On the Security of One Password Authenticated Key Exchange Protocol

Stanislav V. Smyshlyaev*
Igor B. Oshkin†
Evgeniy K. Alekseev‡
Liliya R. Ahmetzyanova§

Abstract

In this paper the Security Evaluated Standardized Password Authenticated Key Exchange (*SESPAKE*) protocol is proposed (this protocol is approved in the standardization system of the Russian Federation) and its cryptographic properties are analyzed. The *SESPAKE* protocol includes a key agreement step and a key authentication step. We define new indistinguishability-based adversary model with a threat of false authentication that is an extension of the original indistinguishability-based model up to the case of protocols with authentication step without key diversification. We prove the protocol security under two types of threats: a classic threat of distinguishing a generated session key from a random string and a threat of false authentication. This protocol is the first password authenticated key exchange protocol (*PAKE*) protocol without key diversification for a full version of which a security proof has been obtained. The paper also contains a brief review of the known results dedicated to analysis of cryptographic properties of *PAKE* protocols.

*Ph.D., Head of Information Security Department, CryptoPro LLC, Russia; smyshsv@gmail.com
†Ph.D., Deputy Head of Information Security Dept., CryptoPro LLC, Russia; oshkin@cryptopro.ru
‡Ph.D., Lead Engineer-Analyst, CryptoPro LLC, Russia; alekseev@cryptopro.ru
§Engineer-Analyst, CryptoPro LLC, Russia; lah@cryptopro.ru

Contents

1 Introduction

The main point of the type of protocols considered in this paper consists in the fact that parties sharing a weak key (a password) generate a strong common key. The main requirement for these protocols implies the active adversary who has an access to a channel isn't able to obtain any information that can be used to find a key in «offline» mode, i.e. without interaction with legitimate participants. Usually such authentication key exchange protocols are called «EKE» — «Encrypted Key Exchange» or «PAKE» — «Password-Authenticated Key Exchange».

The first protocol called *PAKE* was proposed by S. Bellovin and M. Merritt in [19] in 1992 . There are a lot of protocols of this type (e.g. see [12], [13], [15], [18], [17]). In 1993 M. Bellare and P. Rogaway proposed *the indistinguishability-based model* (see [20]) that was developed by M. Bellare, D. Pointcheval и P. Rogaway in [11]. This term was introduced by M. Abdalla in [10]. This model allowed to obtain the security bounds for some protocols of this type.

The *PAKE* protocols include a key agreement step and sometimes also key authentication step. The lower bounds have been obtained as for the protocols without authentication step (see [2]), as for some protocols that include this step (see [15]). For both cases the security has been proven under a threat of distinguishing a generated session key from a random string. But this threat can't be defined correctly for all protocols with authentication step. In the case when authentication step uses a generated key, the adversary trivially obtains a criterion that allows to check whether the target string is random or not. J. Bender, M. Fischlin, D. Kugler point to it explicitly (see [28]). In the papers where bounds for the protocols with authentication step were obtained, the problem with a correct definition of a threat is solved with an additional key diversification. It means that a key for the authentication step is obtained from the common secret with one one-way function and the main key is obtained from the same secret with another one-way function.

We stress that consideration of security under another type of threat — the false authentication threat — is also a very significant part of security proofs for protocols with authentication step. As an example we will consider a protocol PACE (see [28]) which security proof doesn't take into account the threat of false authentication. This protocol is vulnerable to a reflection attack.

It should be noticed also that a commonly considered threat of distinguishing a key may seem not a quite natural one, as it is defined regardless to a protocol execution context. A target key is often used in further interaction between subjects, e.g. for organization of a secure messaging channel. The key exploitation gives a trivial criterion for distinguishing it from a random string.

In this paper we propose the Security Evaluated Standardized Password Authenticated Key Exchange (*SESPAKE*) protocol (this protocol is approved in the standardization system of the Russian Federation (see [8])). There is no key diversification mentioned above, i.e. the authentication step is processed using a generated key. The key agreement step is a modification of the protocol proposed in [2], the authentication step is original. A threat of false authentication is proposed for this protocol and a bound is obtained under this threat (so this protocol is resistant to a reflection attack mentioned above). The proof is divided into three parts: the first one is based on the ideas proposed in [2], the second and the third

steps are original. The protocol proposed in this paper is the first password authenticated key exchange protocol without key diversification for a full version of which a security proof has been obtained.

Also there is a review of some *PAKE* protocols and models used to prove their security.

2 Known models, protocols and bounds

This section contains a brief review of the results that are devoted to analysis of cryptographic properties of *PAKE* protocols. We focus on adversary models used to prove protocol security.

In the paper [10] M. Abdalla considers several adversary models: an indistinguishability-based model, a universal-composable model and a simulation-based model. As we use the indistinguishability-based model to analyze the security of the proposed protocol, we consider this adversary model in a more detailed way.

2.1 The indistinguishability-based model

As it has been said above, in 1993 M. Bellare and P. Rogaway proposed *the indistinguishability-based model* (see [20]) that was later expanded by M. Bellare, D. Pointcheval and P. Rogaway in [11].

Most results about the *PAKE* protocol security belong to a sphere of practice oriented provable security ([25]). A feature of this approach is the fact that it allows to analyze security of particular protocols that do not have infinite security settings. This approach uses a technique that is common for the complexity theory that performs reduction of some task to another one, and the result of this analysis becomes an upper bound of some adversary abilities (usually, a success probability to realize a threat).

The *PAKE* protocol assumes an interaction of two parties called *protocol users*. The user behavior is modelled by the oracle called *user instance*. The adversary can interact with the legitimate users using requests to the special oracles that imitate their possibilities in a real attack.

There are *concurrent* and *non-concurrent* security models. The *concurrent* model assumes that several copies of the protocol can be processed concurrently, therefore several instances can be active simultaneously. For the *non-concurrent* model at most one user instance can be active per user.

Denote by U^i the i-th user U instance. There are five oracles that are used to model the abilities of the adversary: $\mathcal{O}_{exec}$, $\mathcal{O}_{send}$, $\mathcal{O}_{reveal}$, $\mathcal{O}_{corrupt}$, $\mathcal{O}_{test}$.

- $\mathcal{O}_{exec}$. This oracle models a passive attack where the adversary can initiate the interaction between two legitimate user instances U_1^i and U_2^j and listen to their protocol execution making a request (U_1^i, U_2^j) to this oracle. The oracle returns all messages sent by users to each other during the protocol execution.

- $\mathcal{O}_{send}$. This oracle models an active attack where the adversary can intercept messages, modify them, create new ones and send to other user instances, making requests

(U^i, m) to this oracle. The oracle returns a message that user instance U^i sends in response to the message m.

- $\mathcal{O}_{reveal}$. This oracle models a case of insecure usage of a session key. This oracle returns a session key of the user instance U^i, if it is defined.

- $\mathcal{O}_{test}$. By making a request (U^i) to this oracle we check the ability of the adversary to distinguish a session key from a random string. The oracle chooses a bit $b \in_R \{0, 1\}$ and returns a session key of the user instance U^i if $b = 1$, and a random string of the same length if $b = 0$.

Sometimes in addition to the indistinguishability-based model the protocol security is considered in the *random-oracle model* or the *ideal-cipher model*. These models assume idealized properties of some hash-functions and ciphers used in a protocol — a hash-function is replaced with a random mapping and a cipher is replaced with a random permutation indexed by a key. In this case the oracles $\mathcal{O}_{hash}$ or $\mathcal{O}_{cipher}$ are additionally introduced. One of them will be considered more thoroughly in Section 3.

Let us consider the following concepts: *Freshness* and *Partnering*.

Partnering. Denote by $\mathbf{sid_U^i}$ a session identifier that is a concatenation of all messages sent and received by a user instance during the protocol execution. Denote by $\mathbf{pid_U^i}$ a partner identifier that consists of all user instances that U^i wants to interact with and itself. The user instances U_1^i и U_2^j are *partners* if and only if their session and partner identifiers are equal.

Freshness. The user instance is *fresh* if no requests were made to the oracle $\mathcal{O}_{reveal}$ for it or its partner.

2.2 The Protocols and the threats

In this section we describe two different types of threats and consider some cryptographic properties of the following protocols:

- **One-Encryption Key-Exchange (OEKE/AuthA)**

  The AuthA protocol was proposed by M. Bellare and P. Rogaway in 2000 in [12]. In 2003 E. Bresson, O. Chevassut and D. Pointcheval analyzed security of the simple protocol version OEKE (see Table. 1) and the original protocol AuthA in the concurrent indistinguishability-based model with ideal-cipher model and random-oracle model (see [13]). The AuthA protocol is the IEEE standard (P1363.2: Standard Specifications for Password-Based Public-Key Cryptographic Techniques).

  The AuthA protocol includes a key agreement step and a key authentication step.

- **Password Authenticated Key Exchange (PAK)**

  The PAK protocol (see Table. 2) was proposed by V. Boyko, P. MacKenzie and S. Patel in 2000 in [15]. In 2002 P. MacKenzie analyzed security of the PAK protocol in the concurrent indistinguishability-based model with random-oracle model (see [16]). The PAK protocol is used in Lucent Technologies' Plan9 operation systems. It is the IEEE standard P1363.2 too.

The PAK protocol includes a key agreement step and a key authentication step.

- **Password Authenticated Connection Establishment (PACE)**

  The PACE protocol (see Table. 3) was proposed by the German Federal Office for Information Security (BSI) for the deployment in machine readable travel documents in 2008 (see [28]). In 2009 J. Bender, M. Fischlin, D. Kugler analyzed protocol security in the concurrent indistinguishability-based model with random-oracle model and ideal cipher model (see [21]). The protocol is currently under standardization of ISO/IEC JTC1/SC17/WG3.

  The PACE protocol includes a key agreement step and a key authentication step.

- **DragonFly**

  The Dragonfly protocol (see Table 5) is specified by Dan Harkins for exchanging session keys with mutual authentication within mesh networks (see [22]). This protocol has been submitted to the Internet Engineering Task Force as a candidate standard for general internet use. In 2015 security of a very close variant of DragonFly was analyzed by J. Lancrenon and M. Scrobot in the standard model with random oracle (see [24]).

  The Dragonfly protocol includes a key agreement step and a key authentication step.

- **Strong Password-Only Authenticated Key Exchange (SPEKE)**

  The SPEKE protocol (see Table 8) was proposed by D. Jablon in 1996 (see [26]). In 2001 P. MacKenzie analyzed security of the SPEKE protocol in the indistinguishability-based model with random-oracle model (see [27]).

  The SPEKE protocol includes a key agreement step and a key authentication step.

- **Simple Password-Based Encrypted Key Exchange (SPAKE2)**

  The SPAKE2 protocol (see Table 4) and its analysis were proposed by M. Abdalla and D. Pointcheval in 2005 (see [2]). They proved protocol security in two models: the non-concurrent (see protocol SPAKE1 (without pw as input for hash-function)) and concurrent (see protocol SPAKE2) indistinguishability-based models with random oracle. The work described in [2] has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

  The SPAKE2 protocol includes a key agreement step only.

- **Password Authenticated Key Exchange by Juggling (J-PAKE)**.

  The J-PAKE protocol (see Table 6) was proposed by F. Hao and P. Ryan in 2010 (see [33]). In 2015 M. Abdalla, F. Benhamouda, P. MacKenzie analyzed security of the J-PAKE protocol in the indistinguishability-based model with random-oracle model (see [34]).

  The J-PAKE protocol includes a key agreement step and a key authentication step.

- **Augmented Password Authenticated Key Exchange (AugPAKE)**

  The AugPAKE protocol (see Table 7) was proposed by S. Shin and K. Kobara in 2008 (see [30]). In 2010 the same authors analyzed security of the AugPAKE protocol in

the indistinguishability-based models with random-oracle model (see [31]). AugPAKE protocol was proposed as one of the most efficient protocols.

The AugPAKE protocol includes a key agreement step and a key authentication step.

A classic threat for the considered model is a *threat of distinguishing a session key from a random string.*

- We consider an adversary who has access to the oracles $\mathcal{O}_{exec}$, $\mathcal{O}_{send}$, $\mathcal{O}_{reveal}$, $\mathcal{O}_{test}$ (+ $\mathcal{O}_{hash}$, $\mathcal{O}_{cipher}$). The adversary can make a single request to the oracle $\mathcal{O}_{test}$ for *fresh* user instance which has already generated a session key and outputs a bit b'. The adversary abilities are defined by the number of requests to each oracle. Suppose that passwords are chosen uniformly from the dictionary $\mathcal{D}$. A password-based key agreement protocol is secure in the indistinguishability-based model if the probability of the event that occurs if a random bit b chosen by $\mathcal{O}_{test}$ and a bit b' are equal negligibly differs from $O(q/|Dict|) + 1/2$, where q is a number of user instances the adversary makes a request to the oracle $\mathcal{O}_{send}$ for.

The protocols are divided into two types by the authentication step format :

Type 1 Generated session key is used to compute some authentication data. In this case the key that is the target of the adversary is the input of the one-way function used to obtain the authentication data (e.g. SESPAKE).

Type 2 Generated session key is not used to compute any authentication data. It means that a key for the authentication step is obtained from the common secret with one one-way function and the main key is obtained from the same common secret with another one-way function (e.g. PACE, AugPAKE).

For protocols of the first type the adversary model considered above is not correct, as the adversary trivially obtains a criterion that allows to check whether the target string is random or not by computing authentication data with this key. J. Bender, M. Fischlin, D. Kugler point to it explicitly (see [28]). There are papers with proved security for the first step only among such protocols considered by the authors.

For protocols of the second type the adversary model mentioned above is correct. Consider the theorem on the security under a threat of distinguishing a session key from a random string for the **protocol OEKE**.

Denote by $\mathbf{Adv}^{ake}_{oeke}(\mathcal{A}) = 2 \cdot \Pr[b = b'] - 1$ the advantage of the adversary $\mathcal{A}$ in the bit b recognition task. l_1 is the length of the hash function $\mathcal{H}_1$ output. $\mathbf{Succ}^{\mathbf{CDH}}_{\mathbb{G}}(t')$ is the probability of the event that occurs if the adversary can solve the computational Diffie-Hellman problem with at most t' steps.

