M, the reduction sets $N = M^\lambda$, for random non-zero λ, which yields correctly distributed global parameters. Similarly to what happened in the proof of Lemma 6.6, we use the DDH oracle to ensure the simulation is perfect, so it remains to show that our reduction can compute $\mathrm{CDH}(M, M)$ whenever bad_8^2 occurs. There are two cases to consider, depending on whether we are dealing with a session accepted by a user or by a server.

Sessions accepted in SendTerm. We know that the following equation holds.

$$Z = \mathsf{CDH}\left(\frac{M^x}{M^{\mathsf{pw}}}, \frac{g^a M^b N^c}{N^{\mathsf{pw}}}\right)$$

This can be rewritten as:

$$Z = \mathsf{CDH}(M^{x-\mathsf{pw}}, g^a M^b N^{c-\mathsf{pw}}) \Leftrightarrow \frac{Z}{M^{a(x-\mathsf{pw})}} = \mathsf{CDH}(M^{x-\mathsf{pw}}, M^b N^{c-\mathsf{pw}})$$

Replacing $N = M^\lambda$ and rearranging the exponents we get

$$\frac{Z}{M^{a(x-\mathsf{pw})}} = \mathsf{CDH}(M, M)^{(x-\mathsf{pw})(b+\lambda(c-\mathsf{pw}))}$$

We can therefore recover the desired SqDH provided that $(x-\mathsf{pw})(b+\lambda(c-\mathsf{pw})) \neq 0$. So see that this can never happen, note $b + \lambda(c - \mathsf{pw}) \neq 0$, as otherwise bad_{10} would have happened and this entry would not be in the list, and we also excluded $x = \mathsf{pw}$ above. This means that we can compute the SqDH.

Sessions accepted in SendResp. In this case we know that the following equation holds.

$$Z = \mathsf{CDH}\left(\frac{g^a M^b N^c}{M^{\mathsf{pw}}}, \frac{N^y}{N^{\mathsf{pw}}}\right)$$

This can be rewritten as:

$$Z = \mathsf{CDH}(g^a M^{b-\mathsf{pw}} N^c, N^{y-\mathsf{pw}})$$

Replacing $N = M^{\lambda}$ and rearranging the exponents we get

$$Z = \mathsf{CDH}(g^a M^{b-\mathsf{pw}+\lambda c}, M^{\lambda(y-\mathsf{pw})}) \Leftrightarrow \frac{Z}{M^{a\lambda(y-\mathsf{pw})}} = \mathsf{CDH}(M,M)^{(b-\mathsf{pw}+\lambda c)(y-\mathsf{pw})}$$

We can therefore recover the desired SqDH provided that $(b-\mathsf{pw}+\lambda c)(y-\mathsf{pw}) \neq 0$. So see that this can never happen, note that we know that $y-\mathsf{pw} \neq 0$ and it must be the case that $b-\mathsf{pw}+\lambda c \neq 0$, as otherwise bad_{10}^2 would have happened and this entry would not be in the list. This means that the SqDH can be computed. □

Lemma 6.8. *For every attacker $\mathcal{A}$, there exists an attacker $\mathcal{B}_5$, such that*

$$\Pr[\,\mathsf{G}_9 \Rightarrow \mathsf{bad}_8^2\,] \leq 2q_s \cdot \mathsf{Adv}^{\mathsf{GPCCDH}}_{\mathcal{B}_5}(\,)$$

Proof. The reduction is given in Fig. 7. Our adversary $\mathcal{B}_5$ guesses a query in the range [0..q_s-1] and chooses a role uniformly at random to guess whether the challenge should be injected on the responder side or on the initiator side. The behavior conditioned on these guesses is shown in the figure as two different versions of $\mathcal{B}_5$, one for initiator side and another for responder side. Similarly to what happened in the proof of Lemma 6.6, we use the DDH oracle to ensure the simulation is perfect, and the correct result is produced as long as the guesses are correct. On the initiator side, the reduction is slightly more intricate as one needs to embed A_1 in all possible initiator sessions that might lead to the bad event. The bound on the theorem follows from the fact that the reduction only works if our attackers guesses are correct. □

7 Security with Key Confirmation

The result we obtained in the previous section provides guarantees even if the derived secret keys are used before a key confirmation step. However, in most practical settings a key confirmation step exists and, in this case, a much better security bound can be proved. This is what we do in this section. We show the modified protocol in Fig. 2. Note that a one-time secure MAC suffices.

Theorem 7.1. *SPAKE2 with key confirmation is tightly PFS-secure in the random-oracle model under the GCDH assumption. More precisely, for every attacker $\mathcal{A}$ against SPAKE2, there exist attackers $\mathcal{B}_6$ and $\mathcal{B}_3$ against the Gap Diffie-Hellman problem, attacker $\mathcal{B}_2$ against the Gap Discrete Logarithm problem, attacker $\mathcal{B}_7$ against the pseudorandomness of KDF and attacker $\mathcal{B}_8$ against the unforgeability of the MAC scheme such that*

$$\mathsf{Adv}^{\mathrm{SPAKE2}}_{\mathcal{A}}(\,) \leq \frac{q_s}{|\mathcal{P}|} + \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_6}(\,) + \mathsf{Adv}^{\mathsf{GDL}}_{\mathcal{B}_2}(\,) + \mathsf{Adv}^{\mathcal{D}^\star\text{-}\mathsf{GCDH}}_{\mathcal{B}_3}(\,) + q_s \cdot \Big(\mathsf{Adv}^{\mathsf{1PRF}}_{\mathcal{B}_7}(\,) + \mathsf{Adv}^{\mathsf{1UF}}_{\mathcal{B}_8}(\,)\Big) + \frac{(q_s+q_e)^2}{2q}$$

Proof. Again we use a sequence of hops, but we explain here only the differences to the proofs in the previous section. We give the games in Figs. 8 to 10 in Appendix B.

The hops up to game G_4 are identical to those for the proofs in the previous section. In game G_4, as before, we exclude the possibility of the adversary breaking a session where it behaved passively but, in addition to Execute queries, we handle also the sequences of Send queries where the adversary establishes sessions with traces $(U, S, X^\star, Y^\star)$ where the messages were generated by the game. We give a modified reduction to deal with this case in Lemma 7.3 and note that here the gap oracle is not only used to detect the correct CDH answer, but to maintain consistency of all other sessions where the attacker is active.

We now carry on the the discussion of the following hops in detail from game G_6.

Game G_6. This game introduces an independent random oracle for keys derived for (possibly aborted) fresh sessions. Flag bad_6 corresponds to the attacker observing an inconsistency in the hash queries for H. As in the previous proof, we have

$$\mathsf{Adv}^{\mathrm{SPAKE2}}_{\mathcal{A}}(\,) \leq \frac{(q_s+q_e)^2}{2q} + \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_6}(\,) + \mathsf{Adv}^{\mathsf{GDL}}_{\mathcal{B}_2}(\,) + \Pr[\,\mathsf{G}_7 \Rightarrow \mathsf{bad}_6\,],$$

G_2	bad_2	Accepted trace on responder side collides with any previously accepted trace or accepted trace on initiator side collides with trace of any previously accepted initiator session. Bounded by a statistical term.
G_4	bad_4	Random oracle query consistent with the trace accepted as a result of a query to EXECUTE. Such an event would allow attacker to detect an inconsistency between the outputs of TEST or REVEAL and the random oracle. Bounded by a tight reduction to GCDH.
G_6	bad_6	Random oracle query consistent with the trace accepted as a result of a query to one of the SEND oracles and declared fresh. Such an event would allow attacker to detect an inconsistency between the outputs of TEST or REVEAL and the random oracle. Adversary's view is independent of b in this game, but bad_6 is hard to bound.
G_7	bad_6	The meaning is the same as in the previous game, but it now is only triggered in cases where the other bad event in this game does not occur.
	bad_7	Either there is a call to the random oracle with $\mathsf{pw} = x$ or $\mathsf{pw} = y$, where x or y is one of the random values sampled in the game. Or a password was sampled by the game that also collides with one of these values. Bounded by a tight reduction to GDL.
G_8	bad_6	The meaning is the same as in the previous games, but it now is only triggered in cases where the other bad event in this game does not occur. At this point this event can only happen for random oracle queries placed before the associated password was sampled in the game. Furthermore, we know there can be at most q_s entries, which results in a $q/\lvert\mathcal{P}\rvert$ term.
	bad_8^1	The attacker made two different random oracle queries consistent with the trace of the same fresh session, i.e. with different passwords, before the corresponding password was sampled. Bounded by a tight reduction to GCDH.
	bad_8^2	The attacker made a random oracle query consistent with the trace of a fresh session, after the corresponding password was corrupted and hence sampled by the game. We delay bounding the probability of this event until game G_9 for clarity.
G_9	bad_8^2	Same meaning as in the previous game and same probability of occurrence. We fork here the proofs of Theorems 6.1 and 6.2. For the latter we immediately bound the event with a tight reduction to GPCCDH, which yields a loose reduction to GCDH via the splitting lemma of [PS00, AP05]. For Theorem 6.2 we delay bounding this event until G_{10}.
G_{10}	bad_8^2	Same meaning as in the previous game, but now only set when the other bad flag in this game is not set. Bound using a reduction to GSqDH assuming an algebraic adversary.
	bad_{10}	The adversary commited to a representation of a group element in a SEND query that predicts a password sampled later in the game. The probability of this event occurring can be easily bound as $q_s/\lvert\mathcal{P}\rvert$. We also note here that the set of traces for which this event needs to be checked is disjoint from the set of traces for which bad_6 needs to be checked in game G_8. This is why the bound we give for Theorem 6.1 can be improved by removing a $q_e/\lvert\mathcal{P}\rvert$ term using a different sequence of games.

Table 1. Meaning of the bad events in the different games.

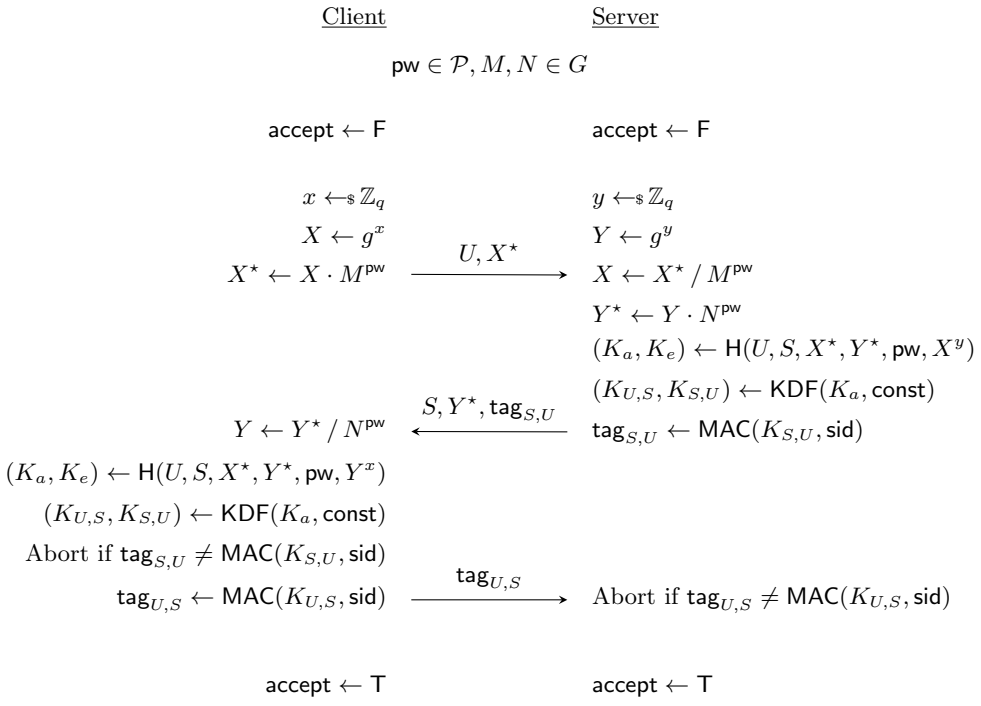

Fig. 2. The SPAKE2 protocol [AP05] with key confirmation. Here, const is a fixed point in the domain of the KDF, $\mathsf{sid} = (U, S, X^\star, Y^\star)$ and K_e is the accepted session key.

where the term corresponding to $\mathcal{B}_6$ highlights the fact that the reduction that deals with passive sessions is the one given in Lemma 7.3. Furthermore, as before, the adversary's view in this game is independent of the secret bit b and it only remains to bound the probability of bad_6 occurring. Let us take a closer look at game G_6 to confirm that this is the case.