Theorem 2.1. *Let us consider the **OEKE** protocol, where SK is the session-key space and $Password$ is a finite dictionary of size N equipped with uniform distribution. Let $\mathcal{A}$ be an adversary against the security under a threat of distinguishing a session key from a random string of **OEKE** within a time bound t, with less than q_s queries to the oracle $\mathcal{O}_{send}$,*

q_p queries to the oracle $\mathcal{O}_{exec}$, and, asking q_h hash queries and q_e encryption/decryption queries. Thus we have

$$\mathbf{Adv}^{ake}_{oeke}(\mathcal{A}) \leq 3 \cdot \frac{q_s}{N} + 8q_h \cdot \mathbf{Succ}^{\mathbf{cdh}}_{\mathbb{G}}(t') + \frac{(2q_e + 3q_s + 3q_p)^2}{q-1} + \frac{q_h^2 + 4q_s}{2^{l_1}},$$

where $t' \leq t + (q_s + q_p + q_e + 1) \cdot \tau_{\mathbb{G}}$, with $\tau_{\mathbb{G}}$ denoting the computational time for an exponentiation in $\mathbb{G}$, q is the order of $\mathbb{G}$.

This theorem proves that the protocol is secure against a *dictionary attack*.

Public Information: G, g, p, q		
A (Client): pw		B (Server): pw
$x \in_R \{1, \ldots, q-1\},\ X \equiv g^x$	$\xrightarrow{A, X}$	$y \in_R \{1, \ldots, q-1\},\ Y \equiv g^y$
$Y \equiv D_{pw}(Y^*)$	$\xleftarrow{B, Y^*}$	$Y^* \equiv E_{pw}(Y)$
$K_A \equiv Y^x$		$K_B \equiv X^y$
$T_A = \mathcal{H}_1(A\|\|B\|\|X\|\|Y\|\|K_A)$	$\xrightarrow{T_A}$	if $T_A \neq \mathcal{H}_1(A\|\|B\|\|X\|\|Y\|\|K_B)$ reject
$sk_A = \mathcal{H}_0(A\|\|B\|\|X\|\|Y\|\|K_A)$		$sk_B = \mathcal{H}_0(A\|\|B\|\|X\|\|Y\|\|K_B)$

Table 1: Protocol OEKE

Also for protocols of the second type the *threat of the false authentication* is considered based on the same adversary model.

In the considered papers the analysis of the protocol security under a false authentication threat is based on the theorems on the security under a threat of key distinguishing. Informally the theorem on the security under the classic threat that is proved using all protocol steps assumes that if the adversary can distinguish the key from the random string with significant probability, then with a significant probability he knows the correct input data for the function used to compute the final session key. It means that he can use the same input data to compute the correct authentication information. Therefore, the methods used in these papers to prove a security under the false authentication threat can be considered as correct. There are different formal definitions of the false authentication threat in different papers.

The OEKE protocol.

$\mathbf{Adv}^{c-auth}_{oeke}(\mathcal{A}) = \Pr[\mathsf{Auth}_0]$ is the probability of the event that occurs if the adversary could impersonate a legitimate participant, where Auth_0 is defined in the following way: the event Auth_0 occurs if $\mathcal{A}$ submits an authenticator $Auth$ that will be accepted by the server and that has been built by the adversary itself in real attack game in the random oracle and ideal-cipher models.

The absence of the «built by the adversary itself» formal definition leads to the adversary model that is defined not quite strictly.

The PAK protocol

The adversary model is defined in the following way: $\mathbf{Adv}^{\mathbf{ma}}_P(\mathcal{A})$ is the probability of the event that occurs if there is a user instance that successfully terminated a protocol execution and was not corrupted with the request to the oracle $\mathcal{O}_{corrupt}$ (this oracle returns a password of the user instance), but he has no partners.

Public Information: G, g, p, q		
A (Client): π_A $x \in_R \mathbb{Z}_q,\ \alpha \equiv g^x$ $\gamma \equiv \mathcal{H}_1(\pi),\ m \equiv \alpha \cdot \gamma$		B (Server): $\pi_B[A] = \langle(\mathcal{H}_1(\pi_A))^{-1}\rangle$
	$\xrightarrow{A, m}$	
		Abort if not acceptable(m) $y \in_R \mathbb{Z}_q,\ \mu \equiv g^y$ $\gamma' \equiv \pi_B(A),\ \alpha \equiv m \cdot \gamma'$ $\sigma \equiv \alpha^y$ $k = \mathcal{H}_2(A\|\|B\|\|m\|\|\mu\|\|\sigma\|\|\gamma')$ $k'' = \mathcal{H}_3(A\|\|B\|\|m\|\|\mu\|\|\sigma\|\|\gamma')$
	$\xleftarrow{\mu, k}$	
$\sigma \equiv \mu^x,\ \gamma' \equiv (\gamma)^{-1}$ Abort if $k \neq \mathcal{H}_2(A\|\|B\|\|m\|\|\mu\|\|\sigma\|\|\gamma')$ $k' = \mathcal{H}_3(A\|\|B\|\|m\|\|\mu\|\|\sigma\|\|\gamma')$		
	$\xrightarrow{k'}$	
		Abort if $k \neq k'$
$sk = \mathcal{H}_4(A\|\|B\|\|m\|\|\mu\|\|\sigma\|\|\gamma')$		

Table 2: The PAK protocol

Theorem 2.2. *Let PAK protocol uses group $\mathbb{G}_q$ and a password dictionary of size N. Fix an adversary $\mathcal{A}$ that runs in time t and makes q_s queries to the oracle $\mathcal{O}_{send}$, q_e queries to the oracle $\mathcal{O}_{exec}$, q_r queries to the oracle $\mathcal{O}_{reveal}$, q_c queries to the oracles $\mathcal{O}_{corrupt}$ and q_h hash queries to the random oracle. Then for $t' = O(t + (q_h^2 + q_s + q_e) \cdot \tau_{\mathbb{G}}$:*

$$\mathbf{Adv}_P^{\mathbf{ma}}(\mathcal{A}) = \frac{q_s}{N} + O\left(q_s \cdot \mathbf{Succ}_{\mathbb{G}}^{\mathbf{CDH}}(t') + \frac{(q_s + q_e)(q_h + q_s + q_e)}{q}\right))$$

Consideration of the protocol security under the false authentication threat is significant part of the security proof. In corroboration we consider **the PACE protocol** that security proof doesn't take into account threat of the false authentication.

This protocol is vulnerable to a reflection attack. Indeed, let A and B are the protocol users. Denote by A' the active adversary who wants to authenticate on the user B (e.g. on the server) instead of the honest user A. According to the protocol description user B is the first who sends informative messages to partner (to be exact values Y_B and T_B) (see Table 3). Let A' intercepts messages of the honest user A and sends modified messages to user B. Adversary modifies messages by the following way: on the key exchange step A' sets point Y_A equal to point Y_B, on the authentication step A' sets value T_A equal to value T_B. Obviously such adversary A' successfully authenticates instead of the honest user A.

The protocol proposed in this paper doesn't have such a vulnerability. Two different techniques are used to achive it. On the one hand generation of data for the authentication step require different prefixes for different protocol parties (see Table 9, rows labeled with 4). On the other hand generation of masked Diffie-Hellman Keys (u_1 and u_2) assumes different

operations for different sides ($-Q_{PW}$ for A, $+Q_{PW}$ for B). So if the adversary B' sends $u_2 = u_1$ to honest user A, then user A will obtain incorrect value $Q_A = u_2 - Q_{PW}^A = \alpha P - 2Q_{PW}^A$ instead of $u_2 - Q_{PW}^A = \beta P$.

Public Information: EC parameters $\mathcal{G} = \{a, b, p, q, G\}$		
A (Client): π		B (Server): π
$K_\pi = \mathcal{H}(\pi \parallel 0)$		$K_\pi = \mathcal{H}(\pi \parallel 0)$
$s \in_R \mathbb{Z}_q$		
$z = E_{K_\pi}(s)$	$\mathcal{G}, z \xrightarrow{}$	abort if $\mathcal{G}$ incorrect
		$s = E_{K_\pi}^{-1}(z)$
	Map2Point(s)	
	$\longleftrightarrow$	
	Jointly generate $\widehat{G}$	
$y_A \in_R \mathbb{Z}_q^*,\ Y_A = y_A \cdot \widehat{G}$	$Y_A \xrightarrow{}$	$y_B \in_R \mathbb{Z}_q^*,\ Y_B = y_B \cdot \widehat{G}$
	$\xleftarrow{} Y_B$	
abort if $Y_B \notin \langle G \rangle \backslash \{0\}$		abort if $Y_A \notin \langle G \rangle \backslash \{0\}$
$K = y_A \cdot Y_B$		$K = y_B \cdot Y_A$
$K_{enc} = \mathcal{H}(K \| 1)$		$K_{enc} = \mathcal{H}(K \| 1)$
$K_{mac} = \mathcal{H}(K \| 2)$		$K_{mac} = \mathcal{H}(K \| 2)$
$T_A = MAC_{K_{mac}}(Y_B, \widehat{G}, \mathcal{G})$	$T_A \xrightarrow{}$	$T_B = MAC_{K_{mac}}(Y_A, \widehat{G}, \mathcal{G})$
	$\xleftarrow{} T_B$	
abort if T_B invalid		abort if T_A invalid

Table 3: The PACE protocol

Let's consider other protocols for their efficiency and security properties.

The **SPAKE2 protocol** (see Table 4) includes 1 round and 2 flows with complexity 4 exponentiation computations in cyclic group G. According to the Abdalla this protocol is one of the most efficient PAKE protocols. This protocol doesn't include a key authentication step, thus the SPAKE2 protocol is secure under a threat of distinguishing a generated session key from a random string only. As the first part of the protocol described in current paper SPAKE2 is a variation of the password-based encrypted key exchange protocol of Bellovin and Merritt ([19]).

Authors discussed a necessity of using hash functions for protecting against «related keys attack». The same technique is used for protocol SESPAKE proposed in this paper (see Table 9, rows labeled with 3).

The Dragonfly protocol (see Table 5) consists of two step: a key generation step and a key authentication step. This protocols includes 1 round and 4 flows with complexity 6 exponentiation computations in cyclic group G.

In 2013 it was shown by Dylan Clarke, Feng Hao that this protocol is vulnerable to a small subgroup attack. There is no checking that the received element E (more specifically, E_A for Bob and E_B for Alice) is a member of the group being used by the cryptographic scheme. For more details see [23]. The protocol proposed in current paper doesn't have such

Public Information: G, g, p, q, M, N		
A (Client): $pw \in \mathbb{Z}_q$		B (Server): $pw \in \mathbb{Z}_q$
$x \in_R \mathbb{Z}_q,\ X = g^x$		$y \in_R \mathbb{Z}_q,\ Y = g^y$
$X^* = X \cdot M^{pw}$		$Y^* = Y \cdot N^{pw}$
	$\xrightarrow{X^*}$	
	$\xleftarrow{Y^*}$	
$K_A = (Y^*/N^{pw})^x$		$K_B = (X^*/M^{pw})^y$
$SK_A = \mathcal{H}(A, B, X^*, Y^*, pw, K_A)$		$SK_B = \mathcal{H}(A, B, X^*, Y^*, pw, K_B)$

Table 4: The SPAKE2 protocol for finite fields

vulnerability as there are verifications of type $\frac{m}{q}Q_B \stackrel{?}{=} 0_E$ (see Table 9, rows labeled with 2) (more precisely in Section 3.9.2).

Public Information: G, p, q		
A (Client): $P \in G$		B (Server): $P \in G$
$r_A, m_A \in_R \{1, \ldots, q\}$		$r_B, m_B \in_R \{1, \ldots, q\}$
$s_A = r_A + m_A$		$s_B = r_B + m_B$
$E_A = P^{-m_A}$		$E_B = P^{-m_B}$
	$\xrightarrow{E_A, s_A}$	
	$\xleftarrow{E_B, s_B}$	
$K = (P^{s_B} E_B)^{r_A} = P^{r_A r_B}$		
$T_A = \mathcal{H}(K\|E_A\|s_A\|E_B\|s_B)$	$\xrightarrow{T_A}$	Verify T_A
		$K = (P^{s_A} E_A)^{r_B} = P^{r_A r_B}$
Verify T_B	$\xleftarrow{T_B}$	$T_B = \mathcal{H}(K\|E_B\|s_B\|E_A\|s_A)$
$SK = \mathcal{H}(K\|E_A \cdot E_B\|(s_A + s_B) \bmod q)$		

Table 5: The DragonFly protocol for finite fields

The J-PAKE protocol (see Table 6) includes the technique of zero-knowledge proof (PK, VK), thus this protocol is not efficient. During the protocol the server and the client need to compute 12 modular exponentiations in cyclic group and need to exchange 12 group elements and 6 scalars whereas the protocol SESPAKE needs only 4 computations of multiple points in the group of elliptic curve points and 2 group elements and 2 hash values exchanges.

The AugPAKE protocol (see Table 7) was proposed as one of the most efficient protocols. According to authors, the user needs to compute only 2 modular exponentiation computations while the server needs to compute 2.17 modular exponentiation computations. AugPAKE needs to exchange 2 group elements and 2 hash values. This protocol is augmented (non-balanced). It is claimed that even for corrupted server there is no possibility to impersonate client without offline dictionary attack.

The fully description of this protocol is got in Internet-Draft (see [32]). But there is no exhaustive scheme of counters exploitation. The counter management is significant part of

Public Information: $G, g, p, q, \sigma \xleftarrow{R} Setup(1^\kappa), s \xleftarrow{?} \{0,1\}^t$		
A (Client): $pw \in \mathbb{Z}_q$ $x_1, x_2 \in_R \mathbb{Z}_q$ $X_1 \equiv g^{x_1},\ X_2 \equiv g^{x_2}$ $\pi_1 \xleftarrow{R} \mathrm{PK}((X_1, g), x_1, A)$ $\pi_2 \xleftarrow{R} \mathrm{PK}((X_2, g), x_2, A)$	$\xrightarrow{(A, X_1, X_2, \pi_1, \pi_2)}$ $\xleftarrow{(B, X_3, X_4, \pi_3, \pi_4)}$	B (Server): $pw \in \mathbb{Z}_q$ $x_3, x_4 \in_R \mathbb{Z}_q$ $X_3 \equiv g^{x_3},\ X_4 \equiv g^{x_4}$ $\pi_3 \xleftarrow{R} \mathrm{PK}((X_3, g), x_3, B)$ $\pi_4 \xleftarrow{R} \mathrm{PK}((X_4, g), x_4, B)$
Abort if $X_4 = 1$ Abort if $\mathrm{VK}((X_3, g), \pi_3, B)$ fails Abort if $\mathrm{VK}((X_4, g), \pi_4, B)$ fails $\alpha \equiv (X_1 X_3 X_4)^{x_2 pw}$ $\pi_\alpha \xleftarrow{R} \mathrm{PK}((\alpha, X_1 X_3 X_4), x_2 pw, A)$	$\xrightarrow{(\alpha, \pi_\alpha)}$ $\xleftarrow{(\beta, \pi_\beta)}$	Abort if $X_2 = 1$ Abort if $\mathrm{VK}((X_1, g), \pi_1, A)$ fails Abort if $\mathrm{VK}((X_2, g), \pi_2, A)$ fails $\beta \equiv (X_1 X_2 X_3)^{x_4 pw}$ $\pi_\beta \xleftarrow{R} \mathrm{PK}((\alpha, X_1 X_2 X_3), x_4 pw, B)$
Abort if $\mathrm{VK}((\beta, X_1 X_2 X_3), \pi_\beta, B)$ fails $K = (\beta X_4^{-x_2 pw})^{x_2}$		Abort if $\mathrm{VK}((\alpha, X_1 X_3 X_4), \pi_\alpha, A)$ fails $K = (\alpha X_4^{-x_4 pw})^{x_4}$
$SK = \mathcal{H}(s \| K)$		

Table 6: The J-PAKE protocol

protocol architecture as it prevents some trivial attack (for more detail see Section 3.9.1).

The protocol SESPAKE proposed in the current paper is balanced.

Public Information: G, g, p, q		
A (Client): $pw \in \mathbb{Z}_q$ $x \in_R \mathbb{Z}_q^*,\ X \equiv g^x$	$\xrightarrow{A, X}$ $\xleftarrow{B, Y}$	B (Server): $W \equiv g^{pw}$ $y \in_R \mathbb{Z}_q^*,\ K \equiv g^y$ $r = \mathcal{H}(A, B, X), Y \equiv (X \cdot W^r)^y$
$r = \mathcal{H}(A, B, X),\ K' \equiv Y^{1/(x + pw \cdot r)}$ $T_A = \mathcal{H}_1(A \| B \| X \| Y \| K')$ if $T_B \neq \mathcal{H}_2(A \| B \| X \| Y \| K')$ reject $SK = \mathcal{H}_3(A \| B \| X \| Y \| K')$	$\xrightarrow{T_A}$ $\xleftarrow{T_B}$	if $T_A \neq \mathcal{H}_1(A \| B \| X \| Y \| K)$ reject $T_B = \mathcal{H}_2(A \| B \| X \| Y \| K)$ $SK = \mathcal{H}_3(A \| B \| X \| Y \| K)$

Table 7: The AugPAKE protocol

The SPEKE protocol (see Table 8) includes 1 round and 2 flows with complexity 4 exponentiation computations in cyclic group G and 2 verifications of the membership of a group element to the cyclic group G. This protocol is vulnerable to an impersonation attack. This attack is realised in concurrent model, when several copies of the protocol can

be processed concurrently. For more details see [29].

Despite the fact that security of the protocol SESPAKE was analyzed in the non-concurrent model this protocol doesn't have such a vulnerability since unique identifiers are used as input for computing authentication data (see Table 9, rows labeled with 6).

Public Information: G, p, q		
A (Client): pw		B (Server): pw
$g = F(pw) \in G$		$g = F(pw) \in G$
$x \in_R \{1, \ldots, q\},\ X \equiv g^x$		$y \in_R \{1, \ldots, q\},\ Y \equiv g^y$
	$\xrightarrow{X}$	
	$\xleftarrow{Y}$	
$K = \mathcal{H}(x \cdot Y)$		$K = \mathcal{H}(y \cdot X)$
$T_A = \mathcal{H}(\mathcal{H}(K))$	$\xrightarrow{T_A}$	Verify T_A
Verify T_B	$\xleftarrow{T_B}$	$T_B = \mathcal{H}(K)$

Table 8: The SPEKE protocol

Flags z_A and z_B are used for protocol implementation in constant time. This technique is used to prevent «side channel attack» (see Table 9, rows labeled by 1).

2.3 Results

The protocol proposed in this paper is the first type protocol, i.e. the generated session key is used to compute authentication data. In order to analyze the security considering all the protocol steps we propose to extend the adversary model. The consideration of the additional oracles $\mathcal{O}_{check}$ and $\mathcal{O}_{auth}$ allows to define a new threat *of distinguishing the real authentication data from the random one*. The formal definition of the adversary model is described in Section 3.

3 The proposed password authenticated key exchange protocol

3.1 Basic concepts and notations

We denote by V_n the set of all strings of length n with elements from the field $GF(2)$.

All operations with points of an elliptic curve are computed in a subgroup E of the prime order q in a group of some elliptic curve points. P is a generator of the subgroup E, m is the order of the group of used elliptic curve points. A run time for computing multiple points in the group of elliptic curve points is denoted by τ, 0_E is the identity element of the group E. E^* is a set $E \setminus \{0_E\}$.

F is a function PBKDF2, defined in [4], H_{256} is a hash-algorithm GOST R 34.11-2012 [5] with output's length 256 bits.

We assume that an *adversary* is a probabilistic Turing machine. We will call the *decisional problem* the task for the adversary that implies a binary answer. It assumes that there is a random bit b and the adversary does not know its value. A result of the adversary's attack is a bit value guess. Here $SUCC$ is an event, that occurs if the result of the adversary's attack and the value of the bit b are equal. For the adversary $\mathcal{A}$ who solves the decisional problem $Task$ it is usual to bound not the success probability, but an advantage $\mathbf{Adv}_{Task}(\mathcal{A})$ defined in the following way:

$$\mathbf{Adv}_{Task}(\mathcal{A}) = 2 \cdot \Pr[SUCC] - 1.$$

We denote by $\mathbf{Adv}_{Task'}(\mathcal{A})$ the success probability for the adversary who solves a computational problem $Task'$.

Depending on the problem the adversary can make requests to some oracles.

$\mathcal{A}(t, q_1, \ldots, q_k)$ is a set of the adversaries who work at most t steps and make at most q_i queries to the oracle $\mathcal{O}_i$, for each $i = 1, 2, \ldots, k$. We denote by $\mathbf{Adv}_{Task}(t, q_1, q_2, \ldots, q_k)$:

$$\mathbf{Adv}_{Task}(t, q_1, q_2, \ldots, q_k) = \max_{\mathcal{A} \in \mathcal{A}(t, q_1, q_2, \ldots, q_k)} \mathbf{Adv}_{Task}(\mathcal{A}).$$

If it is unnecessary to consider a set of the adversaries bounded by all parameters, unuseful parameters are not indicated. E.g. $\mathcal{A}(t)$ is a set of the adversaries who work at most t steps.

If $\mathcal{O}$ is some oracle that accepts the inputs from a set R, then $\mathcal{O}(r)$, $r \in R$ is the output of the oracle $\mathcal{O}$.

3.2 Brief description of the protocol

The protocol assumes an interaction of two participants. Each participant stores some value of a secret parameter. *A client* is a participant who stores a password as a secret parameter, and *a server* is a participant who stores a password-based computed point of the elliptic curve. *Client* and *Server* are a set of clients and a set of servers correspondingly.

A participant identifier is a byte-string of the constant length N (the length is the same for all participants of the protocol), ID ($ID = V_8^N$) is a set of all identifiers. If A is the participant of the protocol, then $A_{ID} \in ID$ is his identifier.

The client and the server (denote by A and B respectively) store $l+1$ provable pseudorandom points $Q_1, \ldots, Q_l$ and P. The provable pseudorandomness ensures that multiplicity of any point under any other point is unknown (more precisely in Section 3.9.3).

The client additionally stores his secret parameter — a password $PW \in V_8^k$.

The server additionally stores the following parameters (only Q_{PW} has to be kept secret):

- a number $ind \in \{1, \ldots, l\}$;
- $salt \in V_{64}$;
- a point $Q_{PW} = F(PW, salt, 2000) \cdot Q_{ind}$.

The parameter l is a random positive integer. Also every client A (server B) has his own open constant string T_A (T_B). There are detailed comments and explanations about the parameters and features of the protocol in Section 3.9.

5	Public Information: $l, P, Q_1, \ldots, Q_l, m, q$			
	A $[A_{ID}, PW]$		$B\,[B_{ID}, Q_{PW}, ind, salt]$	
		$\xrightarrow{A_{ID}}$		$SESPAKE_{KA}$
		$\xleftarrow{B_{ID}, ind, salt}$		
1	$z_A = 0$			
	$Q_{PW}^A = F(PW, salt, 2000) \cdot Q_{ind}$			
	$\alpha \in_R \{1, \ldots, q-1\}$			
4	$u_1 = \alpha \cdot P - Q_{PW}^A$	$\xrightarrow{u_1}$		
			if $u_1 \notin E \Rightarrow$ FINISH	
1			$z_B = 0$	
			$Q_B = u_1 + Q_{PW}$	
			$\beta \in_R \{1, \ldots, q-1\}$	
2			if $\frac{m}{q} Q_B = 0_E \Rightarrow Q_B = P$, $z_B = 1$	
			$src = (\frac{m}{q} \cdot \beta \mod q) Q_B$	
3			$K_B = H_{256}(src)$	
4		$\xleftarrow{u_2}$	$u_2 = \beta \cdot P + Q_{PW}$	
	if $u_2 \notin E \Rightarrow$ FINISH			
	$Q_A = u_2 - Q_{PW}^A$			
2	if $\frac{m}{q} Q_A = 0_E \Rightarrow Q_A = P$, $z_A = 1$			
	$src = (\frac{m}{q} \cdot \alpha \mod q) Q_A$			
3	$K_A = H_{256}(src)$			
6	$tag_A = T_A \vert\vert A_{ID} \vert\vert ind \vert\vert salt \vert\vert u_1 \vert\vert u_2$ $M_A = \text{HMAC}_{K_A}(tag_A)$	$\xrightarrow{M_A}$		$SESPAKE_{KC}$
			$tag = T_A \vert\vert A_{ID} \vert\vert ind \vert\vert salt \vert\vert u_1 \vert\vert u_2$ $M = \text{HMAC}_{K_B}(tag)$	
			If $M \neq M_A$ or $z_B \neq 0 \Rightarrow$ FINISH	
6		$\xleftarrow{M_B}$	$tag_B = T_B \vert\vert B_{ID} \vert\vert ind \vert\vert salt \vert\vert u_1 \vert\vert u_2$ $M_B = \text{HMAC}_{K_B}(tag_B)$	
	$tag = T_B \vert\vert B_{ID} \vert\vert ind \vert\vert salt \vert\vert u_1 \vert\vert u_2$ $M = \text{HMAC}_{K_A}(tag)$			
	If $M \neq M_B$ or $z_A \neq 0 \Rightarrow$ FINISH			

Table 9: Description of the *SESPAKE* protocol

A scheme of the protocol is shown at Table 9.

For convenience we denote by $SESPAKE_{KA}$ a key agreement protocol up to the computation of the function HMAC, $SESPAKE_{KC}$ is the second part of the protocol. This division is shown at Table 9. *SESPAKE* is a key agreement protocol $SESPAKE_{KA}$ with the following key validation step performed by the protocol $SESPAKE_{KC}$.

The first column of protocol scheme is used to explain some properties. The description of labels is:

1. Flag z_A (z_B) and assignment $Q_A = P$ ($Q_B = P$) are used for protocol implementation in constant time. This property is mentioned as required in Internet-Draft «Requirements for PAKE scheme» (see [1]). This technique is used to prevent «side channel attack». For more details about correct using of flags and session counters see Section 3.9.1.

2. The verifications of points for small order is a significant part of the PAKE protocols (for the SESPAKE protocol there are verifications of type $\frac{m}{q}Q_B \stackrel{?}{=} 0_E$). It helps to avoid such trivial attacks as a small subgroup attack. For example, the first version of the DragonFly protocol has such vulnerability (see Section 2.2, the DragonFly protocol). For more details about processing of the points of the small order see Section 3.9.2.

3. Using a hash-function is required to protect from «related keys attack» of the first type (see Section 2.2, the SPAKE2 protocol).

4. Different operations for different sides ($-Q_{PW}$ for A, $+Q_{PW}$ for B) are used to prevent reflection attack (see Section 2.2, the PACE protocol).

5. Several group generators with unknown multiplicity under other generators are required to protect from related keys attack of the second type (see Section 3.9.3).

6. The unique identifiers A_{ID} and B_{ID} are used as inputs for computing authentication data for protection against an impersonation attack in the concurrent model when several copies of the protocol can be processed concurrently (see Section 2.2, the SPEKE protocol).

3.3 Employed models

The proof of the protocol security is made in two steps in this paper. First, we will prove security of the protocol $SESPAKE_{KA}$ under a threat of distinguishing a session key from a random string. Then we will prove security of the protocol *SESPAKE* under the decisional version of the false authentication threat.

In this section we describe the models that are used in our further argumentations: the model of the network parties interaction and the adversary model for the protocols $SESPAKE_{KA}$ and *SESPAKE*.

3.3.1 Model of network parties interaction

We suggest each participant can take part in an arbitrary large number of the protocol sessions, but not at the same time — at every instant any participant can take part in one incomplete session only. We say a session is completed if all the protocol computations and all the transmissions are finished, or if one of the parties terminates the execution with an error caused by incorrectness of transmitted data or unsuccessful validation.

The actions of the participant A are modelled with the use of the $\mathcal{O}_A^i$ oracle in each session, where i is a session number. We assume that a creation of the new session is permitted only if all oracles $\mathcal{O}_A^i$ of the participant A have either generated session keys (for $SESPAKE_{KA}$) or have terminated with error. After the creation of new i-th protocol session with $A \in Client$ and $B \in Server$ the oracle $\mathcal{O}_A^i$ corresponding to the client A waits for the start request in the form of a special string $start$. The oracle $\mathcal{O}_A^i$ outputs the identifier A_{ID} in response to this query. The oracle $\mathcal{O}_B^i$ waits for a query in a form of some client identifier and outputs a pair $(ind, salt)$ in response. Further formats of the queries to the oracles and their responses to these queries are defined within the protocol. It should be additionally noted that after processing of each query the oracle moves to a new state defined by the result of the internal computations (for example, with a result of pseudorandom number generator execution) and the protocol specification. The correctness of a query to the oracle is defined with its state and the protocol (for example, if after the first query (a string $start$) to the oracle $\mathcal{O}_A^i$ ($A \in Client$) not a pair $(ind, salt)$ but a string $start$ comes again as an input, the oracle thinks that the query is incorrect). If a query is incorrect the oracle returns nothing and moves in a state $FAIL$. In this state oracle doesn't respond to any query and does not change its state.

During the interaction each oracle forms a string IDS («ID of Session») by adding all information that was received or was sent according to the protocol. E.g. total values of $IDS_{A,i}$ and $IDS_{B,j}$ of oracles $\mathcal{O}_A^i$ and $\mathcal{O}_B^j$ are equal $A_{ID}||ind||salt||u_1||u_2$ after a correct termination of the protocol $SESPAKE_{KA}$ session.

It should be additionally noted, that these oracles model a participant reaction to the messages received during the interaction according to the protocol, but the adversary completely controls communications, so the oracles do not exchange any information. It allows to model active «man in the middle».