First of all, let us confirm that all the session keys corresponding to fresh accepted sessions have been moved to an independent random oracle, whereas all revealable (accepted) non-fresh keys have not. Notice that random oracle queries are added to T_s rather than T in two occasions:

- When a SEND query starts a responder session for an uncorrupted password; this session may or may not turn out to be accepted and, if it is, it may or may not be fresh.
- When a SEND query is about to conclude a fresh initiator session (such an entry has not yet been previously added to T_s since we never do this for sessions where the attacker is passive); this session may or may not be accepted due to MAC verification.

For the latter type of entries, if the MAC verification passes, then the session will be accepted as fresh, so we must ensure that such an entry is never removed from T_s in this case. Removal could only occur in the CORRUPT oracle, but here the check performed guarantees that entries are only removed from T_s when no accepted fresh session with that trace exists.

For the first type of entry, we must ensure that the entry is removed from T_s whenever it might be accepted as *unfresh*. This is ensured by the corruption oracle: it removes any such entries upon corruption and reprograms the global random oracle with the corresponding key material when corruption occurs before any session with that trace is accepted.

Note further that the corrupt oracle also cleans up any entries in T_s corresponding to traces of potentially fresh sessions that the attacker failed to conclude before corruption (e.g., by delivering an inconsistent MAC). This means that, after corruption of $\mathsf{pw}_{\mathrm{US}}$, the only entries that remain in T_s corresponding to (U, S) interactions must correspond to at least one fresh accepted session.

Now let us confirm that any inconsistencies detected by the adversary in its queries to the global random oracle (because it has observed in the game the output of the independent random oracle or some usage of it) are correctly signalled by activating the bad_6 flag.

The attacker may observe the action of the independent random oracle via the following oracle queries:

- REVEAL or TEST queries to accepted fresh sessions.
- MAC values computed using keys derived from entries in T_s rather than the global random oracle.

For all entries added and never removed from T_s, the bad_6 flag is activated if a matching entry is ever queried to H in the game. This covers the first class of observations. MAC checks and computations for such sessions are handled similarly.

The possibility remains, however, that the attacker might observe a MAC value related to an entry in T_s that is later moved to T. If the query to H that causes the inconsistency comes while the offending entry is in T_s, then bad_6 is activated as before. Otherwise, note that entries moved from T_s to T retain the key material, so subsequent queries to H will actually be consistent.

GAME G_7. The intuition of this hop is identical to that for game G_7 in the proofs of the previous section: we change the way we generate messages in SEND queries and set a bad flag if, due to some unlikely event, one of the random exponents generated by the game to do this occurs as a password in the game. This event can be bounded by a reduction to GDL, as shown in Lemma 6.5, so we have:

$$\mathsf{Adv}^{\mathrm{SPAKE2}}_{\mathcal{A}}() \leq \frac{(q_s + q_e)^2}{2q} + \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_1}() + \mathsf{Adv}^{\mathsf{GDL}}_{\mathcal{B}_2}() + \Pr[\,\mathsf{G}_7 \Rightarrow \mathsf{bad}_6\,]$$

GAME G_8. In this game we reject any tags submitted by the attacker to a SEND oracle that would result in a new fresh session being accepted, when it has no information on the corresponding authentication key. More in detail:

- If a fresh session is about to be accepted on the initiator side, and there does not exist a matching session on the responder side, then the attacker has never observed any computation based on the corresponding T_s entry.
- If a fresh session is about to be accepted on the responder side, and there does not exist a matching session on the initiator side, then the attacker may have observed a MAC computed with $K_{S,U}$, but it has no information about $K_{U,S}$.

Note that, by rejecting these sessions, the attacker can now only complete fresh sessions by behaving passively.[8] We add a bad flag bad_8 to deal with cases where the tag would have been accepted in the previous game and, when this flag is activated, the potentially fresh session is forgotten and removed from the independent random oracle. This also means that, in this game, bad_6 can now only occur after corruption for sessions where the attacker behaved passively, which is why we can further restrict the setting of the bad fact to non-corrupted passwords. To see that this is indeed the case, observe that bad_6 happens while entries are in T_s, but in this game we remove from T_s traces corresponding to active behaviour before the corresponding password is corrupted (as otherwise these sessions would not lead to fresh sessions anyway).

The probability of bad_8 occurring can be bounded using the PRF property of KDF and the standard (one-time) unforgeability of the MAC scheme. A formal proof is given in Lemma 7.4. Therefore we have

$$\Pr[\,\mathsf{G}_7 \Rightarrow \mathsf{bad}_6\,] \leq \Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}_6\,] + q_s \cdot \left(\mathsf{Adv}^{\mathsf{1PRF}}_{\mathcal{B}_7}(\,) + \mathsf{Adv}^{\mathsf{1UF}}_{\mathcal{B}_8}(\,)\right)$$

Game G_9. In this game we introduce changes similar to those we introduced in G_8 of the proofs of Theorems 6.1 and 6.2 in moving all password-related checks to the end of the game. However, unlike in the previous section, we can now move all occurrences of bad_6 to the end, since we handled the problematic queries placed after corruption in the previous game hops. We move checks to the end by introducing a T_{bad} list, and a final bad event bad_9 for executions where the attacker is able to construct two different consistent entries in this list for the same trace and different passwords.

Analysis of bad_9 and bad_6 in game G_9 are identical to the analyses of bad_6 and bad^1_8 in game G_8 of the proofs in the previous section: bad_6 is the probability of passwords guessing, whereas bad_{10} can be reduced to GCDH as proved in Lemma 6.6. We therefore have

$$\Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}_6\,] \leq \Pr[\,\mathsf{G}_9 \Rightarrow \mathsf{bad}_6\,] + \Pr[\,\mathsf{G}_9 \Rightarrow \mathsf{bad}_9\,] \leq \frac{q_e}{|\mathcal{P}|} + \mathsf{Adv}^{\mathsf{GCDH}}_{\mathcal{B}_3}(\,)$$

This concludes the proof. □

The above proof also establishes that SPAKE2 with key confirmation provides explicit authentication defined formally as:

- if a client U accepts a session with server S, then S has a pending unique session where it has derived the same key and session identifier $(U, S, X^\star, Y^\star)$;
- if a server S accepts a session with client U, then U has already accepted a unique session with the same key and session identifier $(U, S, X^\star, Y^\star)$.

We state this as the following corollary, which shows that impersonation can only be achieved with negligible probability beyond guessing the password.

Corollary 7.2. *SPAKE2 with key confirmation provides explicit authentication.*

Proof. The proof follows the same sequence of hops as that of Theorem 7.1. We can observe that in the final game a session is accepted only if the adversary behaved passively, which means that explicit

[8] Forging MACs that result the same trace being accepted as in passive behaviour is not considered an active attack. This maps to the standard existential unforgeability notion.

authentication is guaranteed with the same bound. More precisely, the attacker can break explicit authentication with probability at most

$$\frac{q_s}{|\mathcal{P}|} + \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_6}() + \mathsf{Adv}^{\mathsf{GDL}}_{\mathcal{B}_2}() + \mathsf{Adv}^{\mathcal{D}^\star\text{-}\mathsf{GCDH}}_{\mathcal{B}_3}() + q_s \cdot \left(\mathsf{Adv}^{\mathsf{1PRF}}_{\mathcal{B}_7}() + \mathsf{Adv}^{\mathsf{1UF}}_{\mathcal{B}_8}()\right) + \frac{(q_s+q_e)^2}{2q}$$

□

Remark. We note that the proof strategy we use here for dealing with sessions where the attacker behaves passively actually shows that SPAKE2 is secure in a stronger model where the adversary does not have access to the Execute oracle and can instead adaptively decide how to orchestrate a bounded number of q_e passive attacks concurrently with a bounded number of q_s active attacks. For this, the use of the gap oracle is essential in the proof step that handles passive attacks. The strategy we used in the previous section does not use this strategy for clarity, but the same observations apply.

Lemma 7.3. *For every attacker $\mathcal{A}$, there exists an attacker $\mathcal{B}_6$, such that*

$$\Pr[\,\mathsf{G}_4 \Rightarrow \mathsf{bad}_4\,] \leq \mathsf{Adv}^{\mathcal{U}\text{-}\mathsf{GCDH}}_{\mathcal{B}_6}()$$

Proof. Let us consider the GCDH attacker $\mathcal{B}_6$ in Fig. 11. It gets a generalized challenge of the form $(X_1, \ldots, X_{q_e+q_s}, Y_1, \ldots, Y_{q_e+q_s})$ and finds $\mathsf{CDH}(X_i, Y_j)$, for some $1 \leq i, j \leq q_e$. This problem is tightly equivalent to $\mathcal{U}$-GCDH by random self-reducibility. The attacker embeds X_{k_x} and Y_{k_y} in all game-generated $X^\star$ and $Y^\star$, which means that it cannot compute the CDH for any session. However, it can use the DDH oracle to check when H queries for a given group element Z match traces where $\mathsf{CDH}(X, Y) = Z$. Additionally, the DDH oracle is used to check for the bad event, in which case a solution to the GCDH problem was found. □

Lemma 7.4. *For every attacker $\mathcal{A}$, there exist attackers $\mathcal{B}_7$ and $\mathcal{B}_8$, such that*

$$\Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}_8\,] \leq q_s \cdot \left(\mathsf{Adv}^{\mathsf{1PRF}}_{\mathcal{B}_7}() + \mathsf{Adv}^{\mathsf{1UF}}_{\mathcal{B}_8}()\right)$$

Proof. Consider the modified version of game G_8 we present in Figure 11 (game $G^c_{8'}$) and let us consider for now the behaviour of this game when $c = \mathsf{F}$. In this case, the only difference between the two games is that game $G^{\mathsf{F}}_{8'}$ only sets bad_8 in one special ℓ-th query to Send, while the original version of G_8 might set the flag in many Send queries. However, this special query is chosen uniformly at random and is outside of the adversary's view, so it must be the case that, conditioning on the probability that the randomly chosen ℓ correctly guesses the query that causes the first occurrence of bad_8 in game G_8, the probability of bad_8 occurring in the two games is the same. This implies,

$$\Pr[\,\mathsf{G}_8 \Rightarrow \mathsf{bad}_8\,] = \Pr[\,\mathsf{G}^{\mathsf{F}}_{8'} \Rightarrow \mathsf{bad}_8 \,|\ell \text{ is correct}] \leq q_s \Pr[\,\mathsf{G}^{\mathsf{F}}_{8'} \Rightarrow \mathsf{bad}_8\,]$$

We can now focus on game $\mathsf{G}_{8'}$. Observe that parameter $c = \mathsf{T}$ replaces the keys produced by the KDF with random ones for the ℓ-th query. This means that it is easy to construct an attacker $\mathcal{B}_{\mathsf{I}}$ such that

$$\mathsf{Adv}^{\mathsf{1PRF}}_{\mathcal{B}_7}() \;=\; \left|\Pr[\,\mathsf{G}^{\mathsf{F}}_{8'} \Rightarrow \mathsf{bad}_8\,] - \Pr[\,\mathsf{G}^{\mathsf{T}}_{8'} \Rightarrow \mathsf{bad}_8\,]\right|$$

We omit the details of this reduction. At this point, it suffices to bound the probability of bad_8 in game $\mathsf{G}^{\mathsf{T}}_{8'}$. We do this via a direct reduction to the one-time unforgeability property of the MAC scheme. This is shown in Figure 11 as attacker $\mathcal{B}_8$. Again the reduction is simple: for the critical ℓ-th query, the attacker fixes the authentication key that causes bad_8 implicitly as the external MAC key for which it is trying to forge. When bad_8 occurs, it returns the offending trace and authenticator as a forgery. □

8 Implications

Nothing-up-your-sleeve global parameters. Our security proof applies without change to any distribution of the global parameters (M, N) under which computing the CDH between the two with the help of a DDH oracle is assumed to be hard. This means that simply sampling (M, N) uniformly at random is a good choice when the GCDH assumption holds, and so is choosing $M = \mathsf{H}(K)$ and $N = \mathsf{H}(K')$ when H is modeled as a random oracle. Here the choice of $K \neq K'$ means one is relying on the standard GCDH assumption, whereas $K = K'$ leads to $M = N$, which means relying on the squared GCDH problem.

Removing the global parameters. Our proof extends to a variant of SPAKE2 where (M, N) are not generated in a global setup. Instead one can use $(M, N) = \mathsf{H}(U, S)$ and $N = \mathsf{H}(S, U)$, where different values of these parameters are precomputed for each user-server pair. The online efficiency of the protocol is not affected. Modeling H as a random oracle, this means that each (U, S) pair now poses an independent GCDH challenge to the active attacker. Our proof can be easily modified to cover this case with the same bound by observing that bad events that involve multiple sessions are always defined between sessions established for the same user-server (U, S).

Reusing passwords. Our model assumes that user passwords are sampled independently for each pair (U, S). The simple case of password reuse, where passwords are either repeats or they are sampled independently can be easily addressed by extending the corrupt oracle to exclude additional trivial attacks: it should declare as corrupt all (U', S') pairs that use the same password. The same proof applies and the bound is not affected in this case, as the entropy of non-repeat passwords is assumed to be unaffected.

The treatment of more complex distributions follows in the same lines as described in [KOY09, Section 2.2.1].

Acknowledgments. We would like to thank David Pointcheval for helpful discussions regarding the proof of Lemma 6.6 and Doreen Riepel for discussions that helped us clarify the proofs of Lemmas 6.5 and 6.6. This work was supported in part by the ERC Project aSCEND (H2020 639554) and by the French ANR ALAMBIC Project (ANR-16-CE39-0006). Manuel Barbosa was supported by grant SFRH/BSAB/143018/2018 awarded by FCT, Portugal.

References

ABC+06. M. Abdalla, E. Bresson, O. Chevassut, B. Möller, and D. Pointcheval. Provably secure password-based authentication in TLS. In *ASIACCS 06*, pages 35–45. ACM Press, March 2006.

ABM15. M. Abdalla, F. Benhamouda, and P. MacKenzie. Security of the J-PAKE password-authenticated key exchange protocol. In *2015 IEEE Symposium on Security and Privacy*, pages 571–587. IEEE Computer Society Press, May 2015.

ABR01. M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *CT-RSA 2001*, *LNCS* 2020, pages 143–158. Springer, Heidelberg, April 2001.

AFP05. M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. In *PKC 2005*, *LNCS* 3386, pages 65–84. Springer, Heidelberg, January 2005.

AFP06. M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. *IEE Proceedings — Information Security*, 153(1):27–39, March 2006.

AP05. M. Abdalla and D. Pointcheval. Simple password-based encrypted key exchange protocols. In *CT-RSA 2005*, *LNCS* 3376, pages 191–208. Springer, Heidelberg, February 2005.

BM92. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.

BOS18. J. Becerra, D. Ostrev, and M. Skrobot. Forward secrecy of SPAKE2. In *ProvSec 2018*, *LNCS* 11192, pages 366–384. Springer, Heidelberg, October 2018.

BPR00. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT 2000*, *LNCS* 1807, pages 139–155. Springer, Heidelberg, May 2000.

BR93. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

BR04. M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. http://eprint.iacr.org/2004/331.

FKL18. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *CRYPTO 2018, Part II*, *LNCS* 10992, pages 33–62. Springer, Heidelberg, August 2018.

HL18. B. Haase and B. Labrique. AuCPace: Efficient verifier-based PAKE protocol tailored for the IIoT. Cryptology ePrint Archive, Report 2018/286, 2018. https://eprint.iacr.org/2018/286.

HL19. B. Haase and B. Labrique. Aucpace: Efficient verifier-based pake protocol tailored for the iiot. *IACR TCHES*, 2019(2):1–48, 2019. https://tches.iacr.org/index.php/TCHES/article/view/7384.

HR10. F. Hao and P. Ryan. J-PAKE: Authenticated key exchange without PKI. Cryptology ePrint Archive, Report 2010/190, 2010. http://eprint.iacr.org/2010/190.

HS14. F. Hao and S. F. Shahandashti. The SPEKE protocol revisited. Cryptology ePrint Archive, Report 2014/585, 2014. http://eprint.iacr.org/2014/585.

Jab97. D. P. Jablon. Extended password key exchange protocols immune to dictionary attacks. In *6th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 1997)*, pages 248–255, Cambridge, MA, USA, June 18–20, 1997. IEEE Computer Society.

KOY03. J. Katz, R. Ostrovsky, and M. Yung. Forward secrecy in password-only key exchange protocols. In *SCN 02*, *LNCS* 2576, pages 29–44. Springer, Heidelberg, September 2003.

KOY09. J. Katz, R. Ostrovsky, and M. Yung. Efficient and secure authenticated key exchange using weak passwords. *J. ACM*, 57(1):3:1–3:39, 2009.

Kra05. H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In *CRYPTO 2005*, *LNCS* 3621, pages 546–566. Springer, Heidelberg, August 2005.

Kra10. H. Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In *CRYPTO 2010*, *LNCS* 6223, pages 631–648. Springer, Heidelberg, August 2010.

LK19. W. Ladd and B. Kaduk. SPAKE2, a PAKE. Internet-Draft draft-irtf-cfrg-spake2-09, IRTF, October 2019.

Los19. J. Loss. *New techniques for the modular analysis of digital signature schemes*. PhD Thesis, Ruhr-Universität Bochum, Universitätsbibliothek, 2019.

Mac01. P. MacKenzie. On the security of the SPEKE password-authenticated key exchange protocol. Cryptology ePrint Archive, Report 2001/057, 2001. http://eprint.iacr.org/2001/057.

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

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

A Games and adversaries for the proof of Theorems 6.1 and 6.2

Fig. 3. Security proof for SPAKE2. Games 1 to 4.

Fig. 4. Security proof for SPAKE2. Games 5 to 7.

G_8

proc Initialize()
$b \leftarrow\$ \{0,1\}$; $C \leftarrow \{\}$; $T \leftarrow \{\}$; $T_s \leftarrow \{\}$; $\mathsf{Tst} = \{\}$
$\mathsf{bad}_6 \leftarrow \mathsf{F}$; $\mathsf{bad}_7 \leftarrow \mathsf{F}$; $T_{\mathsf{bad}} \leftarrow \{\}$; $\mathsf{bad}_8^1 \leftarrow \mathsf{F}$; $\mathsf{bad}_8^2 \leftarrow \mathsf{F}$
$m \leftarrow\$ \mathbb{Z}_q^\star$; $n \leftarrow\$ \mathcal{D}^\star(m)$; $M \leftarrow g^m$; $N \leftarrow g^n$
Return (M, N)

proc SendInit(U, i, S)
If $\pi_U^i \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow M^x$
$\pi_U^i \leftarrow (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$
Return $(U, X^\star)$

proc SendResp$(S, i, U, X^\star)$
If $\pi_S^i \neq \perp$ Return $\perp$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow N^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$ Return $\perp$
$\mathsf{fr} \leftarrow (U, S) \notin C$
If $\neg\mathsf{fr}$
  $X \leftarrow X^\star \,/\, M^{\mathsf{pw}_{\mathrm{US}}}$; $\hat{y} \leftarrow ny - n\mathsf{pw}_{\mathrm{US}}$
  $K \leftarrow \mathsf{H}(U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, X^{\hat{y}})$
Else
  For all $(U, S, X^\star, Y^\star, \mathsf{pw}, Z) \in T$
    $X \leftarrow X^\star \,/\, M^{\mathsf{pw}}$; $\hat{y} \leftarrow ny - n\mathsf{pw}$
    If $Z = X^{\hat{y}}$ do $T_{\mathsf{bad}} \leftarrow T_{\mathsf{bad}} \cup \{U, S, X^\star, Y^\star, \mathsf{pw}, Z\}$
  $K \leftarrow\$ \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (S, y, K)$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{fr})$
Return $(S, Y^\star)$

proc SendTerm$(U, i, S, Y^\star)$
If $\pi_U^i \neq (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$ Return $\perp$
If $\exists P \in \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$ Return $\perp$
$\mathsf{fr} \leftarrow \exists j, (U, S, X^\star, Y^\star) = \pi_S^j.\mathsf{tr} \wedge \pi_S^j.\mathsf{fr} = \mathsf{T}$
$\mathsf{fr} \leftarrow \mathsf{fr} \vee (U, S) \notin C$
If $\neg\mathsf{fr}$
  $Y \leftarrow Y^\star \,/\, N^{\mathsf{pw}_{\mathrm{US}}}$; $\hat{x} \leftarrow mx - m\mathsf{pw}_{\mathrm{US}}$
  $K \leftarrow \mathsf{H}(U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Y^{\hat{x}})$
Else If $(U, S, X^\star, Y^\star) \in T_s$
  $(S, y, K) \leftarrow T_s[U, S, X^\star, Y^\star]$
Else
  For all $(U, S, X^\star, Y^\star, \mathsf{pw}, Z) \in T$
    $Y \leftarrow Y^\star \,/\, N^{\mathsf{pw}}$; $\hat{x} \leftarrow mx - m\mathsf{pw}$
    If $Z = Y^{\hat{x}}$ do $T_{\mathsf{bad}} \leftarrow T_{\mathsf{bad}} \cup \{U, S, X^\star, Y^\star, \mathsf{pw}, Z\}$
  $K \leftarrow\$ \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (U, x, K)$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{fr})$
Return T

proc Exec(U, S, i, j)
If $\pi_U^i \neq \perp \vee \pi_S^j \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow g^x$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow g^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$K \leftarrow\leftarrow\$ \mathcal{K}$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{T})$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{T})$
Return $(U, X^\star, S, Y^\star)$

proc H$(U, S, X^\star, Y^\star, \mathsf{pw}, Z)$
If $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] = K \neq \perp$ Return K
If $(U, S, X^\star, Y^\star) \in T_s$
  If $T_s[U, S, X^\star, Y^\star] = (U, x, K)$
    $Y \leftarrow Y^\star \,/\, N^{\mathsf{pw}}$; $\hat{x} \leftarrow mx - m\mathsf{pw}$; $Z' \leftarrow Y^{\hat{x}}$
  If $T_s[U, S, X^\star, Y^\star] = (S, y, K)$
    $X \leftarrow X^\star \,/\, M^{\mathsf{pw}}$; $\hat{y} \leftarrow my - n\mathsf{pw}$; $Z' \leftarrow X^{\hat{y}}$
  If $Z' = Z$:
    If $(U, S) \in C \wedge \mathsf{pw} = \mathsf{pw}_{\mathrm{US}}$ do $\mathsf{bad}_8^2 \leftarrow \mathsf{T}$
    If $(U, S) \notin C$ do $T_{\mathsf{bad}} \leftarrow T_{\mathsf{bad}} \cup \{U, S, X^\star, Y^\star, \mathsf{pw}, Z\}$
$T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] \leftarrow\$ \mathcal{K}$
Return $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z]$

proc Corrupt(U, S)
$C \leftarrow C \cup \{(U, S)\}$; $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
Return $\mathsf{pw}_{\mathrm{US}}$