For any oracle $\mathcal{O}_A^i$ the following conditions may be met:

- $ACCEPTED$ — $\mathcal{O}_A^i$ has generated a common key according to the protocol procedure;
- $OPENED$ — the adversary has found out a session key generated by the oracle $\mathcal{O}_A^i$ with help of the oracle $\mathcal{O}_{reveal}$ (this oracle is described below)
- $UNOPENED$ — for the oracle $\mathcal{O}_A^i$ the condition $OPENED$ is not met.

We assume that the oracle uses random analogues of the described cryptographic functions (a hash function and authentication codes) i.e., for example, for a function HMAC with a key K we use a mapping $G(K, \cdot) : V^* \to V_{256}$, where G is a family of the random functions. The family G is chosen in the beginning of the execution by the following way: a mapping $G(K, \cdot)$, $K \in V_{256}$, is chosen independently and uniformly from the set of all mappings from V^* to V_{256}.

3.3.2 Adversary model for the protocol $SESPAKE_{KA}$

We can informally describe the adversary model by the following way:

Attack. The adversary can initiate any number of sessions between any couple of the participants. He has a complete access to a transmission channel (an active «man in the middle»). The adversary can find out the session keys of any initiated connection except for one (we will call this connection special; the adversary chooses it during his execution).

Threat. The adversary wants to reveal any information about the session key of the special connection.

It should be noted that the adversary's ability to reveal the session keys reflects the fact that these session keys can be used by a participant in a wrong way after its generation (e.g. as keys of the weak cipher).

It is necessary to define one more property of the oracle $\mathcal{O}_A^i$ before a formal description of the adversary model .

Let us say that the oracle $\mathcal{O}_A^i$ is *fresh* if:

1) for $\mathcal{O}_A^i$ conditions $ACCEPTED$ and $UNOPENED$ are satisfied;

2) for any oracle $\mathcal{O}_B^j$, whose IDS value is equal to IDS value of the oracle $\mathcal{O}_A^i$, the condition $UNOPENED$ is satisfied.

We'll now describe abilities of the adversary using the set of the oracles he can make requests to:

1. $\mathcal{O}_{exec}$ — in response to request (A_{ID}, B_{ID}) a new session for the participants A and B is created (here the oracles $\mathcal{O}_A^i$ and $\mathcal{O}_B^j$ are reserved for unused numbers i and j). The oracle $\mathcal{O}_{exec}$ outputs as a result a complete log of the messages that were sent to each other by A and B during the protocol execution (a consequence of answers of the oracles $\mathcal{O}_A^i$ and $\mathcal{O}_B^j$);

2. $\mathcal{O}_{send}$ — in response to the request (A, i, m) the oracle returns message $\mathcal{O}_{send}(A, i, m)$, that the oracle $\mathcal{O}_A^i$ returns at the moment, receiving message m (i.e. $\mathcal{O}_{send}(A, i, m) = \mathcal{O}_A^i(m)$ for the current state $\mathcal{O}_A^i$);

3. $\mathcal{O}_{reveal}$ — if for the oracle $\mathcal{O}_A^i$ the condition $ACCEPTED$ is satisfied, then in response to request (A, i) the oracle $\mathcal{O}_{reveal}$ returns message $\mathcal{O}_{reveal}(A, i)$, equal to $K_{A,i}$ — a session key of $\mathcal{O}_A^i$;

4. $\mathcal{O}_{test}$ — accepts requests of the form (A, i). If the oracle $\mathcal{O}_A^i$ is fresh, the oracle $\mathcal{O}_{test}$ chooses a random bit b ($b \in_R \{0, 1\}$) and, if $b = 0$, outputs a random string as long as a session key, and if $b = 1$, outputs a session key $K_{A,i}$;

5. $\mathcal{O}_H$ is the oracle, computing the same function H used by the participant oracles.

A concept of the fresh oracle is useful for prevention from the considered threat degeneration because of the oracle $\mathcal{O}_{reveal}$ existence. If we don't limit the access abilities to the oracle $\mathcal{O}_{test}$, the adversary can carry out the considered threat trivially: after a legal generation of the common key by the oracles $\mathcal{O}_A$ and $\mathcal{O}_B$ the adversary, using $\mathcal{O}_{reveal}$, gets a key K_A

of the oracle $\mathcal{O}_A$ and then makes a request to the oracle $\mathcal{O}_{test}$ to recognise a key of the oracle $\mathcal{O}_B$, comparing a string received from $\mathcal{O}_{test}$ with K_A. The adversary, following this strategy, can fail with negligible probability.

Procedure 3.1. *Let us consider a procedure, where the adversary has access to the oracles $\mathcal{O}_{exec}$, $\mathcal{O}_{send}$, $\mathcal{O}_{reveal}$, $\mathcal{O}_H$ and $\mathcal{O}_{test}$, moreover he can make one request to the oracle $\mathcal{O}_{test}$ and for a fresh oracle $\mathcal{O}_A^i$ only. The adversary outputs either 0 or 1 as a result. Let SUCC is an event that occurs if the adversary result is equal to a bit b of the oracle $\mathcal{O}_{test}$.*

Denote by $\mathbf{Adv}_{SESPAKE_{KA}}(\mathcal{A})$ an advantage of the adversary $\mathcal{A}$ in the experiment described above.

3.4 Adversary model for the protocol *SESPAKE*

We consider the adversary model under some variant of false authentication threat. More precisely we consider a threat of distinction a string M, that was computed using a real session key of participant, from a string M', that was computed using a random key.

Let us describe the abilities of the adversary $\mathcal{A}$ for the protocol *SESPAKE*. Besides the abilities of the adversary for the protocol $SESPAKE_{KA}$, the adversary $\mathcal{A}$ can make a request to the oracles $\mathcal{O}_{\text{HMAC}}$, $\mathcal{O}_{check}$ and $\mathcal{O}_{auth}$. These oracles work according to the following rules:

- $\mathcal{O}_{\text{HMAC}}$: realizes a family of the random mappings $\{G_K : V^* \to V_{256}\}_{K \in V_{256}}$, that are chosen independently and uniformly before the attack; in response to request (K, T) the oracle $\mathcal{O}_{\text{HMAC}}$ returns a value $G_K(T)$;

- $\mathcal{O}_{check}$: in response to request (A, i) the oracle returns a value $\text{HMAC}_{K_{A,i}}(T_A||IDS_{A,i})$ (for HMAC we use the same family as for the oracle $\mathcal{O}_{\text{HMAC}}$). The employed key is a session key generated by the oracle $\mathcal{O}_A^i$, the argument is a string T_A concatenated with $IDS_{A,i}$.

- $\mathcal{O}_{auth}$: we define a threat using this oracle. Oracle $\mathcal{O}_{auth}$ receives as input a pair (A, i), chooses $b \in \{0, 1\}$ uniformly and depending on a value b returns either a pair (M_A, M') (if $b = 0$), or a pair (M_A, M) (if $b = 1$). At that $M' = \text{HMAC}_{K'}(T_B||IDS_{A,i})$, where $K' \in_R V_{256}$, $M_A = \text{HMAC}_{K_{A,i}}(T_A||IDS_{A,i})$, and $M = \text{HMAC}_{K_{A,i}}(T_B||IDS_{A,i})$.

As in the model for the $SESPAKE_{KA}$ protocol, the adversary can make a request to the oracle $\mathcal{O}_{auth}$ for fresh oracles only. A fresh oracle is such an oracle $\mathcal{O}_A^i$ that there were not any requests to the oracle $\mathcal{O}_{check}$ for it and the adversary did not get a value M by the trivial way, i. e. he did not make a request to the oracle $\mathcal{O}_{check}$ for the oracle sharing with $\mathcal{O}_A^i$ a common key. Informally the adversary warns which participant and session he will realize a threat to. Emphasize that the adversary chooses his target not in the beginning of attack but as the work advances.

The adversary returns a value $a \in \{0, 1\}$. Denote by $SUCC_{auth}$ an event that occurs if $a = b$, where b is a value that was used by the oracle $\mathcal{O}_{auth}$

3.4.1 Special case: some standard adversary model

This section is called upon to illustrate the fact that the model described above contains all the most common practical adversary models. The *SESPAKE* protocol is intended to be applied mainly for communications between a key carrier (card) and a client (computer). In this case the client stores a password, the card stores a point Q_{PW}. In this section we will use the terminology *client-card*.

Example 3.2. *The adversary is a client. Attack: the adversary imitates a client and can send any message to a card and receive answers. Threat: the adversary successfully authorizes on the card.*

This scenario gets covered with help of the oracles described in Section 3.3.2. In this case the adversary makes a request to the oracle $\mathcal{O}_{send}$ *to send messages to the card on behalf of the client. The successful authentication of the adversary on the card means that he has generated a common key with this card. So the adversary solves the problem at least as complicated as the distinguishing a session key from a random string problem.*

Example 3.3. *The adversary is a man in the middle. He can intercept any message that participants send to each other according to the protocol. He has an ability to change, keep, send arbitrary messages to the arbitrary participant on behalf of the arbitrary participant. A threat consists in the fact that the adversary knows some information about the key.*

The abilities of the adversary to make different manipulations with messages get covered by his ability to make requests to the oracle $\mathcal{O}_{send}$ *described in Section 3.3.2. Thus «man in the middle» is not stronger than the adversary, who has an access to the oracle* $\mathcal{O}_{send}$ *in the sense of its own abilities. To carry out a threat to get some information about the session key is not easier than to distinguish the session key from a random string.*

To complete a picture we should mention the following example.

Example 3.4. *The adversary is a card. Suppose the adversary is a fake card that wants to know the client password (even at least point* Q_{PW}*).*

In this case the adversary can interact with the client as a usual card. This behavior is completely described by the ability to use the oracle $\mathcal{O}_{send}$*. The client password obtaining leads to the situation that the keys of all further sessions will be known by the adversary-card. Thus the adversary-card certainly solves his problem to distinguish the session key from the random string.*

It should be noted that necessity of the oracle $\mathcal{O}_{reveal}$ consideration is conditioned by a wrong utilization of the session key (e.g. as a key of a weak cryptographic algorithm).

3.5 Accompanying problems and relations between them

In this section we consider some problems that will be used to prove the protocol security.

All these problems are formulated for subgroup E of a elliptic curve points group, moreover $|E| = q$, where q is a prime number.

3.5.1 CDH

We formulate the CDH problem (the computational Diffie-Hellman problem) using an experiment $\mathbf{Exp}^{\mathrm{CDH}}$:

- Choose two numbers $x, y \in \mathbb{Z}_q$ and a point $P \in E$ independently at random, $P \neq 0_E$. Compute $X = xP$ and $Y = yP$;
- Send points P, $X = xP$, $Y = yP$ to the adversary $\mathcal{A}$;
- The adversary $\mathcal{A}$ returns a point Z;
- If $Z = xyP$, then a result $\mathbf{Exp}^{\mathrm{CDH}}(\mathcal{A})$ of this experiment is 1, else is 0.

The success probability for the adversary $\mathcal{A}$ is:

$$\mathbf{Adv}_{\mathrm{CDH}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\mathrm{CDH}}(\mathcal{A}) = 1\right].$$

A characteristic that reflects the complexity of solving the problem CDH is

$$\mathbf{Adv}_{\mathrm{CDH}}(t) = \max_{\mathcal{A} \in \mathcal{A}(t)} \mathbf{Adv}_{\mathrm{CDH}}(\mathcal{A}).$$

For $X = xP$ and $Y = yP$ we denote by $\mathrm{CDH}_P(X, Y)$ a point xyP. We will simply write $\mathrm{CDH}(X, Y)$ if it is clear from the context what P is used.

3.5.2 SCDH

The SCDH problem ([3]) differs from the problem CDH in fact that we send to the adversary points X and P only, and his task is to compute $\mathrm{CDH}_P(X, X)$. We denote by $\mathrm{SCDH}_P(X)$, where $X = xP$, a point x^2P.

3.5.3 QCCDH

Now we will describe the following experiment, which will be further denoted as $\mathbf{Exp}^{\mathrm{QCCDH}}$.

- Choose points $X, Q, P \in E$, $P \neq 0_E$ independently at random and send them to the adversary;
- The adversary $\mathcal{A}$ returns points Y, U and V;
- If $U = \mathrm{CDH}_P(X, Y)$ and $V = \mathrm{CDH}_P(X - Q, Y - Q)$ then return 1 else return 0.

The success probability for the adversary $\mathcal{A}$ is:

$$\mathbf{Adv}_{\mathrm{QCCDH}}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\mathrm{QCCDH}}(\mathcal{A}) = 1\right].$$

3.5.4 $\mathrm{PCCDH}_{\mathcal{D}}$

Denote by $\mathcal{D} \subset \mathbb{Z}_q$ a set of passwords. P is a generator of subgroup E.

The experiment $\mathbf{Exp}^{\mathrm{PCCDH}}(\mathcal{A})$ for the $\mathrm{PCCDH}_{\mathcal{D}}$ problem consists of two steps, and the adversary $\mathcal{A}$ in this experiment uses two random tapes for the different steps and returns a content of the first one along with the value of the point Y. The content of the random tape for the first step can be set before the beginning of the experiment arbitrarily. If it is not set, then the tape is filled uniformly at random.

The experiment $\mathbf{Exp}^{\mathrm{PCCDH}}(\mathcal{A})$ is performed as follows:

- Choose points $Q_1, Q_2, X \in_R E$;
- Send Q_1, Q_2, X to the adversary $\mathcal{A}$;
- The adversary returns $Y \in E$ and u, a content of his random tape used for computations on the first step;
- Choose a password $r \in_R \mathcal{D}$ and send it to the adversary $\mathcal{A}$;
- The adversary returns $K \in E$;
- If $K = \mathrm{CDH}_P(X - r \cdot Q_1, Y - r \cdot Q_2)$, then return 1 as a result, else return 0.

Remark 3.5. *The adversary, who has guessed the right password r before the returning Y, solves this problem guaranteed, returning $Y = P + r \cdot Q_2$ and $K = X - r \cdot Q_1$.*

Remark 3.6. *Returning the content of the random tape gives an ability to «start up» the adversary $\mathcal{A}$ in such a way that the result of his work for the first step (a point Y) is not changed for the same input data. It means that after receiving the point Y we can «start up» the adversary many times for the second step only.*

3.5.5 $\mathrm{QPCCDH}_{\mathcal{D}}$

The $\mathrm{QPCCDH}_{\mathcal{D}}$ problem differs from the $\mathrm{PCCDH}_{\mathcal{D}}$ problem in fact that points Q_1 and Q_2 are the same.