proc Reveal(P, i)
If $\pi_P^i.\mathsf{ac} \neq \mathsf{T} \vee (P, i) \in \mathsf{Tst}$ Return $\perp$
$\forall (j, Q)$ s.t. $\pi_Q^j.\mathsf{tr} = \pi_P^i.\mathsf{tr}$ do $\pi_Q^j.\mathsf{fr} = \mathsf{F}$
Return $\pi_P^i.K$

proc Test(P, i)
If $\pi_P^i.\mathsf{fr} = \mathsf{F}$ Return $\perp$
$K_0 \leftarrow$ Reveal(P, i); $K_1 \leftarrow\$ \mathcal{K}$
$\mathsf{Tst} \leftarrow \mathsf{Tst} \cup (P, i)$; Return K_b

proc Finalize(b')
For $(U, S) \in (\mathcal{U} \times \mathcal{S}) \setminus C$ do $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
If $\exists\, U\, S, (\star, \mathsf{pw}_{\mathrm{US}}, \star) \in T_s$ do $\mathsf{bad}_7 \leftarrow \mathsf{T}$
If $\exists\, \mathsf{pw}, (\star, \mathsf{pw}, \star) \in T_s \wedge (\star, \star, \star, \star, \mathsf{pw}, \star) \in T$
  $\mathsf{bad}_7 \leftarrow \mathsf{T}$
If $\mathsf{bad}_7 = \mathsf{F}$
  If $\exists \mathsf{pw} \neq \mathsf{pw}'$,
    $(U, S, X^\star, Y^\star, \mathsf{pw}, Z) \in T_{\mathsf{bad}} \wedge$
    $(U, S, X^\star, Y^\star, \mathsf{pw}', Z') \in T_{\mathsf{bad}}$ do $\mathsf{bad}_8^1 \leftarrow \mathsf{T}$
  Else
    If $(U, S, \star, \star, \mathsf{pw}_{\mathrm{US}}, \star) \in T_{\mathsf{bad}}$ do $\mathsf{bad}_6 \leftarrow \mathsf{T}$
Else $\mathsf{bad}_8^2 \leftarrow \mathsf{F}$
Return $b = b'$

G_9

proc Initialize()
$b \leftarrow\$ \{0,1\}$; $C \leftarrow \{\}$; $T \leftarrow \{\}$; $T_s \leftarrow \{\}$; $\mathsf{Tst} = \{\}$
$\mathsf{bad}_7 \leftarrow \mathsf{F}$; $\mathsf{bad}_8^2 \leftarrow \mathsf{F}$
$m \leftarrow\$ \mathbb{Z}_q^\star$; $n \leftarrow\$ \mathcal{D}^\star(m)$; $M \leftarrow g^m$; $N \leftarrow g^n$
Return (M, N)

proc SendInit(U, i, S)
If $\pi_U^i \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow M^x$
$\pi_U^i \leftarrow (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$
Return $(U, X^\star)$

proc SendResp$(S, i, U, X^\star, \mathsf{alg})$
If $\pi_S^i \neq \perp$ Return $\perp$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow N^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$\mathsf{fr} \leftarrow (U, S) \notin C$
If $\neg\mathsf{fr}$
  $X \leftarrow X^\star \,/\, M^{\mathsf{pw}_{\mathrm{US}}}$; $\hat{y} \leftarrow ny - n\mathsf{pw}_{\mathrm{US}}$
  $K \leftarrow \mathsf{H}(U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, X^{\hat{y}})$
Else
  $K \leftarrow\$ \mathcal{K}$
  If $\exists (U, S, \star, \star, y, \star) \in T$ do $\mathsf{bad}_7 \leftarrow \mathsf{T}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (S, y, K, \mathsf{alg})$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{fr})$
Return $(S, Y^\star)$

proc SendTerm$(U, i, S, Y^\star, \mathsf{alg})$
If $\pi_U^i \neq (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$ Return $\perp$
If $\exists P \in \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$\mathsf{fr} \leftarrow \exists j, (U, S, X^\star, Y^\star) = \pi_S^j.\mathsf{tr} \wedge \pi_S^j.\mathsf{fr} = \mathsf{T}$
$\mathsf{fr} \leftarrow \mathsf{fr} \vee (U, S) \notin C$
If $\neg\mathsf{fr}$
  $Y \leftarrow Y^\star \,/\, N^{\mathsf{pw}_{\mathrm{US}}}$; $\hat{x} \leftarrow mx - m\mathsf{pw}_{\mathrm{US}}$
  $K \leftarrow \mathsf{H}(U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Y^{\hat{x}})$
Else If $(U, S, X^\star, Y^\star) \in T_s$
  $(S, y, K, \mathsf{alg}) \leftarrow T_s[U, S, X^\star, Y^\star]$
Else
  $K \leftarrow\$ \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (U, x, K, \mathsf{alg})$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{fr})$
Return T

proc Exec(U, S, i, j)
If $\pi_U^i \neq \perp \vee \pi_S^j \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow g^x$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow g^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$K \leftarrow\leftarrow\$ \mathcal{K}$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{T})$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{T})$
Return $(U, X^\star, S, Y^\star)$

proc H$(U, S, X^\star, Y^\star, \mathsf{pw}, Z)$
If $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] = K \neq \perp$ Return K
If $(U, S, X^\star, Y^\star) \in T_s$
  If $T_s[U, S, X^\star, Y^\star] = (U, x, K, \mathsf{alg})$
    $Y \leftarrow Y^\star \,/\, N^{\mathsf{pw}}$; $\hat{x} \leftarrow mx - m\mathsf{pw}$; $Z' \leftarrow Y^{\hat{x}}$
  If $T_s[U, S, X^\star, Y^\star] = (S, y, K, \mathsf{alg})$
    $X \leftarrow X^\star \,/\, M^{\mathsf{pw}}$; $\hat{y} \leftarrow my - n\mathsf{pw}$; $Z' \leftarrow X^{\hat{y}}$
  If $Z' = Z$:
    If $(U, S) \in C \wedge \mathsf{pw} = \mathsf{pw}_{\mathrm{US}}$ do $\mathsf{bad}_8^2 \leftarrow \mathsf{T}$
$T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] \leftarrow\$ \mathcal{K}$
Return $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z]$

proc Corrupt(U, S)
$C \leftarrow C \cup \{(U, S)\}$; $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
Return $\mathsf{pw}_{\mathrm{US}}$

proc Reveal(P, i)
If $\pi_P^i.\mathsf{ac} \neq \mathsf{T} \vee (P, i) \in \mathsf{Tst}$ Return $\perp$
$\forall (j, Q)$ s.t. $\pi_Q^j.\mathsf{tr} = \pi_P^i.\mathsf{tr}$ do $\pi_Q^j.\mathsf{fr} = \mathsf{F}$
Return $\pi_P^i.K$

proc Test(P, i)
If $\pi_P^i.\mathsf{fr} = \mathsf{F}$ Return $\perp$
$K_0 \leftarrow$ Reveal(P, i); $K_1 \leftarrow\$ \mathcal{K}$
$\mathsf{Tst} \leftarrow \mathsf{Tst} \cup (P, i)$; Return K_b

proc Finalize(b')
For $(U, S) \in (\mathcal{U} \times \mathcal{S}) \setminus C$ do $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
If $\exists\, U\, S, (\star, \mathsf{pw}_{\mathrm{US}}, \star) \in T_s$ do $\mathsf{bad}_7 \leftarrow \mathsf{T}$
If $\exists\, \mathsf{pw}, (\star, \mathsf{pw}, \star) \in T_s \wedge (\star, \star, \star, \star, \mathsf{pw}, \star) \in T$
  $\mathsf{bad}_7 \leftarrow \mathsf{T}$
If $\mathsf{bad}_7 = \mathsf{F}$ do $\mathsf{bad}_8^2 \leftarrow \mathsf{F}$
Return $b = b'$

G_{10}

proc Initialize()
$b \leftarrow\$ \{0,1\}$; $C \leftarrow \{\}$; $T \leftarrow \{\}$; $T_s \leftarrow \{\}$; $\mathsf{Tst} = \{\}$
$m \leftarrow\$ \mathbb{Z}_q^\star$; $n \leftarrow\$ \mathcal{D}^\star(m)$; $M \leftarrow g^m$; $N \leftarrow g^n$
$\mathsf{bad}_7 \leftarrow \mathsf{F}$; $\mathsf{bad}_8^2 \leftarrow \mathsf{F}$; $\mathsf{bad}_{10} \leftarrow \mathsf{F}$; $\lambda \leftarrow n/m$
Return (M, N)

proc SendInit(U, i, S)
If $\pi_U^i \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow M^x$
$\pi_U^i \leftarrow (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$
Return $(U, X^\star)$

proc SendResp$(S, i, U, X^\star, \mathsf{alg})$
If $\pi_S^i \neq \perp$ Return $\perp$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow N^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$\mathsf{fr} \leftarrow (U, S) \notin C$
If $\neg\mathsf{fr}$
  $X \leftarrow X^\star \,/\, M^{\mathsf{pw}_{\mathrm{US}}}$; $\hat{y} \leftarrow ny - n\mathsf{pw}_{\mathrm{US}}$
  $K \leftarrow \mathsf{H}(U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, X^{\hat{y}})$
Else
  $K \leftarrow\$ \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (S, y, K, \mathsf{alg})$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{fr})$
Return $(S, Y^\star)$

proc SendTerm$(U, i, S, Y^\star, \mathsf{alg})$
If $\pi_U^i \neq (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$ Return $\perp$
If $\exists P \in \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$\mathsf{fr} \leftarrow \exists j, (U, S, X^\star, Y^\star) = \pi_S^j.\mathsf{tr} \wedge \pi_S^j.\mathsf{fr} = \mathsf{T}$
$\mathsf{fr} \leftarrow \mathsf{fr} \vee (U, S) \notin C$
If $\neg\mathsf{fr}$
  $Y \leftarrow Y^\star \,/\, N^{\mathsf{pw}_{\mathrm{US}}}$; $\hat{x} \leftarrow mx - m\mathsf{pw}_{\mathrm{US}}$
  $K \leftarrow \mathsf{H}(U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Y^{\hat{x}})$
Else If $(U, S, X^\star, Y^\star) \in T_s$
  $(S, y, K, \mathsf{alg}) \leftarrow T_s[U, S, X^\star, Y^\star]$
Else
  $K \leftarrow\$ \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (U, x, K, \mathsf{alg})$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{fr})$
Return T

proc Exec(U, S, i, j)
If $\pi_U^i \neq \perp \vee \pi_S^j \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow g^x$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow g^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$K \leftarrow\leftarrow\$ \mathcal{K}$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{T})$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{T})$
Return $(U, X^\star, S, Y^\star)$

proc H$(U, S, X^\star, Y^\star, \mathsf{pw}, Z)$
If $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] = K \neq \perp$ Return K
If $(U, S, X^\star, Y^\star) \in T_s$
  If $T_s[U, S, X^\star, Y^\star] = (U, x, K, \mathsf{alg})$
    $Y \leftarrow Y^\star \,/\, N^{\mathsf{pw}}$; $\hat{x} \leftarrow mx - m\mathsf{pw}$; $Z' \leftarrow Y^{\hat{x}}$
  If $T_s[U, S, X^\star, Y^\star] = (S, y, K, \mathsf{alg})$
    $X \leftarrow X^\star \,/\, M^{\mathsf{pw}}$; $\hat{y} \leftarrow my - n\mathsf{pw}$; $Z' \leftarrow X^{\hat{y}}$
  If $Z' = Z \wedge \mathsf{bad}_{10} = \mathsf{F}$:
    If $(U, S) \in C \wedge \mathsf{pw} = \mathsf{pw}_{\mathrm{US}}$ do $\mathsf{bad}_8^2 \leftarrow \mathsf{T}$
$T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] \leftarrow\$ \mathcal{K}$
Return $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z]$