3.5.6 $\mathrm{S\text{-}PCCDH}_{\mathcal{D},s}$

The $\mathrm{S\text{-}PCCDH}_{\mathcal{D},s}$ problem with parameters $s \in \mathbb{N}$ differs from the problem $\mathrm{PCCDH}_{\mathcal{D}}$ in fact that instead of one key K the adversary $\mathcal{A}$ can return a set of key $\mathcal{K}$ of cardinality s as a result. We return 1 as a result of the experiment $\mathbf{Exp}^{\mathrm{S\text{-}PCCDH}}(\mathcal{A})$, if a point $\mathrm{CDH}_P(X - r \cdot Q_1, Y - r \cdot Q_2)$ is in a set $\mathcal{K}$.

3.5.7 $\mathrm{S\text{-}QPCCDH}_{\mathcal{D},s}$

The $\mathrm{S\text{-}QPCCDH}_{\mathcal{D},s}$ problem differs from the $\mathrm{S\text{-}PCCDH}_{\mathcal{D},s}$ problem in fact that in the experiment $\mathbf{Exp}^{\mathrm{S\text{-}QPCCDH}}(\mathcal{A})$ the points Q_1 and Q_2 are equal.

Further, if it is clear from the context for what parameters s and $\mathcal{D}$ we consider the problems described above, then their names will be used without indexes.

3.5.8 Known relations

In this section some statements are proved which estimate the bounds of the success probability of the S-QPCCDH problem solution through the success probability of the CDH problem solution.

We have produced a full detailed proof of the first theorem in order to explain the used methods as well as the used technique.

Theorem 3.7. *The following inequalities hold:*

$$\mathbf{Adv}^2_{\mathrm{SCDH}}(t) \leqslant \mathbf{Adv}_{\mathrm{CDH}}(2t+\tau),$$

$$\mathbf{Adv}_{\mathrm{CDH}}(t) \leqslant \mathbf{Adv}_{\mathrm{SCDH}}(t+\tau).$$

Proof. Suppose $\mathcal{A}$ is an adversary, who solves the SCDH problem and works at most t steps. We describe an adversary $\mathcal{A}'$ using $\mathcal{A}$ as a black box to solve the CDH problem. Notice, that

$$\mathrm{CDH}(X,Y) = \frac{1}{4}\left(\mathrm{SCDH}(X+Y) - \mathrm{SCDH}(X-Y)\right).$$

For the points X and Y (we want to find the value $\mathrm{CDH}(X,Y)$ for them) $\mathcal{A}'$ firstly sends $X+Y$ as an input to $\mathcal{A}$ and then $X-Y$. $\mathcal{A}'$ computes $\mathrm{c} = 1/4 \cdot (a-b)$ after receiving answers a and b from $\mathcal{A}$ correspondingly and returns c as a result.

Let bound the run time and the success probability for the adversary $\mathcal{A}'$. The run time of $\mathcal{A}'$ is bounded in the following way: the adversary $\mathcal{A}$ works at most $2t$ steps (two starts), we need 3 additive and 1 multiplicative operations for the computation c from a and b. We neglect the run time of additive operations and thus we can deduce that $\mathcal{A}' \in \mathcal{A}(2t+\tau)$.

The success probability $\mathbf{Adv}_{\mathrm{CDH}}(\mathcal{A}')$ for the adversary $\mathcal{A}'$ is bounded below by a value $(\mathbf{Adv}_{\mathrm{SCDH}}(\mathcal{A}))^2$, so if $\mathcal{A}$ solves his problem SCDH for both $X+Y$ and $X-Y$ successfully, then the adversary $\mathcal{A}'$ finds $\mathrm{CDH}(X,Y)$.

As the adversary $\mathcal{A}$ is arbitrarily selected we can choose such $\mathcal{A}$ that $\mathbf{Adv}_{\mathrm{SCDH}}(\mathcal{A}) = \mathbf{Adv}_{\mathrm{SCDH}}(t)$. The first statement of theorem is followed from the chain of relations:

$$\mathbf{Adv}_{\mathrm{CDH}}(2t+\tau) \geqslant \mathbf{Adv}_{\mathrm{CDH}}(\mathcal{A}') \geqslant \mathbf{Adv}^2_{\mathrm{SCDH}}(\mathcal{A}) = \mathbf{Adv}^2_{\mathrm{SCDH}}(t).$$

Let consider the second inequality. Now suppose $\mathcal{A}$ is an adversary, who solves the CDH problem and works at most t steps. We describe an adversary $\mathcal{A}'$ using $\mathcal{A}$ as a black box to solve the SCDH problem for point X.

Firstly $\mathcal{A}'$ chooses $y \in_R \mathbb{Z}_q \backslash \{0\}$ and set $Y = X + yP$. Notice, that

$$\mathrm{CDH}(X,Y) = \mathrm{CDH}(X, X+yP) = \mathrm{CDH}(X,X) + y \cdot \mathrm{CDH}(X,P) = \mathrm{SCDH}(X) + yX.$$

Then $\mathcal{A}'$ sends X, Y as an input to $\mathcal{A}$, receives a point a and returns $b = a - yX$ as a result. Similarly to proof of the previous inequality we get

$$\mathbf{Adv}_{\mathrm{SCDH}}(t+\tau) \geqslant \mathbf{Adv}_{\mathrm{SCDH}}(\mathcal{A}') \geqslant \mathbf{Adv}_{\mathrm{CDH}}(\mathcal{A}) = \mathbf{Adv}_{\mathrm{CDH}}(t).$$

□

Theorem 3.8. *The following inequality holds:*

$$\mathbf{Adv}_{\mathrm{QCCDH}}(t) \leqslant \mathbf{Adv}_{\mathrm{SCDH}}(t+4\tau) \leqslant \sqrt{\mathbf{Adv}_{\mathrm{CDH}}(2t+9\tau)}.$$

Proof. Suppose $\mathcal{A} \in \mathcal{A}(t)$ is an adversary who solves the QCCDH problem with the success probability ε. For the points $Q \in E$, $P \in E \setminus \{0_E\}$ we choose a number $b \in \{0, 1, \ldots, q-1\}$ randomly and independently of points Q and P. Let Y, U, V is a triple of numbers that the adversary $\mathcal{A}$ outputs as a result, when he received (X, Q), where $X = b \cdot Q$. If he solves the QCCDH problem successfully, thus

$$U = \mathrm{CDH}(b \cdot Q, Y) = b \cdot \mathrm{CDH}(Q, Y),$$

$$V = \mathrm{CDH}((b-1) \cdot Q, Y - Q) = (b-1) \cdot (\mathrm{CDH}(Q, Y) - \mathrm{SCDH}(Q)).$$

A point

$$R = \left(\frac{1}{b} \mod q\right) \cdot U - \left(\frac{1}{b-1} \mod q\right) \cdot V$$

will be a solution of the SCDH problem for the pair Q, P with probability at least ε.

The second inequality is a corollary of the first one and the theorem 3.7. □

Theorem 3.9. *The following inequality holds:*

$$\mathbf{Adv}_{\mathrm{QPCCDH}_{\mathcal{D}}}(t) \leqslant \frac{1}{|\mathcal{D}|} + \sqrt{\mathbf{Adv}_{\mathrm{QCCDH}}(2t+2\tau)}.$$

Proof. Suppose $\mathcal{A} \in \mathcal{A}(t)$ is an adversary, who solves the $\mathrm{QPCCDH}_{\mathcal{D}}$ problem with the success probability δ. Put $n = |\mathcal{D}|$ and $\varepsilon = \mathbf{Adv}_{\mathrm{QCCDH}}(2t+2\tau)$.

Suppose $Q, X \in E$ are independently chosen random points, we need to solve the QCCDH problem for. We will use the adversary $\mathcal{A}$ to solve this problem.

Choose a random and uniform content u of a random tape for the first work step of $\mathcal{A}$ (notice that it is sufficient to consider a tape of the finite length). Choose random $r \neq r' \in \mathcal{D}$ and compute $\gamma = r' - r$.

Compute $Q' = \frac{1}{\gamma}Q$ and $X' = X + rQ'$. Then use $\mathcal{A}$ twice to solve the QPCCDH problem for the same input parameters for the first step and the different input parameters for the second step.

Give to the adversary $\mathcal{A}$ a pair Q', X' as an input, recording a value u on his random tape of the first step. Let Y' is a result that the adversary returns on the first step (the content of the random tape on the first step returned by $\mathcal{A}$ is equal to u). Give to $\mathcal{A}$ a number r' as an input for the second step. Let K' is a value that the adversary returns as a result.

Use $\mathcal{A}$ again, giving to him the same pair Q', X' and recording u on the his random tape of the first step. The adversary $\mathcal{A}$ returns the same value Y'. Give to $\mathcal{A}$ a number r as an input for the second step. Let K is a value that the adversary returns as a result on the second step.

If the adversary $\mathcal{A}$ solves both problems correctly, thus $K' = \mathrm{CDH}(X' - r'Q', Y' - r'Q')$, and $K = \mathrm{CDH}(X' - rQ', Y' - rQ')$. Let $Y = Y' - rQ'$, then a triple Y, K, K' will be the solution of the QCCDH problem for the input parameters X, Q. Indeed,

$$K = \mathrm{CDH}(\underbrace{X'}_{X+rQ'} - rQ', \underbrace{Y' - rQ'}_{Y}) = \mathrm{CDH}(X, Y),$$

$$K' = \mathrm{CDH}(X' - r'Q', Y' - r'Q') = \mathrm{CDH}(X + rQ' - r'Q', Y + rQ' - r'Q) = \\ = \mathrm{CDH}(X - (r' - r)Q', Y - (r' - r)Q') = \mathrm{CDH}(X - Q, Y - Q),$$

as $Q' = \frac{1}{r'-r}Q$.

Thus the probability to solve the QCCDH problem for the pair X, Q (we denote this event as $SUCC_{X,Q}$) for the adversary working this way is not smaller than the probability that $\mathcal{A}$ successfully solves both problems at the same time for the inputs X', Q' (the probability is determined by values r, r', u and the random content of tape for the second step) (event $SUCC_{X',Q'}$):

$$\Pr[SUCC_{X,Q}] \geqslant \Pr_{r,r',u,v}[SUCC_{X',Q'}] = \sum_u \Pr[u] \sum_{r \neq r'} \Pr[r, r'] \Pr[SUCC_{X',Q',r,r',u}],$$

where $SUCC_{X',Q',r,r',u}$ is an event that occurs if $\mathcal{A}$ solves both problems with parameters X', Q', r, r' and the content u recorded on the tape of the first step. Let bound the average success probability to solve the QCCDH problem by the way described above.

$$E_{X,Q}\Pr[SUCC_{X,Q}] \geqslant \sum_{X,Q} \Pr[X, Q] \sum_u \Pr[u] \sum_{r \neq r'} \Pr[r, r'] \Pr[SUCC_{X',Q',r,r',u}] \geqslant \\ \geqslant \sum_u \Pr[u] \sum_{r \neq r'} \Pr[r, r'] \sum_{X,Q} \Pr[X, Q] \Pr[SUCC_{X',Q',r,r',u}] \geqslant \\ \geqslant \sum_u \Pr[u] \sum_{r \neq r'} \Pr[r, r'] \sum_{A,B} \Pr[A, B] \Pr[SUCC_{A,B,r,r',u}].$$

The last inequality follows from the fact that mapping $(X, Q) \to (X', Q')$ is bijective, and the distribution of pairs $A, B \in E \times E$ is uniform. The probability $\Pr[SUCC_{A,B,r,r',u}]$ is a probability of the fact that the adversary $\mathcal{A}$ correctly solves the QPCCDH problem firstly for inputs A, B (the first step) and r (the second step) with u as a content of the random tape for the first step, and then the same problem but for parameter r' as an input for the second step. Here parameters A, B, r, r' and u are independent, thus $\Pr[SUCC_{A,B,r,r',u}] = \Pr[SUCC_{A,B,r,u}] \cdot \Pr[SUCC_{A,B,r',u}]$. Here $SUCC_{A,B,r,u}$ is an event that occurs if the adversary $\mathcal{A}$ solves the QPCCDH problem for input parameters A, B and r with u as a content of random tape for the first step. For shot, denote by π_r the probability $\Pr[SUCC_{A,B,r,u}]$, and by $\pi_{r'}$ the probability $\Pr[SUCC_{A,B,r',u}]$. Then

$$E_{X,Q}\Pr[SUCC_{X,Q}] \geqslant \sum_{A,B} \Pr[A, B] \sum_u \Pr[u] \sum_{r \neq r'} \Pr[r, r'] \pi_r \pi_{r'}. \tag{3.1}$$

Consider the sum $\sum_{r \neq r'} \Pr[r, r'] \pi_r \pi_{r'}$. A probability $\Pr[r, r']$ is equal to $\frac{1}{n(n-1)}$. Thus the next relation holds

$$\sum_{r \neq r'} \pi_r \pi_{r'} = n^2 \left(\sum_r \frac{1}{n} \pi_r \right)^2 - \sum_r \pi_r^2.$$

Use the result to bound the $E_{X,Q}\Pr[SUCC_{X,Q}]$ (we use the Jensen's inequality to bound

the first sum).

$$\frac{1}{n(n-1)}\left[n^2\underbrace{\sum_{A,B,u}\Pr[A,B,u]\left(\sum_r\frac{1}{n}\pi_r\right)^2}_{\geqslant\left(\sum_{A,B,u,r}\Pr[A,B,u,r]\pi_r\right)^2=\delta^2}-n\sum_{A,B,u}\Pr[A,B,u]\sum_r\frac{1}{n}\underbrace{\pi_r^2}_{\leqslant\pi_r}\right]\geqslant$$

$$\geqslant\frac{n}{n-1}\delta^2-\frac{1}{n}\delta.$$

We have $\varepsilon\geqslant E_{X,Q}\Pr[SUCC_{X,Q}]\geqslant\frac{n}{n-1}\delta^2-\frac{1}{n}\delta$ as a result. It is easy to show that the next relations hold as a corollary:

$$\delta\leqslant\frac{1+\sqrt{1+4\varepsilon n(n-1)}}{2n}\leqslant\frac{1}{2n}+\sqrt{\frac{1}{4n^2}+\varepsilon}\leqslant\frac{1}{n}+\sqrt{\varepsilon}. \tag{3.2}$$

We got the required inequality. □

Theorem 3.10. *The following relation holds:*

$$\mathbf{Adv}_{\text{S-QPCCDH}_{s,\mathcal{D}}}(t)\leqslant\frac{1}{|\mathcal{D}|}+\sqrt[4]{\mathbf{Adv}_{\text{SCDH}}(4t+\Theta+O(s\tau))+\frac{2s^4}{q}},$$

where Θ *is a run time of finding a pair of identical elements in two sets of cardinality* s^2.

Proof. Suppose $\mathcal{A}\in\mathcal{A}(t)$ is an adversary who solves the problem $\text{S-QPCCDH}_{s,\mathcal{D}}$ problem with the success probability δ. Put $n=|\mathcal{D}|$.

Suppose $Q\in E$ and $P\in E\setminus\{0_E\}$ are points that we need to solve the SCDH problem for. We will use the adversary $\mathcal{A}$ to solve this problem.

Choose numbers $x_1,x_2,q_1,q_2\in\mathbb{Z}_q$ independently at random. Compute the points $X_i=x_iQ$ and $Q_i=Q+q_iP$. Choose the independent contents u_1 and u_2 of the random tape for the adversary $\mathcal{A}$ at random. Then we use $\mathcal{A}$ as a black box in the same way as was described for proof of theorem 3.9 for the pair X_1,Q_1 and the content u_1 firstly and then for the pair Q_2,X_2 and the content u_2. These experiments are independent because of the choice parameters b_i, q_i and u_i. We get four sets with s points in each as a result of this procedure.

The probability of finding the solutions of the $\text{S-QPCCDH}_{s,\mathcal{D}}$ problem in both sets got in one procedure as a result is bounded below by the value $\frac{n}{n-1}\delta^2-\frac{1}{n-1}\delta$ (argumentations is the same as in the proof for the theorem 3.9). As we choose the parameters for the procedures independently, the probability that there will be a correct solution in each of four sets is at least $(\frac{n}{n-1}\delta^2-\frac{1}{n-1}\delta)^2$.

We denote by S_1, S_1', S_2 a S_2' the sets got in mentioned procedures as a result.

If the adversary $\mathcal{A}$ solves both problems in one of the procedures, then there are such multiple points K_i and K_i' in sets S_i and S_i' that the following equalities hold:

$$K_i=\text{CDH}(X_i,Y_i)=x_i\text{CDH}(Q,Y_i),$$

$$
\begin{aligned}
K'_i = \mathrm{CDH}(X_i - Q_i, Y_i - Q_i) = \\
= \mathrm{CDH}(X_i, Y_i) + \mathrm{CDH}(Q_i, Q_i) - \mathrm{CDH}(X_i, Q_i) - \mathrm{CDH}(Y_i, Q_i) = \\
= x_i\mathrm{CDH}(Q, Y_i) + \mathrm{CDH}(Q + q_iP, Q + q_iP) - \mathrm{CDH}(x_iQ, Q + q_iP) - \mathrm{CDH}(Q + q_iP, Y_i) = \\
= x_i\mathrm{CDH}(Q, Y_i) + \mathrm{SCDH}(Q) + 2q_iQ + q_i^2P - x_i\mathrm{SCDH}(Q) - x_iq_iQ - \mathrm{CDH}(Q, Y_i) - q_iY_i = \\
= (x_i - 1)\mathrm{CDH}(Q, Y_i) - (x_i - 1)\mathrm{SCDH}(Q) + \underbrace{2q_iQ + q_i^2P - x_iq_iQ - q_iY_i}_{C}.
\end{aligned}
$$

We deduce from here that

$$
\mathrm{SCDH}(Q) = \frac{1}{x_i - 1}\left(\frac{x_i - 1}{x_i}K_i - K'_i + C\right) = \frac{1}{x_i}K_i - \frac{1}{x_i - 1}K'_i + \frac{1}{x_i - 1}C. \tag{3.3}
$$

A point $\frac{1}{x_i - 1}C$ is computed once for each of two procedures. s points $\frac{1}{x_i}K_i$ or $\frac{1}{x_i - 1}K'_i$ (depending on the considered set) are computed for each of four sets. Then we form two sets with s^2 elements in each using the relation 3.3. The first element of the first set we found the same element in the second set is returned as supposed value $\mathrm{SCDH}(Q)$. The probability of the accidental coincidence is bounded by the value $\frac{2s^4}{q}$.

Thus the mentioned algorithm solves the SCDH problem with probability at least $\left(\frac{n}{n-1}\delta^2 - \frac{1}{n-1}\delta\right)^2 - \frac{s^4}{2q}$. Therefore,

$$
\mathbf{Adv}_{\mathrm{SCDH}}(4t + \Theta + O(s\tau)) \geqslant \left(\frac{n}{n-1}\delta^2 - \frac{1}{n-1}\delta\right)^2 - \frac{2s^4}{q}.
$$

From it we have

$$
\delta \leqslant \frac{1}{|\mathcal{D}|} + \sqrt[4]{\mathbf{Adv}_{\mathrm{SCDH}}(4t + \Theta + O(s\tau)) + \frac{2s^4}{q}}.
$$

□

3.6 A proof of security of the $SESPAKE_{KA}$ protocol

In this section we prove the lower bound of the advantage of some adversary in distinguishing a session key from a random string for the $SESPAKE_{KA}$ protocol.

Theorem 3.11. *For $q_{send} \geqslant 2$ and some t required for the interaction according to the protocol by one client the following inequality holds:*

$$
\mathbf{Adv}_{SESPAKE_{KA}}(t, q_{send}) \geqslant \frac{1}{|\mathcal{D}|} - \frac{1}{q}.
$$

Proof. We will instance an adversary who solves a task of distinguishing a session key from a random string for the $SESPAKE_{KA}$ protocol. The adversary sends some A_{ID} to the participant B using the oracle $\mathcal{O}_{send}$ and receives $(ind, salt)$ in response. The adversary chooses a password PW' from the dictionary $\mathcal{D}$ uniformly and computes $Q_{PW'}$. Also he chooses an element α from the set $\{1, \ldots, q-1\}$ uniformly. Then he computes u_1 according to the specification of the protocol (as if he were the participant A) and sends

u_1 to participant B using a query to the oracle $\mathcal{O}_{send}$. Receiving his key the adversary computes a session key k' according to the specification of the protocol using PW' as a password.

Then the adversary makes a request to the oracle $\mathcal{O}_{test}$ to distinguish this key, that has just been generated by the participant B. After receiving some string k the adversary compares this string with k' and outputs $b' = 1$ as a result if they are equal and $b' = 0$ otherwise.

If k is the key that was generated by the participant B during the interaction with the adversary, then $\Pr[k' = k] \geqslant \frac{1}{|\mathcal{D}|}$ (we neglect the probability of collision for hash-function during a key generation and for the function F during generation of the $Q_{PW'}$ value). Let b is a random bit of the oracle $\mathcal{O}_{test}$. The following relation holds:

$$\Pr[b' = b] = \Pr[b' = 0|b = 0]\Pr[b = 0] + \Pr[b' = 1|b = 1] \cdot \Pr[b = 1] =$$
$$= \frac{1}{2}\left((1 - \frac{1}{q}) + \Pr[b = 1|b' = 1]\right) \geqslant \frac{1}{2}\left((1 - \frac{1}{q}) + \frac{1}{|\mathcal{D}|}\right) = \frac{1}{2} + \frac{1}{2|\mathcal{D}|} - \frac{1}{2q}.$$

Thus $\mathbf{Adv}_{SESPAKE_{KA}}(t, q_{send}) \geqslant \frac{1}{|\mathcal{D}|} - \frac{1}{q}$. □

Taking into account that $|\mathcal{D}| \ll q$ the advantage of the adversary described above is a value of the order $\frac{1}{|\mathcal{D}|}$.

A proof of the $SESPAKE_{KA}$ protocol security (with some parameters) is made in the model with a random oracle $\mathcal{O}_H$ (we suggest that the hash-function H used in the protocol acts as a random function). Finally the security is based on the complexity of the computational Diffie-Hellman problem (CDH) in the corresponding group.

Remark 3.12. *Here we present some comments about the concept of the random oracle used to improve the understanding of some steps of the proofs. The term «random oracle» is widely used in papers about the mathematic cryptography. In our case the modelling of a function $H : A \to B$ using the random oracle means that computation of the function H is realized with some computer $\mathcal{O}_H$ whose input is an element from the set A and output is an element from the set B (oracle is an accepted definition for this computer). A «randomness» of the oracle $\mathcal{O}_H$ consists in the way he chooses a value from the set B that oracle outputs in response to a value-request from A. When processing the first request the oracle $\mathcal{O}_H$ chooses uniformly at random a function from the set of all functions which are defined on the set A and have values from the set B, returns a value of the chosen function for the input argument. For all next requests the oracle $\mathcal{O}_H$ returns the corresponding values of the chosen function.*

It is more convenient to interpret the definition of oracle behavior mentioned above by the following way. The oracle stores a table T of size $|A|$ indexed with the elements from A, all values are not defined before the first query. For input $a \in A$ the oracle $\mathcal{O}_H$ either returns value $T(a)$ if it has already been defined or chooses $b \in B$ uniformly at random, sets $T(a) = b$ and returns b in response. So in fact the oracle defines his function at the work advances.

Consider a task to distinguish two scenarios in the following experiment. Let $\mathcal{O}_1$ and $\mathcal{O}_2$ are the random oracles which realize a mapping $A \to B$. The adversary can make requests

of the form (i, a), $i \in \{1, 2\}$, $a \in A$, *to the oracle* $\mathcal{O}$ *that returns an element from* B *using the oracles* $\mathcal{O}_1$ *and* $\mathcal{O}_2$. *In the beginning of the work the oracle* $\mathcal{O}$ *chooses uniformly at random a value* $\beta \in \{1, 2\}$. *The oracle* $\mathcal{O}$ *stores a table* T *that is initially empty. In response to the request* (i, a) *the oracle* $\mathcal{O}$ *acts by the following way:*

- *if* $\beta = 1$, *then* $\mathcal{O}(i, a) = \mathcal{O}_1(a)$;
- *if* $\beta = 2$, *then, if* $T(a)$ *is not defined, then* $\mathcal{O}(i, a) = \mathcal{O}_i(a)$ *and* $T(a) = \mathcal{O}_i(a)$, *else* $\mathcal{O}(i, a) = T(a)$.

The adversary needs to solve a task to distinguish two scenarios: $\beta = 1$ *or* $\beta = 2$.

Such a detailed description was made for the strictness. Informally the adversary should define whether he receives answers from one random oracle or he receives the answers from two random oracles. Moreover he has no opportunity to make requests with identical second argument a *to two different oracles (to compare the function results by coordinates).*

It is clear that for every β *the adversary makes requests to one random oracle (for the second case the oracle is defined non-typically). So he can obtain no information about* β. *The advantage of the adversary to solve this task is equal to* 0 *independent of the computing resources.*

Further q_{exec}, q_{send}, q_H are the numbers of queries to the oracles $\mathcal{O}_{exec}$, $\mathcal{O}_{send}$, $\mathcal{O}_H$ correspondingly that the adversary $\mathcal{A}_{SESPAKE_{KA}}$ makes. In the theorem formulated below the values of these parameters can be arbitrary. In practice the limitation of these values is a very important condition for the protocol security. There are detailed comments in Section 3.9.1.

Theorem 3.13. *The following inequality holds:*

$$\mathbf{Adv}_{SESPAKE_{KA}}(t, q_H, q_{send}, q_{exec}, q_{reveal}) \leqslant$$
$$\leqslant \frac{(2q_{exec} + q_{send})^2}{q} + 2q_H \mathbf{Adv}_{\text{CDH}}(t + 2\tau q_{exec}) + 2q_{send} \mathbf{Adv}_{\text{S-QPCCDH}_{\mathcal{D}, 2q_H}}(t + q_{S1}\tau + C),$$

where C *is some constant,* τ *is a run time for computing a multiple point in the group* E, *and* q_{S1} *is a number of queries to the oracle* $\mathcal{O}_{send}$ *of the form* $(ind, salt)$.

A proof structure. Let $\mathcal{A}_{SESPAKE_{KA}}$ is an adversary, $\Pr[SUCC]$ is his success probability in Procedure 3.1. We will use the adversary $\mathcal{A}_{SESPAKE_{KA}}$ as a black box, intercept his queries to the oracles $\mathcal{O}_{exec}$, $\mathcal{O}_{send}$, $\mathcal{O}_{reveal}$ and $\mathcal{O}_{test}$ and model the work of the participants according to the protocol. Then step-by-step we will disable the adversary to get true answers from the oracles. At that we will consider the difference of the success probability for the adversary $\mathcal{A}_{SESPAKE_{KA}}$ in the step-by-step modified experiments. After some numbers of modifications we will get an experiment where the success probability is $\frac{1}{2}$. Thus a deviation of the success probability in the «clear» experiment from $\frac{1}{2}$ can be majorized by the sum of all bounds obtained in each experiment modification.

Describe a qualitative structure of the bound obtained in a such way. By modifying the conditions of the experiment in fact we disable the adversaries by some definite way and exclude from consideration some group of adversaries who can not be successful under new

conditions (i.e. who use locked abilities essentially). A group excluded with randomization the oracle $\mathcal{O}_{exec}$ answers consists of the passive adversaries who listen to the channel. The main argumentations are assigned to bound an advantage of excluded adversaries. Then the obtained bound joins as a summand to the total one. Totally if a set of the adversaries is divided into appropriate groups, e.g. into groups B_1, B_2 and B_3, then the total bound has the following form:

$$\mathbf{Adv}(A) = \max\{\mathbf{Adv}(B_1), \mathbf{Adv}(B_2), \mathbf{Adv}(B_3)\} \leqslant C_1 + C_2 + C_3,$$

where C_i is an upper bound for $\mathbf{Adv}(B_i)$.

Proof. Let $\Pr[SUCC_0]$ is the success probability for the adversary $\mathcal{A}_{SESPAKE_{KA}}$. We will use the adversary $\mathcal{A}_{SESPAKE_{KA}}$ as a black box and intercept his queries to the oracles. At that we have interest in the success probability for the adversary $\mathcal{A}_{SESPAKE_{KA}}$ in modified experiment. To process his queries to the oracles we will model a behavior of all participants: choosing a random password for each participant, generating random points on the generation key step and so on. In this case a modelled set of the participants and the set used in the «clear» experiment are the same for the adversary. Thus the probability $\Pr[SUCC_1]$ is exactly equal to $\Pr[SUCC_0]$.

Then we will use the adversary $\mathcal{A}_{SESPAKE_{KA}}$ as a black box in experiment that differs from the previous one with the valid network model. The difference is in fact that we will finish the work and return a random bit if there are the same values among the points $\{u_1\}$ and $\{u_2\}$ obtained as a result of queries made by adversary $\mathcal{A}_{SESPAKE_{KA}}$ to the oracles $\mathcal{O}_{send}$ and $\mathcal{O}_{exec}$ (modelled by us). The success probability is at most $\frac{(2q_{exec}+q_{send})^2}{2q}$. Thus the success probability $\Pr[SUCC_2]$ in this experiment differs from $\Pr[SUCC_1]$ at most by $\frac{(2q_{exec}+q_{send})^2}{2q}$. This experiment is used to consider no more the adversaries, who, for example, send the queries with the same u_1 to participant B, getting a key K_B with help of the oracle $\mathcal{O}_{reveal}$ until the value u_2 appears twice. If this happens, then the key K_A for the last session with u_2 should be set equal to the key K_B that corresponds to the first session with u_2.