proc Corrupt(U, S)
$C \leftarrow C \cup \{(U, S)\}$; $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
For all $(U, S, X^\star, Y^\star) \in T_s$
  If $T_s[U, S, X^\star, Y^\star] = (U, x, K, \mathsf{alg})$
    Rewrite $\mathsf{alg} = [(g, a), (M, b), (N, c)]$
    If $b - \mathsf{pw}_{\mathrm{US}} + \lambda c = 0$ do $\mathsf{bad}_{10} \leftarrow \mathsf{T}$
  If $T_s[U, S, X^\star, Y^\star] = (S, y, K, \mathsf{alg})$
    Rewrite $\mathsf{alg} = [(g, \star), (M, b), (N, \star)]$
    If $b + \lambda(c - \mathsf{pw}_{\mathrm{US}}) = 0$ do $\mathsf{bad}_{10} \leftarrow \mathsf{T}$
Return $\mathsf{pw}_{\mathrm{US}}$

proc Reveal(P, i)
If $\pi_P^i.\mathsf{ac} \neq \mathsf{T} \vee (P, i) \in \mathsf{Tst}$ Return $\perp$
$\forall (j, Q)$ s.t. $\pi_Q^j.\mathsf{tr} = \pi_P^i.\mathsf{tr}$ do $\pi_Q^j.\mathsf{fr} = \mathsf{F}$
Return $\pi_P^i.K$

proc Test(P, i)
If $\pi_P^i.\mathsf{fr} = \mathsf{F}$ Return $\perp$
$K_0 \leftarrow$ Reveal(P, i); $K_1 \leftarrow\$ \mathcal{K}$
$\mathsf{Tst} \leftarrow \mathsf{Tst} \cup (P, i)$; Return K_b

proc Finalize(b')
For $(U, S) \in (\mathcal{U} \times \mathcal{S}) \setminus C$ do $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
If $\exists\, U\, S, (\star, \mathsf{pw}_{\mathrm{US}}, \star) \in T_s$ do $\mathsf{bad}_7 \leftarrow \mathsf{T}$
If $\exists\, \mathsf{pw}, (\star, \mathsf{pw}, \star) \in T_s \wedge (\star, \star, \star, \star, \mathsf{pw}, \star) \in T$
  $\mathsf{bad}_7 \leftarrow \mathsf{T}$
If $\mathsf{bad}_7 = \mathsf{F}$ do $\mathsf{bad}_8^2 \leftarrow \mathsf{F}$
Return $b = b'$

Fig. 5. Security proof for SPAKE2. Games 8 to 10.

Fig. 6. Security proof for SPAKE2. GCDH and GDL attackers.

B Games and adversary for the proof of Theorem 7.1

ADVERSARY $\mathcal{B}_4$

proc INITIALIZE(M)
$b \leftarrow\$ \{0,1\}$; $C \leftarrow \{\}$; $T \leftarrow \{\}$; $T_s \leftarrow \{\}$; $\mathsf{Tst} = \{\}$
$\mathsf{bad}_7 \leftarrow \mathsf{F}$; $\mathsf{bad}_{10} \leftarrow \mathsf{F}$
$m \leftarrow\$ \mathbb{Z}_q$; $n \leftarrow\$ \mathcal{D}^\star(m)$; $\lambda \leftarrow n/m$; $N \leftarrow M^\lambda$
Return (M, N)

proc SENDINIT(U, i, S)
If $\pi_U^i \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow M^x$
$\pi_U^i \leftarrow (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$
Return $(U, X^\star)$

proc SENDRESP($S, i, U, X^\star, \mathsf{alg}$)
If $\pi_S^i \neq \perp$ Return $\perp$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow N^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$\mathsf{fr} \leftarrow (U, S) \notin C$
If $\neg\mathsf{fr}$
  $X \leftarrow X^\star / M^{\mathsf{pw}_{\mathrm{US}}}$; $Y \leftarrow Y^\star / N^{\mathsf{pw}_{\mathrm{US}}}$
  If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z) \in T$ s.t. $\mathsf{DDH}(X, Y, Z) = \mathsf{T}$
    $K \leftarrow T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z]$
  Else $K \leftarrow\$ \mathcal{K}$; $T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, (X, Y)] \leftarrow K$
Else
  $K \leftarrow\$ \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (S, y, K, \mathsf{alg})$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{fr})$
Return $(S, Y^\star)$

proc SENDTERM($U, i, S, Y^\star, \mathsf{alg}$)
If $\pi_U^i \neq (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$ Return $\perp$
If $\exists P \in \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$\mathsf{fr} \leftarrow \exists j, (U, S, X^\star, Y^\star) = \pi_S^j.\mathsf{tr} \wedge \pi_S^j.\mathsf{fr} = \mathsf{T}$
$\mathsf{fr} \leftarrow \mathsf{fr} \vee (U, S) \notin C$
If $\neg\mathsf{fr}$
  $X \leftarrow X^\star / M^{\mathsf{pw}_{\mathrm{US}}}$; $Y \leftarrow Y^\star / N^{\mathsf{pw}_{\mathrm{US}}}$
  If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z) \in T$ s.t. $\mathsf{DDH}(X, Y, Z) = \mathsf{T}$
    $K \leftarrow T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z]$
  Else $K \leftarrow\$ \mathcal{K}$; $T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, (X, Y)] \leftarrow K$
Else If $(U, S, X^\star, Y^\star) \in T_s$
  $(S, y, K, \mathsf{alg}') \leftarrow T_s[U, S, X^\star, Y^\star]$
Else
  $K \leftarrow\$ \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (U, x, K, \mathsf{alg})$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{fr})$
Return T

proc EXEC(U, S, i, j)
Unchanged

proc H($U, S, X^\star, Y^\star, \mathsf{pw}, Z$)
If $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] = K \neq \perp$ Return K
If $(U, S, X^\star, Y^\star) \in T_s$
  $X \leftarrow X^\star / M^{\mathsf{pw}}$; $Y \leftarrow Y^\star / N^{\mathsf{pw}}$
  If $\mathsf{DDH}(X, Y, Z) = \mathsf{T} \wedge (U, S) \in C \wedge \mathsf{pw} = \mathsf{pw}_{\mathrm{US}} \wedge \mathsf{bad}_{10} = \mathsf{F}$
    Call FINALIZE($\mathsf{CDH}(M, M)$)
If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}, (X, Y) \in T$ s.t. $\mathsf{DDH}(X, Y, Z) = \mathsf{T}$
  Return $T[U, S, X^\star, Y^\star, \mathsf{pw}, (X, Y)]$
$T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] \leftarrow\$ \mathcal{K}$
Return $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z]$

proc CORRUPT(U, S)
$C \leftarrow C \cup \{(U, S)\}$; $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
For all $(U, S, X^\star, Y^\star) \in T_s$
  If $T_s[U, S, X^\star, Y^\star] = (U, x, K, \mathsf{alg})$
    Rewrite $\mathsf{alg} = [(g, a), (M, b), (N, c)]$
    If $b - \mathsf{pw}_{\mathrm{US}} + \lambda c = 0$ do $\mathsf{bad}_{10} = \mathsf{T}$
  If $T_s[U, S, X^\star, Y^\star] = (S, y, K, \mathsf{alg})$
    Rewrite $\mathsf{alg} = [(g, a), (M, b), (N, c)]$
    If $b + \lambda(c - \mathsf{pw}_{\mathrm{US}}) = 0$ do $\mathsf{bad}_{10} = \mathsf{T}$
Return $\mathsf{pw}_{\mathrm{US}}$

proc REVEAL(P, i)
If $\pi_P^i.\mathsf{ac} \neq \mathsf{T} \vee (P, i) \in \mathsf{Tst}$ Return $\perp$
$\forall (j, Q)$ s.t. $\pi_Q^j.\mathsf{tr} = \pi_P^i.\mathsf{tr}$ do $\pi_Q^j.\mathsf{fr} = \mathsf{F}$
Return $\pi_P^i.K$

proc TEST(P, i)
If $\pi_P^i.\mathsf{fr} = \mathsf{F}$ Return $\perp$
$K_0 \leftarrow$ REVEAL(P, i); $K_1 \leftarrow\$ \mathcal{K}$
$\mathsf{Tst} \leftarrow \mathsf{Tst} \cup (P, i)$; Return K_b

proc FINALIZE(b')
For $(U, S) \in (\mathcal{U} \times \mathcal{S}) \setminus C$ do $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
If $\exists\, U\, S$, $(\star, \mathsf{pw}_{\mathrm{US}}, \star) \in T_s$ do $\mathsf{bad}_7 \leftarrow \mathsf{T}$
If $\exists\, \mathsf{pw}, (\star, \mathsf{pw}, \star) \in T_s \wedge (\star, \star, \star, \star, \mathsf{pw}, \star) \in T$
  $\mathsf{bad}_7 \leftarrow \mathsf{T}$
Stop.

ADVERSARY $\mathcal{B}_5^{\mathsf{F},i}$

proc INITIALIZE(B_1, B_2, A_1)
$b \leftarrow\$ \{0,1\}$; $C \leftarrow \{\}$; $T \leftarrow \{\}$; $T_s \leftarrow \{\}$; $\mathsf{Tst} = \{\}$
$\mathsf{bad}_7 \leftarrow \mathsf{F}$
$M \leftarrow B_1$; $N \leftarrow B_2$; $k \leftarrow 0$; $\mathsf{tr} \leftarrow \perp$
Return (M, N)

proc SENDINIT(U, i, S)
If $\pi_U^i \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow A_1 \cdot g^x$
$\pi_U^i \leftarrow (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$
$k \leftarrow k + 1$; Return $(U, X^\star)$

proc SENDRESP($S, i, U, X^\star, \mathsf{alg}$)
If $\pi_S^i \neq \perp$ Return $\perp$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow N^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$\mathsf{fr} \leftarrow (U, S) \notin C$
If $\neg\mathsf{fr}$
  $X \leftarrow X^\star / M^{\mathsf{pw}_{\mathrm{US}}}$; $Y \leftarrow Y^\star / N^{\mathsf{pw}_{\mathrm{US}}}$
  If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z) \in T$ s.t. $\mathsf{DDH}(X, Y, Z) = \mathsf{T}$
    $K \leftarrow T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z]$
  Else $K \leftarrow\$ \mathcal{K}$; $T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, (X, Y)] \leftarrow K$
Else
  $K \leftarrow\$ \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (S, y, K, \mathsf{alg})$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{fr})$
$k \leftarrow k + 1$; Return $(S, Y^\star)$

proc SENDTERM($U, i, S, Y^\star, \mathsf{alg}$)
If $\pi_U^i \neq (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$ Return $\perp$
If $\exists P \in \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$\mathsf{fr} \leftarrow \exists j, (U, S, X^\star, Y^\star) = \pi_S^j.\mathsf{tr} \wedge \pi_S^j.\mathsf{fr} = \mathsf{T}$
$\mathsf{fr} \leftarrow \mathsf{fr} \vee (U, S) \notin C$
If $\neg\mathsf{fr}$
  $X \leftarrow X^\star / M^{\mathsf{pw}_{\mathrm{US}}}$; $Y \leftarrow Y^\star / N^{\mathsf{pw}_{\mathrm{US}}}$
  If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z) \in T$ s.t. $\mathsf{DDH}(X, Y, Z) = \mathsf{T}$
    $K \leftarrow T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z]$
  Else $K \leftarrow\$ \mathcal{K}$; $T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, (X, Y)] \leftarrow K$
Else If $(U, S, X^\star, Y^\star) \in T_s$
  $(S, y, K, \mathsf{alg}') \leftarrow T_s[U, S, X^\star, Y^\star]$
Else
  If $k = i$
    $\mathsf{tr} \leftarrow (U, S, X^\star, Y^\star)$
    Output $Y^\star$ to get $\mathsf{pw}_{U,S}$
  $K \leftarrow\$ \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (U, x, K, \mathsf{alg})$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{fr})$
$k \leftarrow k + 1$; Return T

proc EXEC(U, S, i, j)
If $\pi_U^i \neq \perp \vee \pi_S^j \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow g^x$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow g^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$K \leftarrow\leftarrow\$ \mathcal{K}$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{T})$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{T})$
Return $(U, X^\star, S, Y^\star)$

proc H($U, S, X^\star, Y^\star, \mathsf{pw}, Z$)
If $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] = K \neq \perp$ Return K
If $(U, S, X^\star, Y^\star) \in T_s$
  $X \leftarrow X^\star / M^{\mathsf{pw}}$; $Y \leftarrow Y^\star / N^{\mathsf{pw}}$
  If $\mathsf{DDH}(X, Y, Z) = \mathsf{T} \wedge (U, S) \in C \wedge \mathsf{pw} = \mathsf{pw}_{\mathrm{US}}$
    If $\mathsf{tr} \neq (U, S, X^\star, Y^\star)$ do Abort
    Else Get x from T_s; Output $Z / M^{x - \mathsf{pw}_{\mathrm{US}}}$
If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}, (X, Y) \in T$ s.t. $\mathsf{DDH}(X, Y, Z) = \mathsf{T}$
  Return $T[U, S, X^\star, Y^\star, \mathsf{pw}, (X, Y)]$
$T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] \leftarrow\$ \mathcal{K}$
Return $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z]$

proc CORRUPT(U, S)
$C \leftarrow C \cup \{(U, S)\}$
If $\mathsf{tr} \neq (U, S, \star, \star)$ do $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
Return $\mathsf{pw}_{\mathrm{US}}$

proc REVEAL(P, i)
If $\pi_P^i.\mathsf{ac} \neq \mathsf{T} \vee (P, i) \in \mathsf{Tst}$ Return $\perp$
$\forall (j, Q)$ s.t. $\pi_Q^j.\mathsf{tr} = \pi_P^i.\mathsf{tr}$ do $\pi_Q^j.\mathsf{fr} = \mathsf{F}$
Return $\pi_P^i.K$

proc TEST(P, i)
If $\pi_P^i.\mathsf{fr} = \mathsf{F}$ Return $\perp$
$K_0 \leftarrow$ REVEAL(P, i); $K_1 \leftarrow\$ \mathcal{K}$
$\mathsf{Tst} \leftarrow \mathsf{Tst} \cup (P, i)$; Return K_b

proc FINALIZE(b')
For $(U, S) \in (\mathcal{U} \times \mathcal{S}) \setminus C$ do $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
If $\exists\, U\, S$, $(\star, \mathsf{pw}_{\mathrm{US}}, \star) \in T_s$ do $\mathsf{bad}_7 \leftarrow \mathsf{T}$
If $\exists\, \mathsf{pw}, (\star, \mathsf{pw}, \star) \in T_s \wedge (\star, \star, \star, \star, \mathsf{pw}, \star) \in T$
  $\mathsf{bad}_7 \leftarrow \mathsf{T}$
Stop.

ADVERSARY $\mathcal{B}_5^{\mathsf{T},i}$

proc INITIALIZE(B_1, B_2, A_1)
$b \leftarrow\$ \{0,1\}$; $C \leftarrow \{\}$; $T \leftarrow \{\}$; $T_s \leftarrow \{\}$; $\mathsf{Tst} = \{\}$
$\mathsf{bad}_7 \leftarrow \mathsf{F}$
$M \leftarrow B_2$; $N \leftarrow B_1$; $k \leftarrow 0$; $\mathsf{tr} \leftarrow \perp$
Return (M, N)

proc SENDINIT(U, i, S)
If $\pi_U^i \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow M^x$
$\pi_U^i \leftarrow (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$
$k \leftarrow k + 1$; Return $(U, X^\star)$

proc SENDRESP($S, i, U, X^\star, \mathsf{alg}$)
If $\pi_S^i \neq \perp$ Return $\perp$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow N^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$\mathsf{fr} \leftarrow (U, S) \notin C$
If $\neg\mathsf{fr}$
  $X \leftarrow X^\star / M^{\mathsf{pw}_{\mathrm{US}}}$; $Y \leftarrow Y^\star / N^{\mathsf{pw}_{\mathrm{US}}}$
  If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z) \in T$ s.t. $\mathsf{DDH}(X, Y, Z) = \mathsf{T}$
    $K \leftarrow T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z]$
  Else $K \leftarrow\$ \mathcal{K}$; $T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, (X, Y)] \leftarrow K$
Else
  If $k = i$
    $Y^\star \leftarrow A_1$; $\mathsf{tr} \leftarrow (U, S, X^\star, Y^\star)$
    Output $X^\star$ to get $\mathsf{pw}_{U,S}$
  $K \leftarrow\$ \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (S, y, K, \mathsf{alg})$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{fr})$
$k \leftarrow k + 1$; Return $(S, Y^\star)$

proc SENDTERM($U, i, S, Y^\star, \mathsf{alg}$)
If $\pi_U^i \neq (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$ Return $\perp$
If $\exists P \in \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$\mathsf{fr} \leftarrow \exists j, (U, S, X^\star, Y^\star) = \pi_S^j.\mathsf{tr} \wedge \pi_S^j.\mathsf{fr} = \mathsf{T}$
$\mathsf{fr} \leftarrow \mathsf{fr} \vee (U, S) \notin C$
If $\neg\mathsf{fr}$
  $X \leftarrow X^\star / M^{\mathsf{pw}_{\mathrm{US}}}$; $Y \leftarrow Y^\star / N^{\mathsf{pw}_{\mathrm{US}}}$
  If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z) \in T$ s.t. $\mathsf{DDH}(X, Y, Z) = \mathsf{T}$
    $K \leftarrow T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z]$
  Else $K \leftarrow\$ \mathcal{K}$; $T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, (X, Y)] \leftarrow K$
Else If $(U, S, X^\star, Y^\star) \in T_s$
  $(S, y, K, \mathsf{alg}') \leftarrow T_s[U, S, X^\star, Y^\star]$
Else
  $K \leftarrow\$ \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (U, x, K, \mathsf{alg})$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{fr})$
$k \leftarrow k + 1$; Return T

proc EXEC(U, S, i, j)
If $\pi_U^i \neq \perp \vee \pi_S^j \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow g^x$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow g^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$K \leftarrow\leftarrow\$ \mathcal{K}$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{T})$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K, \mathsf{T}, \mathsf{T})$
Return $(U, X^\star, S, Y^\star)$

proc H($U, S, X^\star, Y^\star, \mathsf{pw}, Z$)
If $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] = K \neq \perp$ Return K
If $(U, S, X^\star, Y^\star) \in T_s$
  $X \leftarrow X^\star / M^{\mathsf{pw}}$; $Y \leftarrow Y^\star / N^{\mathsf{pw}}$
  If $\mathsf{DDH}(X, Y, Z) = \mathsf{T} \wedge (U, S) \in C \wedge \mathsf{pw} = \mathsf{pw}_{\mathrm{US}}$
    If $\mathsf{tr} \neq (U, S, X^\star, Y^\star)$ do Abort
    Else Output Z
If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}, (X, Y) \in T$ s.t. $\mathsf{DDH}(X, Y, Z) = \mathsf{T}$
  Return $T[U, S, X^\star, Y^\star, \mathsf{pw}, (X, Y)]$
$T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] \leftarrow\$ \mathcal{K}$
Return $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z]$

proc CORRUPT(U, S)
$C \leftarrow C \cup \{(U, S)\}$
If $\mathsf{tr} \neq (U, S, \star, \star)$ do $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
Return $\mathsf{pw}_{\mathrm{US}}$

proc REVEAL(P, i)
If $\pi_P^i.\mathsf{ac} \neq \mathsf{T} \vee (P, i) \in \mathsf{Tst}$ Return $\perp$
$\forall (j, Q)$ s.t. $\pi_Q^j.\mathsf{tr} = \pi_P^i.\mathsf{tr}$ do $\pi_Q^j.\mathsf{fr} = \mathsf{F}$
Return $\pi_P^i.K$

proc TEST(P, i)
If $\pi_P^i.\mathsf{fr} = \mathsf{F}$ Return $\perp$
$K_0 \leftarrow$ REVEAL(P, i); $K_1 \leftarrow\$ \mathcal{K}$
$\mathsf{Tst} \leftarrow \mathsf{Tst} \cup (P, i)$; Return K_b

proc FINALIZE(b')
For $(U, S) \in (\mathcal{U} \times \mathcal{S}) \setminus C$ do $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
If $\exists\, U\, S$, $(\star, \mathsf{pw}_{\mathrm{US}}, \star) \in T_s$ do $\mathsf{bad}_7 \leftarrow \mathsf{T}$
If $\exists\, \mathsf{pw}, (\star, \mathsf{pw}, \star) \in T_s \wedge (\star, \star, \star, \star, \mathsf{pw}, \star) \in T$
  $\mathsf{bad}_7 \leftarrow \mathsf{T}$
Stop.

Fig. 7. Security proof for SPAKE2. Algebraic reduction to GCDH and alternative reduction to GPCCDH.

Fig. 8. Security proof for SPAKE2 with confirmation. Games 1 to 4.

G_5

proc Initialize( )
$b \leftarrow\$ \{0,1\}$; $C \leftarrow \{\}$; $T \leftarrow \{\}$; $\mathsf{Tst} = \{\}$
For $U \in \mathcal{U}$, $S \in \mathcal{S}$ do $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$
$m \leftarrow\$ \mathbb{Z}_q^\star$; $n \leftarrow\$ \mathcal{D}^\star(m)$; $M \leftarrow g^m$; $N \leftarrow g^n$
Return (M, N)

proc SendInit(U, i, S)
If $\pi_U^i \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow g^x \cdot M^{\mathsf{pw}_{\mathrm{US}}}$
$\pi_U^i \leftarrow (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$
Return $(U, X^\star)$

proc SendResp$(S, i, U, X^\star)$
If $\pi_S^i \neq \perp$ Return $\perp$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow g^y \cdot N^{\mathsf{pw}_{\mathrm{US}}}$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$ Return $\perp$
If $\exists j, \pi_U^j = (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$
  $(K_a, K_e) \leftarrow\$ \mathcal{K} \times \mathcal{K}$
Else
  $X \leftarrow X^\star / M^{\mathsf{pw}_{\mathrm{US}}}$
  $(K_a, K_e) \leftarrow \mathsf{H}(U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, X^y)$
$(K_{U,S}, K_{S,U}) \leftarrow \mathsf{KDF}(K_a, \mathsf{const})$
$\mathsf{tag} \leftarrow \mathsf{MAC}(K_{S,U}, (U, S, X^\star, Y^\star))$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), (K_a, K_e), \mathsf{F}, \mathsf{F})$
Return $(S, Y^\star, \mathsf{tag})$

proc SendTermInit$(U, i, S, Y^\star, \mathsf{tag})$
If $\pi_U^i \neq (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$ Return $\perp$
If $\exists P \in \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$ Return $\perp$
$\mathsf{fr} \leftarrow \exists j, (U, S, X^\star, Y^\star) = \pi_S^j.\mathsf{tr} \wedge \pi_S^j.\mathsf{fr} = \mathsf{T}$
$\mathsf{fr} \leftarrow \mathsf{fr} \vee (U, S) \notin C$
If $\exists j, \pi_S^j = (y, (U, S, X^\star, Y^\star), (K_a, K_e), \mathsf{F}, \mathsf{F})$
  $(K_a, K_e) \leftarrow \pi_S^j.K$
Else
  $Y \leftarrow Y^\star / N^{\mathsf{pw}_{\mathrm{US}}}$
  $(K_a, K_e) \leftarrow \mathsf{H}(U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, X^y)$
$(K_{U,S}, K_{S,U}) \leftarrow \mathsf{KDF}(K_a, \mathsf{const})$
If $\mathsf{tag} \neq \mathsf{MAC}(K_{S,U}, (U, S, X^\star, Y^\star))$
  $\pi_U^i \leftarrow \mathsf{aborted}$; Return $\perp$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K_e, \mathsf{T}, \mathsf{fr})$
$\mathsf{tag} \leftarrow \mathsf{MAC}(K_{U,S}, (U, S, X^\star, Y^\star))$
Return tag