In the next experiment modification (denote by $\mathbf{Exp}_3$) we will output in response to query to the oracle $\mathcal{O}_{exec}$ as a session key not $VKO(\alpha, \beta, 1)$ but a value of the random function $H'(u_1, u_2)$ unknown and inaccessible for computing by the adversary $\mathcal{A}_{SESPAKE_{KA}}$. Let the success probability in this experiment differs from the success probability in the previous one on $\triangle P$, i.e. $\triangle P = |\Pr[SUCC_2] - \Pr[SUCC_1]|$. As the function H used for key generation is random the differences between the adversary behaviors in present and previous experiments can be noticed only if he made a request $UKM \cdot \frac{m}{q} \cdot \mathrm{CDH}(u_1 + Q_{PW}, u_2 - Q_{PW})$ to the oracle $\mathcal{O}_H$. I.e. in fact with probability ΔP there is a solution of the CDH problem for the pair $(u_1 + Q_{PW}, u_2 - Q_{PW})$ among q_H requests to the oracle $\mathcal{O}_H$. We can use it to solve the CDH problem for any points Q_1 and Q_2. To do that we will use points $(Q_1 + \alpha P) - Q_{PW}$ and $(Q_2 + \beta P) + Q_{PW}$ as u_1 and u_2 in response to requests to the oracle $\mathcal{O}_{exec}$. If there is a value $\mathrm{CDH}(u_1 + Q_{PW}, u_2 - Q_{PW})$ among the requests to the oracle $\mathcal{O}_H$ (altogether q_H requests), then

$$\mathrm{CDH}(Q_1, Q_2) = \mathrm{CDH}(u_1 + Q_{PW}, u_2 - Q_{PW}) - \beta Q_1 - \alpha Q_2 - \alpha\beta P.$$

So we can conclude, that ΔP is at most $q_H \cdot Adv^{\mathrm{CDH}}(t + 2\tau q_{exec} + 3\tau)$. It means that for passively listening to channel adversary the task to distinguish a session key from a random string is not easier than the CDH problem.

On the next step the previous experiment $\mathbf{Exp}_3$ is modified with the randomization of the oracle $\mathcal{O}_{send}$: we will return random points in response to the queries that request informal cryptographic answers and form the session keys using the function H' described above (denote by $\mathbf{Exp}_4$ the new experiment). So in the new experiment the adversary obtains no information from the queries to the oracles. Thus the success probability $\Pr[SUCC_4]$ in this experiment is $\frac{1}{2}$.

Lemma 3.14. *The following inequality holds:*

$$|\Pr[SUCC_3] - \underbrace{\Pr[SUCC_4]}_{=1/2}| \leqslant q_{send} \cdot \mathbf{Adv}^{\text{S-QPCCDH}}_{|\mathcal{D}|,2q_H}(t + q_{S1}\tau + C).$$

Proof. In order to bound the difference between the success probabilities in $\mathbf{Exp}_3$ and $\mathbf{Exp}_4$ we will construct a special consequence of $q_u = q_{u_1} + q_{u_2} + 1$ (q_{u_1}, q_{u_2} are the numbers of queries to the oracle $\mathcal{O}_{send}$ with arguments u_1 and u_2 respectively) experiments $\mathbf{Hyb}_0, \ldots, \mathbf{Hyb}_{q_{u_1}+q_{u_2}}$, each of them will slightly more differ from the experiment $\mathbf{Exp}_3$ and will be little more similar to the experiment $\mathbf{Exp}_4$:

$$\mathbf{Exp}_3 = \mathbf{Hyb}_0 \rightsquigarrow \mathbf{Hyb}_1 \rightsquigarrow \cdots \rightsquigarrow \mathbf{Hyb}_{q_{u_1}+q_{u_2}} = \mathbf{Exp}_4. \tag{3.4}$$

Further in the experiments $\mathbf{Hyb}_j$ we will denote by i the number of such a query to the oracle $\mathcal{O}_{send}$ that parameter is either u_1 or (u_2, UKM) (i.e. this number takes into account such queries only).

The description of the experiment $\mathbf{Hyb}_j$: queries to all oracles except $\mathcal{O}_{send}$ are processed by the same way as in $\mathbf{Exp}_3$. Queries to the oracle $\mathcal{O}_{send}$ are processed in the following way:

- A query with number $1 \leqslant i \leqslant j$ to the oracle $\mathcal{O}_{send}$ is processed as in the experiment $\mathbf{Exp}_4$;
- A query with number $i > j$ to the oracle $\mathcal{O}_{send}$ is processed as in the experiment $\mathbf{Exp}_3$.

If P_j is the success probability of the adversary $\mathcal{A}_{SESPAKE_{KA}}$ in the experiment $\mathbf{Hyb}_j$, then

$$|\Pr[SUCC_3] - \Pr[SUCC_4]| \leqslant \sum_{j=1}^{q_u} |P_j - P_{j-1}|.$$

Now we describe a set of the adversaries D_j and D'_j, $j = 1, \ldots, q_u$, we will bound the difference $|P_j - P_{j-1}|$ using it. We will use these adversaries to solve the S-QPCCDH problem with some modifications. In particular for description of D_j we suggest that the multiplicity of the input points W and Q_s under P is known: $W = wP$ and $Q_s = zP$. For D'_j such a modification is not considered (we do not use w and z for the algorithm description).

The adversary D_j

We will denote by i the number of the next query to the oracle $\mathcal{O}_{Send}$ with parameter u_1 (sending from A to B, a pair (u_2, UKM) is returned in response to this sending under the correct state) or with parameter (u_2, UKM) (sending from A to B that A forms a key K_A under the correct state after). We will denote these queries by $S2$ and $S3$. Also D_j will process queries with parameters $(ind, salt)$ (denoted by $S1$) by the special way.

The adversary D_j processes the queries to all oracles as in $\mathbf{Exp}_3$ except the queries to the oracle $\mathcal{O}_{send}$. Queries to the oracle $\mathcal{O}_{send}$ are processed by the following way.

- Query $S1$:
  - If $i \leqslant j$, then it chooses a random x^*, returns $W + x^* \cdot P$ as an answer.
  - If $i > j$, then it processes the query as in $\mathbf{Exp}_3$.
- Query $S2$ (with parameter (u_1)):
  - If $i < j$, the query is processed as in $\mathbf{Exp}_4$.
  - If $i = j$, then it returns $(B, u_2 = W)$ in response. It returns $(st, Y = u_1)$ as an answer on the first step of the S-QPCCDH problem, receiving a password (st, PW') in response. It sets the password, shared by A and B, equal to PW', generates UKM and sets a key

    $$K_B = H_{256}((UKM \cdot \frac{m}{q} \cdot (w - z \cdot r') \mod q) \cdot (u_1 - r' \cdot Q_s)),$$

    where $r' = F(PW', salt, 2000)$.
  - If $i > j$, the query is processed as in $\mathbf{Exp}_3$.
- A query $S3$ (with parameter $(u2, UKM)$):
  - If $i < j$, the query is processed as in $\mathbf{Exp}_4$.
  - If $i = j$, then it returns $(st, Y = u_2)$ as an answer on the first step of the S-QPCCDH problem. Receives an answer (st, PW'). Sets PW' as a password, shared by A and B. Sets a key

    $$K_A = H_{256}((UKM \cdot \frac{m}{q} \cdot (w + x^* - z \cdot r') \mod q) \cdot (u_2 - r' \cdot Q_s)),$$

    where $r' = F(PW', salt, 2000)$.
  - If $i > j$, the query is processed as in $\mathbf{Exp}_3$.

The adversary D_j extracts keys K (i.e. a point multiplied by $UKM \cdot \frac{m}{q}$) from the queries to the oracle $\mathcal{O}_H$. He forms two points $K' = K$ and $K'' = K - x^* \cdot (Y - r \cdot Q_s)$ for each point K. Then he returns a list of $2q_H$ points $\{K'_1, K''_1, K'_2, K''_2, \ldots, K'_{q_H}, K''_{q_H}\}$ as an answer.

The adversary D'_j differs from D_j in fact that he uses an inaccessible for the adversary $\mathcal{A}_{SESPAKE_{KA}}$ function H' to generate a key during the queries $S2$ and $S3$ processing.

We will note the important facts that allow us to bound the difference $|P_j - P_{j-1}|$.

From the point of $\mathcal{A}_{SESPAKE_{KA}}$ the behavior of the adversary D'_j, i.e. the way of processing the queries to the oracles $\mathcal{O}_{send}$, $\mathcal{O}_{exec}$ and so on, totally coincides with the experiment

$\mathbf{Hyb}_j$ because the «honest» processing the queries to the oracle $\mathcal{O}_{send}$ starts with $j+1$-th query. For the adversary D_j the «honest» processing starts with j-th query. For example, independently on parameter u_1 in query $S2$ a multiplier for a point $u_1 - r' \cdot Q_s$ is equal to multiplicity of the point returned as an answer, i. e. W after deleting a «mask» $r'Q_s$. So A, receiving this answers, gets the same key as B gets if A was initially modelled according to the protocol (i.e. from the query $S1$).

The adversary behavior in experiments D_j and D'_j will be different if he makes a query to the oracle $\mathcal{O}_H$ with parameters among which there is a value $\mathrm{CDH}(u_1 - r'Q_s, u_2 - r'Q_s)$ where one of the values u_1 and u_2 was in j-th query $S2$ or $S3$. If there was such a parameter among the queries to $\mathcal{O}_H$, then we can find $\mathrm{CDH}(W - r'Q_s, Y - r'Q_s)$, as

$$K_A = \mathrm{CDH}(u_2 - r'Q_s, W + x^*P - r'Q_s) = \mathrm{CDH}(W - r'Q_s, Y - r'Q_s) + x^* \cdot (Y - r'Q_s),$$

$$K_B = \mathrm{CDH}(u_1 - r'Q_s, W - r'Q_s) = \mathrm{CDH}(W - r'Q_s, Y - r'Q_s).$$

Taking into account the way we form the list with $2q_H$ points of the adversary D'_j we can conclude that there is a required value in it. Thus the probability that there will be $\mathrm{CDH}(u_1 - r'Q_s, u_2 - r'Q_s)$ with u_1, u_2 from the j-th query $S2$ or $S3$ among the parameters of queries to the oracle $\mathcal{O}_H$ can be majorized with the highest success probability of solving the S-QPCCDH problem by the adversary running in $t + q_{S1}\tau + C$, where q_{S1} is the number of queries $S1$ to participant A, C is some constant

So the following inequality holds:

$$|\Pr[SUCC_3] - \underbrace{\Pr[SUCC_4]}_{=1/2}| \leqslant \sum_{j=1}^{q_u} |P_j - P_{j-1}| \leqslant q_{send} \cdot \mathbf{Adv}^{\text{S-QPCCDH}}_{|\mathcal{D}|,2q_H}(t + q_{S1}\tau + C).$$

□

Taking into account all bounds obtained above, we can conclude, that the following inequality holds:

$$|\Pr[SUCC_0] - \frac{1}{2}| \leqslant \frac{(2q_{exec} + q_{send})^2}{2q} + q_H \mathbf{Adv}^{\text{CDH}}(t + 2\tau q_{exec}) + q_{send} \cdot \mathbf{Adv}^{\text{S-QPCCDH}}_{|\mathcal{D}|,2q_H}(t + q_{S1}\tau + C).$$

The final inequality for $\mathbf{Adv}^{SESPAKE}(t, q_H, q_{send}, q_{exec}, q_{reveal})$ directly follows from this relation.

□

3.7 A proof of security of the *SESPAKE* protocol

3.7.1 Used subtasks

For argumentations of the *SESPAKE* protocol security besides the model described in Section 3.4 a few additional tasks are introduced as intermediate. The difference between the models consists in the way the oracle $\mathcal{O}_{auth}$ works.

One specification of these tasks should be noticed. The oracles used for a threat formulation for $SESPAKE_{KA}$ and *SESPAKE* return the «honest» data if $b = 1$ and random data if $b = 0$. For tasks 1HMAC and 2HMAC we changed the meaning of value returned by the oracle $\mathcal{O}_{auth}$. Here «more honest» data is returned if $b = 0$ not if $b = 1$. It was done to bound the main parameters.

3.7.2 A task 1HMAC

In this task the adversary has an access to all oracles as in the model for the protocol *SESPAKE*. The difference consists in fact that the oracle $\mathcal{O}_{auth}$ returns not a pair of strings but either M'_A (if $b = 1$) or M_A (if $b = 0$) only.

3.7.3 A task 2HMAC

In task 2HMAC the oracle $\mathcal{O}_{auth}$ returns in response to the adversary either strings (M'_A, M') (if $b = 1$) or strings (M_A, M'') (if $b = 0$). The strings are defined in the following way:

1. $M_A = \mathrm{HMAC}_{K_{A,i}}(T_A||IDS_{A,i})$,
2. $M'_A = \mathrm{HMAC}_{K'}(T_A||IDS_{A,i})$ and $M' = \mathrm{HMAC}_{K'}(T_B||IDS_{A,i})$, where $K' \in_R V_{256}$.
3. $M'' = \mathrm{HMAC}_{K''}(T_B||IDS_{A,i})$, where $K'' \in_R V_{256}$,

3.7.4 The bounds of security

It should be noticed that the advantage $\mathbf{Adv}(\mathcal{A})$ of some adversary who solves a task of a bit b recognition can be modified by the following way:

$$\mathbf{Adv}(\mathcal{A}) = 2\Pr[SUCC] - 1 = 2\left[\frac{1}{2}\Pr[\mathcal{A} = 1|b = 1] + \frac{1}{2}(1 - \Pr[\mathcal{A} = 1|b = 0])\right] - 1 = \\ = \Pr[\mathcal{A} = 1|b = 1] - \Pr[\mathcal{A} = 1|b = 0].$$

Further we will bound the value modified by this way in our argumentations.