proc SendTermResp(S, i, U, tag)
If $\pi_S^i \neq (y, (U, S, X^\star, Y^\star), (K_a, K_e), \mathsf{F}, \mathsf{F})$ Return $\perp$
$\mathsf{fr} \leftarrow \exists j, (U, S, X^\star, Y^\star) = \pi_U^j.\mathsf{tr} \wedge \pi_U^j.\mathsf{fr} = \mathsf{T}$
$\mathsf{fr} \leftarrow \mathsf{fr} \vee (U, S) \notin C$
$(K_{U,S}, K_{S,U}) \leftarrow \mathsf{KDF}(K_a, \mathsf{const})$
If $\mathsf{tag} \neq \mathsf{MAC}(K_{U,S}, (U, S, X^\star, Y^\star))$
  $\pi_S^i \leftarrow \mathsf{aborted}$; Return $\perp$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K_e, \mathsf{T}, \mathsf{fr})$
Return T

proc Exec(U, S, i, j)
If $\pi_U^i \neq \perp \vee \pi_S^j \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow g^x$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow g^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$
  Return $\perp$
$(K_a, K_e) \leftarrow\$ \mathcal{K} \times \mathcal{K}$
$(K_{U,S}, K_{S,U}) \leftarrow \mathsf{KDF}(K_a, \mathsf{const})$
$\mathsf{tag}_1 \leftarrow \mathsf{MAC}(K_{U,S}, (U, S, X^\star, Y^\star))$
$\mathsf{tag}_2 \leftarrow \mathsf{MAC}(K_{S,U}, (U, S, X^\star, Y^\star))$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K_e, \mathsf{T}, \mathsf{T})$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K_e, \mathsf{T}, \mathsf{T})$
Return $(U, X^\star, S, Y^\star, \mathsf{tag}_1, \mathsf{tag}_2)$

proc H$(U, S, X^\star, Y^\star, \mathsf{pw}, Z)$
If $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] = K \neq \perp$ Return K
$T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] \leftarrow\$ \mathcal{K} \times \mathcal{K}$
Return $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z]$

proc Corrupt(U, S)
$C \leftarrow C \cup \{(U, S)\}$
Return $\mathsf{pw}_{\mathrm{US}}$

proc Reveal(P, i)
If $\pi_P^i.\mathsf{ac} \neq \mathsf{T} \vee (P, i) \in \mathsf{Tst}$ Return $\perp$
$\forall (j, Q)$ s.t. $\pi_Q^j.\mathsf{tr} = \pi_P^i.\mathsf{tr}$ do $\pi_Q^j.\mathsf{fr} = \mathsf{F}$
Return $\pi_P^i.K$

proc Test(P, i)
If $\pi_P^i.\mathsf{fr} = \mathsf{F}$ Return $\perp$
$K_0 \leftarrow$ Reveal(P, i); $K_1 \leftarrow\$ \mathcal{K}$
$\mathsf{Tst} \leftarrow \mathsf{Tst} \cup (P, i)$; Return K_b

proc Finalize(b')
Return $b = b'$

G_6

proc Initialize( )
$b \leftarrow\$ \{0,1\}$; $C \leftarrow \{\}$; $T \leftarrow \{\}$; $\mathsf{Tst} = \{\}$; $T_s \leftarrow \{\}$
For $U \in \mathcal{U}$, $S \in \mathcal{S}$ do $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$; $\mathsf{bad}_6 \leftarrow \mathsf{F}$
$m \leftarrow\$ \mathbb{Z}_q^\star$; $n \leftarrow\$ \mathcal{D}^\star(m)$; $M \leftarrow g^m$; $N \leftarrow g^n$
Return (M, N)

proc SendInit(U, i, S)
If $\pi_U^i \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow g^x \cdot M^{\mathsf{pw}_{\mathrm{US}}}$
$\pi_U^i \leftarrow (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$
Return $(U, X^\star)$

proc SendResp$(S, i, U, X^\star)$
If $\pi_S^i \neq \perp$ Return $\perp$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow g^y \cdot N^{\mathsf{pw}_{\mathrm{US}}}$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$ Return $\perp$
If $\exists j, \pi_U^j = (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$
  $(K_a, K_e) \leftarrow\$ \mathcal{K} \times \mathcal{K}$
Else If $(U, S) \in C$
  $X \leftarrow X^\star / M^{\mathsf{pw}_{\mathrm{US}}}$
  $(K_a, K_e) \leftarrow \mathsf{H}(U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, X^y)$
Else
  If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}, Z) \in T \wedge \mathsf{pw} = \mathsf{pw}_{\mathrm{US}}$
    $X \leftarrow X^\star / M^{\mathsf{pw}}$
    If $Z = X^y$ do $\mathsf{bad}_6 \leftarrow \mathsf{T}$
  $(K_a, K_e) \leftarrow\$ \mathcal{K} \times \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (S, y, (K_a, K_e))$
$(K_{U,S}, K_{S,U}) \leftarrow \mathsf{KDF}(K_a, \mathsf{const})$
$\mathsf{tag} \leftarrow \mathsf{MAC}(K_{S,U}, (U, S, X^\star, Y^\star))$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), (K_a, K_e), \mathsf{F}, \mathsf{F})$
Return $(S, Y^\star, \mathsf{tag})$

proc SendTermInit$(U, i, S, Y^\star, \mathsf{tag})$
If $\pi_U^i \neq (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$ Return $\perp$
If $\exists P \in \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$ Return $\perp$
$\mathsf{fr} \leftarrow \exists j, (U, S, X^\star, Y^\star) = \pi_S^j.\mathsf{tr} \wedge \pi_S^j.\mathsf{fr} = \mathsf{T}$
$\mathsf{fr} \leftarrow \mathsf{fr} \vee (U, S) \notin C$
If $\exists j, \pi_S^j = (y, (U, S, X^\star, Y^\star), (K_a, K_e), \mathsf{F}, \mathsf{F})$
  $(K_a, K_e) \leftarrow \pi_S^j.K$
Else If $\neg\mathsf{fr}$
  $Y \leftarrow Y^\star / N^{\mathsf{pw}_{\mathrm{US}}}$
  $(K_a, K_e) \leftarrow \mathsf{H}(U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Y^x)$
Else (cannot be in T_s)
  If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}, Z) \in T \wedge \mathsf{pw} = \mathsf{pw}_{\mathrm{US}}$
    $Y \leftarrow Y^\star / N^{\mathsf{pw}}$
    If $Z = Y^x$ do $\mathsf{bad}_6 \leftarrow \mathsf{T}$
  $K_a \times K_e \leftarrow\$ \mathcal{K} \times \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (U, x, (K_a, K_e))$
$(K_{U,S}, K_{S,U}) \leftarrow \mathsf{KDF}(K_a, \mathsf{const})$
If $\mathsf{tag} \neq \mathsf{MAC}(K_{S,U}, (U, S, X^\star, Y^\star))$
  $\pi_U^i \leftarrow \mathsf{aborted}$; Return $\perp$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K_e, \mathsf{T}, \mathsf{fr})$
$\mathsf{tag} \leftarrow \mathsf{MAC}(K_{U,S}, (U, S, X^\star, Y^\star))$
Return tag

proc SendTermResp(S, i, U, tag)
If $\pi_S^i \neq (y, (U, S, X^\star, Y^\star), (K_a, K_e), \mathsf{F}, \mathsf{F})$ Return $\perp$
$\mathsf{fr} \leftarrow \exists j, (U, S, X^\star, Y^\star) = \pi_U^j.\mathsf{tr} \wedge \pi_U^j.\mathsf{fr} = \mathsf{T}$
$\mathsf{fr} \leftarrow \mathsf{fr} \vee (U, S) \notin C$
$(K_{U,S}, K_{S,U}) \leftarrow \mathsf{KDF}(K_a, \mathsf{const})$
If $\mathsf{tag} \neq \mathsf{MAC}(K_{U,S}, (U, S, X^\star, Y^\star))$
  $\pi_S^i \leftarrow \mathsf{aborted}$; Return $\perp$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K_e, \mathsf{T}, \mathsf{fr})$
Return T

proc H$(U, S, X^\star, Y^\star, \mathsf{pw}, Z)$
If $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] = K \neq \perp$ Return K
If $(U, S, X^\star, Y^\star) \in T_s \wedge \mathsf{pw} = \mathsf{pw}_{\mathrm{US}}$
  If $T_s[U, S, X^\star, Y^\star] = (U, x, K)$
    $Y \leftarrow Y^\star / N^{\mathsf{pw}}$; $Z' \leftarrow Y^x$
  If $T_s[U, S, X^\star, Y^\star] = (S, y, K)$
    $X \leftarrow X^\star / M^{\mathsf{pw}}$; $Z' \leftarrow X^y$
  If $Z' = Z$ do $\mathsf{bad}_6 \leftarrow \mathsf{T}$
$T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] \leftarrow\$ \mathcal{K} \times \mathcal{K}$
Return $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z]$

proc Corrupt(U, S)
$C \leftarrow C \cup \{(U, S)\}$
For all $(U, S, X^\star, Y^\star) \in T_s$
  If $\nexists\, j\ P, \pi_P^j.\mathsf{tr} = (U, S, X^\star, Y^\star) \wedge \pi_P^j.\mathsf{ac} = \mathsf{T}$
    If $T_s[U, S, X^\star, Y^\star] = (U, x, K)$
      $Y \leftarrow Y^\star / N^{\mathsf{pw}_{\mathrm{US}}}$; $Z \leftarrow Y^x$
    If $T_s[U, S, X^\star, Y^\star] = (S, y, K)$
      $X \leftarrow X^\star / M^{\mathsf{pw}_{\mathrm{US}}}$; $Z \leftarrow X^y$
    $T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z] \leftarrow (K_a, K_e)$
    $T_s[U, S, X^\star, Y^\star] \leftarrow \perp$
Return $\mathsf{pw}_{\mathrm{US}}$

proc Finalize(b')

Return $b = b'$

procs Exec, Reveal, Test unchanged

G_7

proc Initialize( )
$b \leftarrow\$ \{0,1\}$; $C \leftarrow \{\}$; $T \leftarrow \{\}$; $\mathsf{Tst} = \{\}$; $T_s \leftarrow \{\}$
For $U \in \mathcal{U}$, $S \in \mathcal{S}$ do $\mathsf{pw}_{\mathrm{US}} \leftarrow\$ \mathcal{P}$; $\mathsf{bad}_6 \leftarrow \mathsf{F}$; $\mathsf{bad}_7 \leftarrow \mathsf{F}$
$m \leftarrow\$ \mathbb{Z}_q^\star$; $n \leftarrow\$ \mathcal{D}^\star(m)$; $M \leftarrow g^m$; $N \leftarrow g^n$
Return (M, N)

proc SendInit(U, i, S)
If $\pi_U^i \neq \perp$ Return $\perp$
$x \leftarrow\$ \mathbb{Z}_q$; $X^\star \leftarrow M^x$
$\pi_U^i \leftarrow (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$
Return $(U, X^\star)$

proc SendResp$(S, i, U, X^\star)$
If $\pi_S^i \neq \perp$ Return $\perp$
$y \leftarrow\$ \mathbb{Z}_q$; $Y^\star \leftarrow N^y$
If $\exists P \in \mathcal{S} \cup \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$ Return $\perp$
If $\exists j, \pi_U^j = (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$
  $(K_a, K_e) \leftarrow\$ \mathcal{K} \times \mathcal{K}$
Else If $(U, S) \in C$
  $X \leftarrow X^\star / M^{\mathsf{pw}_{\mathrm{US}}}$; $\hat{y} \leftarrow ny - n\mathsf{pw}_{\mathrm{US}}$
  $(K_a, K_e) \leftarrow \mathsf{H}(U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, X^{\hat{y}})$
Else
  If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}, Z) \in T \wedge \mathsf{pw} = \mathsf{pw}_{\mathrm{US}}$
    $X \leftarrow X^\star / M^{\mathsf{pw}}$; $\hat{y} \leftarrow ny - n\mathsf{pw}$
    If $Z = X^{\hat{y}}$ do $\mathsf{bad}_6 \leftarrow \mathsf{T}$
  $(K_a, K_e) \leftarrow\$ \mathcal{K} \times \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (S, y, (K_a, K_e))$
$(K_{U,S}, K_{S,U}) \leftarrow \mathsf{KDF}(K_a, \mathsf{const})$
$\mathsf{tag} \leftarrow \mathsf{MAC}(K_{S,U}, (U, S, X^\star, Y^\star))$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), (K_a, K_e), \mathsf{F}, \mathsf{F})$
Return $(S, Y^\star, \mathsf{tag})$

proc SendTermInit$(U, i, S, Y^\star, \mathsf{tag})$
If $\pi_U^i \neq (x, (U, S, X^\star, \perp), \perp, \mathsf{F}, \mathsf{F})$ Return $\perp$
If $\exists P \in \mathcal{U}, (U, S, X^\star, Y^\star) = \pi_P^j.\mathsf{tr}$ Return $\perp$
$\mathsf{fr} \leftarrow \exists j, (U, S, X^\star, Y^\star) = \pi_S^j.\mathsf{tr} \wedge \pi_S^j.\mathsf{fr} = \mathsf{T}$
$\mathsf{fr} \leftarrow \mathsf{fr} \vee (U, S) \notin C$
If $\exists j, \pi_S^j = (y, (U, S, X^\star, Y^\star), (K_a, K_e), \mathsf{F}, \mathsf{F})$
  $(K_a, K_e) \leftarrow \pi_S^j.K$
Else If $\neg\mathsf{fr}$
  $Y \leftarrow Y^\star / N^{\mathsf{pw}_{\mathrm{US}}}$; $\hat{x} \leftarrow mx - m\mathsf{pw}_{\mathrm{US}}$
  $(K_a, K_e) \leftarrow \mathsf{H}(U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Y^{\hat{x}})$
Else (cannot be in T_s)
  If $\exists (U, S, X^\star, Y^\star, \mathsf{pw}, Z) \in T \wedge \mathsf{pw} = \mathsf{pw}_{\mathrm{US}}$
    $Y \leftarrow Y^\star / N^{\mathsf{pw}}$; $\hat{x} \leftarrow mx - m\mathsf{pw}$
    If $Z = Y^{\hat{x}}$ do $\mathsf{bad}_6 \leftarrow \mathsf{T}$
  $K_a \times K_e \leftarrow\$ \mathcal{K} \times \mathcal{K}$
  $T_s[U, S, X^\star, Y^\star] \leftarrow (U, x, (K_a, K_e))$
$(K_{U,S}, K_{S,U}) \leftarrow \mathsf{KDF}(K_a, \mathsf{const})$
If $\mathsf{tag} \neq \mathsf{MAC}(K_{S,U}, (U, S, X^\star, Y^\star))$
  $\pi_U^i \leftarrow \mathsf{aborted}$; Return $\perp$
$\pi_U^i \leftarrow (x, (U, S, X^\star, Y^\star), K_e, \mathsf{T}, \mathsf{fr})$
$\mathsf{tag} \leftarrow \mathsf{MAC}(K_{U,S}, (U, S, X^\star, Y^\star))$
Return tag

proc SendTermResp(S, i, U, tag)
If $\pi_S^i \neq (y, (U, S, X^\star, Y^\star), (K_a, K_e), \mathsf{F}, \mathsf{F})$ Return $\perp$
$\mathsf{fr} \leftarrow \exists j, (U, S, X^\star, Y^\star) = \pi_U^j.\mathsf{tr} \wedge \pi_U^j.\mathsf{fr} = \mathsf{T}$
$\mathsf{fr} \leftarrow \mathsf{fr} \vee (U, S) \notin C$
$(K_{U,S}, K_{S,U}) \leftarrow \mathsf{KDF}(K_a, \mathsf{const})$
If $\mathsf{tag} \neq \mathsf{MAC}(K_{U,S}, (U, S, X^\star, Y^\star))$
  $\pi_S^i \leftarrow \mathsf{aborted}$; Return $\perp$
$\pi_S^i \leftarrow (y, (U, S, X^\star, Y^\star), K_e, \mathsf{T}, \mathsf{fr})$
Return T

proc H$(U, S, X^\star, Y^\star, \mathsf{pw}, Z)$
If $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] = K \neq \perp$ Return K
If $(U, S, X^\star, Y^\star) \in T_s \wedge \mathsf{pw} = \mathsf{pw}_{\mathrm{US}}$
  If $T_s[U, S, X^\star, Y^\star] = (U, x, K)$
    $Y \leftarrow Y^\star / N^{\mathsf{pw}}$; $\hat{x} \leftarrow mx - m\mathsf{pw}$; $Z' \leftarrow Y^{\hat{x}}$
  If $T_s[U, S, X^\star, Y^\star] = (S, y, K)$
    $X \leftarrow X^\star / M^{\mathsf{pw}}$; $\hat{y} \leftarrow ny - n\mathsf{pw}$; $Z' \leftarrow X^{\hat{y}}$
  If $Z' = Z$ do $\mathsf{bad}_6 \leftarrow \mathsf{T}$
$T[U, S, X^\star, Y^\star, \mathsf{pw}, Z] \leftarrow\$ \mathcal{K} \times \mathcal{K}$
Return $T[U, S, X^\star, Y^\star, \mathsf{pw}, Z]$

proc Corrupt(U, S)
$C \leftarrow C \cup \{(U, S)\}$
For all $(U, S, X^\star, Y^\star) \in T_s$
  If $\nexists\, j\ P, \pi_P^j.\mathsf{tr} = (U, S, X^\star, Y^\star) \wedge \pi_P^j.\mathsf{ac} = \mathsf{T}$
    If $T_s[U, S, X^\star, Y^\star] = (U, x, K)$
      $Y \leftarrow Y^\star / N^{\mathsf{pw}_{\mathrm{US}}}$; $\hat{x} \leftarrow mx - m\mathsf{pw}$; $Z \leftarrow Y^{\hat{x}}$
    If $T_s[U, S, X^\star, Y^\star] = (S, y, K)$
      $X \leftarrow X^\star / M^{\mathsf{pw}_{\mathrm{US}}}$; $\hat{y} \leftarrow ny - n\mathsf{pw}$; $Z \leftarrow X^{\hat{y}}$
    $T[U, S, X^\star, Y^\star, \mathsf{pw}_{\mathrm{US}}, Z] \leftarrow (K_a, K_e)$
    $T_s[U, S, X^\star, Y^\star] \leftarrow \perp$
Return $\mathsf{pw}_{\mathrm{US}}$

proc Finalize(b')
If $\exists\, U\ S, (\star, \mathsf{pw}_{\mathrm{US}}, \star) \in T_s$ do $\mathsf{bad}_7 \leftarrow \mathsf{T}$
If $\exists\, \mathsf{pw}, (\star, \mathsf{pw}, \star) \in T_s \wedge (\star, \star, \star, \star, \mathsf{pw}, \star) \in T$ do $\mathsf{bad}_7 \leftarrow \mathsf{T}$
If $\mathsf{bad}_7 = \mathsf{T}$ do $\mathsf{bad}_6 \leftarrow \mathsf{F}$
Return $b = b'$

procs Exec, Reveal, Test unchanged

Fig. 9. Security proof for SPAKE2 with confirmation. Games 5 to 7.

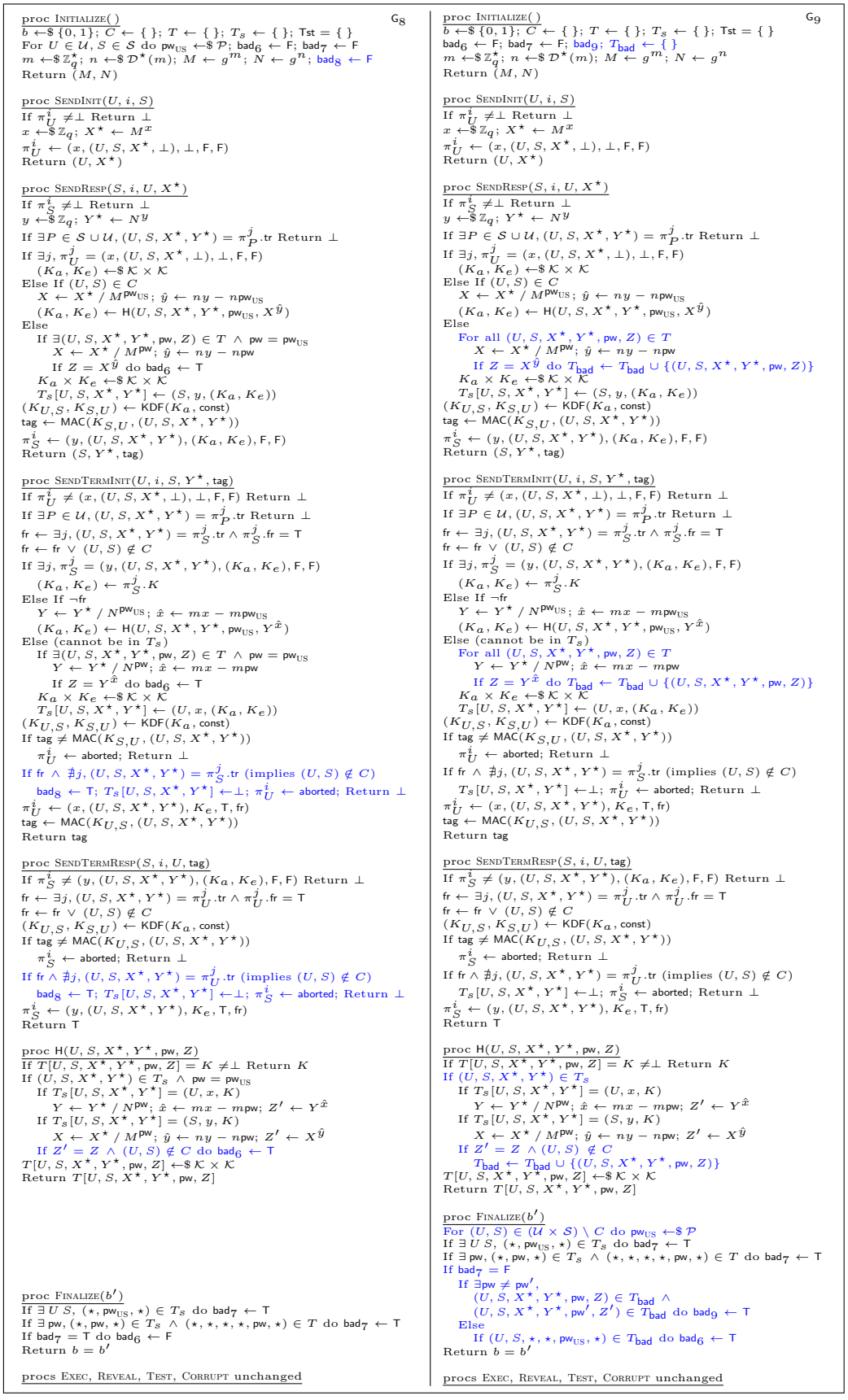

Fig. 10. Security proof for SPAKE2 with confirmation. Games 8 to 9.

Fig. 11. Security proof for SPAKE2 with confirmation. Query guess game and adversaries.