Theorem 3.15. *The following relation holds:*

$$\mathbf{Adv}_{1\mathrm{HMAC}}(t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}, q_{check}) \leqslant \\ \leqslant \mathbf{Adv}_{SESPAKE_{KA}}(t + q_{\mathrm{HMAC}} + q_{check}, q_{exec}, q_{send}, q_H) + O\left(\frac{1}{2^n}\right).$$

Proof. Suppose $\mathcal{A}$ is an adversary who solves a task 1HMAC. We will describe an adversary $\mathcal{A}'$ who realises a threat to distinguish a key for the protocol $SESPAKE_{KA}$ and uses $\mathcal{A}$ as a black box. $\mathcal{A}'$ transmits all queries to the available for him oracles ($\mathcal{O}_{exec}$, $\mathcal{O}_{send}$, $\mathcal{O}_{reveal}$ and $\mathcal{O}_H$) and their answers without changes. For the query to the oracle $\mathcal{O}_{check}$ the adversary $\mathcal{A}'$ makes a request to the oracle $\mathcal{O}_{reveal}$, obtaining a key for the oracle that the adversary $\mathcal{A}$ wants to get authentication data for. Then $\mathcal{A}'$ returns to $\mathcal{A}$ the «honest» required data, using the obtained key and the oracle $\mathcal{O}_{\mathrm{HMAC}}$. For the query to the oracle $\mathcal{O}_{auth}$ $\mathcal{A}'$ makes a request to the oracle $\mathcal{O}_{test}$, obtaining either a random key K' or the real key K stored by the testing oracle. Then $\mathcal{A}'$ transmits to $\mathcal{A}$ a value HMAC computed with an obtained key. $\mathcal{A}'$ returns a value converse to $\mathcal{A}$ result.

From the description of the experiment we can see that $\mathcal{A}'$ models the experiment conditions of the adversary $\mathcal{A}$ perfectly. The advantage $\mathbf{Adv}_{SESPAKE_{KA}}(\mathcal{A}')$ differs from the advantage $\mathbf{Adv}_{1\mathrm{HMAC}}(\mathcal{A})$ at most by value of order 2^{-n} explained that a value HMAC

computed on a random key can coincide with a value HMAC computed on the session key of the attacked oracle. Thus

$$\mathbf{Adv}_{SESPAKE_{KA}}(\mathcal{A}') \geqslant \mathbf{Adv}_{1\mathrm{HMAC}}(\mathcal{A}) - O\left(\frac{1}{2^n}\right).$$

As $\mathbf{Adv}_{SESPAKE_{KA}}(t + q_{\mathrm{HMAC}} + q_{check}, q_{exec}, q_{send}, q_H) \geqslant \mathbf{Adv}_{SESPAKE_{KA}}(\mathcal{A}')$, then

$$\mathbf{Adv}_{1\mathrm{HMAC}}(\mathcal{A}) = \mathbf{Adv}_{1\mathrm{HMAC}}(t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}, q_{check}) \leqslant$$
$$\leqslant \mathbf{Adv}_{SESPAKE_{KA}}(t + q_{\mathrm{HMAC}} + q_{check}, q_{exec}, q_{send}, q_H) + O\left(\frac{1}{2^n}\right).$$

□

Theorem 3.16. *The following relation holds:*

$$\mathbf{Adv}_{2\mathrm{HMAC}}(t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}, q_{check}) \leqslant$$
$$\leqslant \mathbf{Adv}_{1\mathrm{HMAC}}(t+1, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}, q_{check}+1) + \frac{q_{\mathrm{HMAC}}}{2^{n-1}}.$$

Proof. Suppose $\mathcal{A} \in \mathcal{A}(t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}, q_{check})$ is an adversary who solves the task 2HMAC with an advantage $\mathbf{Adv}_{2\mathrm{HMAC}}(t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}, q_{check})$. We will now construct an adversary $\mathcal{A}'$ who uses $\mathcal{A}$ as a black box and solves the task 1HMAC.

$\mathcal{A}'$ changes nothing in the interactions of $\mathcal{A}$ with all available for him oracles except for $\mathcal{O}_{auth}$. If $\mathcal{A}$ makes a request to the oracle $\mathcal{O}_{auth}$, then $\mathcal{A}'$ makes a request to his oracle $\mathcal{O}_{auth}$ and transmits to $\mathcal{A}$ its answer and a value M'' of the HMAC function computed with a random key K''. As a result the adversary $\mathcal{A}$ distinguishes the following situations: he has received a pair (M_A, M'') or a pair (M'_A, M''). The difference between the «honest» and current experiments consists in fact that a pair (M'_A, M'') is generated with two independent random keys.

The following relation holds:

$$\mathbf{Adv}_{1\mathrm{HMAC}}(\mathcal{A}') = \Pr[\mathcal{A}=1|\mathcal{A} \leftarrow (M'_A, M'')] - \Pr[\mathcal{A}=1|\mathcal{A} \leftarrow (M_A, M'')] =$$
$$= \Pr[\mathcal{A}=1|\mathcal{A} \leftarrow (M'_A, M'')] - \Pr[\mathcal{A}=1|\mathcal{A} \leftarrow (M'_A, M')] +$$
$$+ \underbrace{\Pr[\mathcal{A}=1|\mathcal{A} \leftarrow (M'_A, M')] - \Pr[\mathcal{A}=1|\mathcal{A} \leftarrow (M_A, M'')]}_{=\mathbf{Adv}_{2\mathrm{HMAC}}(\mathcal{A})} \geqslant$$
$$\geqslant -\frac{2q_{\mathrm{HMAC}}}{2^n} + \mathbf{Adv}_{2\mathrm{HMAC}}(\mathcal{A}).$$

Indeed, the first difference is not greater than the advantage of the best adversary who works with parameters of the adversary $\mathcal{A}$ and solves a task to distinguish a pair of random function outputs obtained with one random key from the pair obtained with two random keys. The adversary $\mathcal{A}$ can do it only if there is a key K' or a key K'' of the oracle $\mathcal{O}_{auth}$ among his queries to $\mathcal{O}_{\mathrm{HMAC}}$. As both keys are chosen at random the probability of such event is at most $2q_{\mathrm{HMAC}}/2^n$.

Also notice that the adversary $\mathcal{A}'$ generates a key K'' additionally and makes an additional request to $\mathcal{O}_{\mathrm{HMAC}}$.

As a result we have

$$\mathbf{Adv}_{2\mathrm{HMAC}}(\mathcal{A}') \leqslant \mathbf{Adv}_{1\mathrm{HMAC}}(\mathcal{A}') + \frac{q_{\mathrm{HMAC}}}{2^{n-1}} =$$
$$= \mathbf{Adv}_{1\mathrm{HMAC}}(t+1, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}, q_{check}+1) + \frac{q_{\mathrm{HMAC}}}{2^{n-1}}.$$

□

Theorem 3.17. *The following relation holds:*

$$\mathbf{Adv}_{SESPAKE}(t, q_{exec}, q_{send}, q_{check}, q_H, q_{\mathrm{HMAC}}) \leqslant$$
$$\leqslant 2\mathbf{Adv}_{SESPAKE_{KA}}(t + q_{\mathrm{HMAC}} + q_{check} + 2, q_{exec}, q_{send}, q_H) + \frac{q_{\mathrm{HMAC}}}{2^{n-1}} + O\left(\frac{1}{2^n}\right).$$

Proof. Suppose $\mathcal{A} \in \mathcal{A}(t, q_{exec}, q_{send}, q_{check}, q_H, q_{\mathrm{HMAC}})$ is an arbitrary adversary who realizes a threat for the *SESPAKE* protocol with the advantage $\mathbf{Adv}_{SESPAKE}(\mathcal{A})$. We will describe an adversary $\mathcal{A}'$ who uses $\mathcal{A}$ to realize a threat for the $SESPAKE_{KA}$ protocol. $\mathcal{A}'$ uses $\mathcal{A}$ by the way described in the theorem 3.15. So if the oracle $\mathcal{O}_{test}$ returns a random key K', then the adversary $\mathcal{A}$ gets in response to query to $\mathcal{O}_{auth}$ a pair $(M'_A, M') = (\mathrm{HMAC}_{K'}(T_A||IDS_{A,i}), \mathrm{HMAC}_{K'}(T_B||IDS_{A,i}))$, otherwise the adversary $\mathcal{A}$ gets a pair $(M_A, M) = (\mathrm{HMAC}_{K_{A,i}}(T_A||IDS_{A,i}), \mathrm{HMAC}_{K_{A,i}}(T_B||IDS_{A,i}))$. Since $\mathcal{A}'$ returns in response that the adversary $\mathcal{A}$ returns, hence for the advantage $\mathbf{Adv}(\mathcal{A}')$ the following relation holds:

$$\mathbf{Adv}(\mathcal{A}') = \Pr[\mathcal{A}=1|(M_A, M)] - \Pr[\mathcal{A}=1|(M'_A, M')] =$$
$$= \underbrace{\Pr[\mathcal{A}=1|(M_A, M)] - \Pr[\mathcal{A}=1|(M_A, M'')]}_{S} +$$
$$+ \underbrace{\Pr[\mathcal{A}=1|(M_A, M'')] - \Pr[\mathcal{A}=1|(M'_A, M')]}_{T},$$

where $M'' = \mathrm{HMAC}_{K''}(T_B||IDS_{A,i})$ and $K'' \in_R V_{256}$. Notice, that S is exactly equal to $\mathbf{Adv}_{SESPAKE}(\mathcal{A})$.

Let us bound value T, i.e. the advantage of the adversary $\mathcal{A}$ for the task to distinguish the following answers of the oracle $\mathcal{O}_{auth}$: (M_A, M'') and (M'_A, M'). We will majorize the value $-T$ getting the lower bound for T. The relation $-T = \Pr[\mathcal{A}=1|(M'_A, M')] - \Pr[\mathcal{A}=1|(M_A, M'')] \leqslant \mathbf{Adv}_{2\mathrm{HMAC}}(t, q_{exec}, q_{send}, q_{check}, q_H, q_{\mathrm{HMAC}})$ holds because $\mathcal{A}$ is a particular adversary from the set $\mathcal{A}(t, q_{exec}, q_{send}, q_{check}, q_H, q_{\mathrm{HMAC}})$, and the advantage in the right part of the inequality specifies the most successful adversary from this set for the task 2HMAC. Using the theorem 3.15 and the theorem 3.16, we obtain the inequality

$$T \geqslant -\mathbf{Adv}_{SESPAKE_{KA}}(t + q_{\mathrm{HMAC}} + q_{check} + 2, q_{exec}, q_{send}, q_H) - \frac{q_{\mathrm{HMAC}}}{2^{n-1}} - O\left(\frac{1}{2^n}\right).$$

As a result we obtain the inequality:

$$\mathbf{Adv}_{SESPAKE_{KA}}(t + q_{\mathrm{HMAC}} + q_{check}, q_{exec}, q_{send}, q_H) \geqslant \mathbf{Adv}_{SESPAKE_{KA}}(\mathcal{A}') \geqslant$$
$$\geqslant \mathbf{Adv}_{SESPAKE}(\mathcal{A}) - \mathbf{Adv}_{SESPAKE_{KA}}(t + q_{\mathrm{HMAC}} + q_{check} + 2, q_{exec}, q_{send}, q_H) - \frac{q_{\mathrm{HMAC}}}{2^{n-1}} - O\left(\frac{1}{2^n}\right).$$

Thus

$$\mathbf{Adv}_{SESPAKE}(\mathcal{A}) = \mathbf{Adv}_{SESPAKE}(t, q_{exec}, q_{send}, q_{check}, q_H, q_{\mathrm{HMAC}}) \leqslant$$
$$\leqslant 2\mathbf{Adv}_{SESPAKE_{KA}}(t + q_{\mathrm{HMAC}} + q_{check} + 2, q_{exec}, q_{send}, q_H) + \frac{q_{\mathrm{HMAC}}}{2^{n-1}} + O\left(\frac{1}{2^n}\right).$$

□

The main point of the bound proved in theorem 3.17 is in fact that adding an authentication step to the protocol $SESPAKE_{KA}$ gives to the adversary just a criterion to check a session key, but does not give any new abilities for the password-revealing attacks. So the best way to reveal the password is to guess it with an active involvement in the channel interaction.

3.8 The adversary model «without warning»

Previously we have considered the advantage $\mathbf{Adv}_{SESPAKE}$ of the adversary $\mathcal{A}_{SESPAKE}$ who uses the queries to the oracle $\mathcal{O}_{check}$ to warn which session he will not try to realize a threat to. The theorem proved above refers to this particular adversary model. Denote his advantage by $\mathbf{Adv}_{SESPAKE,w}$.

Now we consider the advantage $\mathbf{Adv}_{SESPAKE}$ of the adversary $\mathcal{A}_{SESPAKE}$ who can make a request to the oracle $\mathcal{O}_{auth}$ for any protocol participant no matter whether there was a request to the oracle $\mathcal{O}_{check}$ or not. This adversary works under the same conditions except for the response form of query to the oracle $\mathcal{O}_{auth}$:

- The oracle $\mathcal{O}_{auth}$ receives as an input a pair (A, i), chooses a value $b \in \{0, 1\}$ uniformly at random and returns either M' (if $b = 0$) or M_B (if $b = 1$). Here $M' = \mathrm{HMAC}_{K'}(T_B||IDS_{A,i})$, where $K' \in_R V_{256}$, $M_B = \mathrm{HMAC}_{K_{A,i}}(T_B||IDS_{A,i})$.

The adversary can make a request to the oracle $\mathcal{O}_{auth}$ for fresh oracles only. A fresh oracle is such an oracle $\mathcal{O}_A^i$ for which the value M_B has not been got yet by the adversary in the trivial way, i. e. he has not made a request to the oracle $\mathcal{O}_{check}$ for the oracle sharing with $\mathcal{O}_A^i$ a common key.

Let bound the advantage $\mathbf{Adv}_{SESPAKE}$.

Theorem 3.18. *The following relation holds:*

$$\mathbf{Adv}_{SESPAKE}(t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}) \leqslant$$
$$\leqslant q_{send} \cdot \mathbf{Adv}_{SESPAKE,w}(t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}) +$$
$$+ 2q_H \cdot \mathbf{Adv}_{\mathrm{CDH}}(t + \tau(2q_{exec} + q_{send})) + O\left(\frac{q_H + q_{\mathrm{HMAC}}}{2^n}\right).$$

Proof. Suppose $\mathcal{A}_{SESPAKE}$ is an adversary in the model described above with parameters $t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}$ and the advantage $\mathbf{Adv}_{SESPAKE}(t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}})$.

Consider two auxiliary adversaries $\mathcal{A}_{SESPAKE_{send}}$ and $\mathcal{A}_{SESPAKE_{exec}}$ based on the adversary $\mathcal{A}_{SESPAKE}$.

We will say that a session is a *Send*-session if it was initiated with a query to the oracle $\mathcal{O}_{send}$ and *Exec*-session if it was initiated with a query to the oracle $\mathcal{O}_{exec}$. Denote by s_{auth} a session for which the adversary made a request to the oracle $\mathcal{O}_{auth}$. A fact that the session s_{auth} is a *Send*-session (*Exec*-session) will be denoted by $s_{auth} \in Send$ ($s_{auth} \in Exec$).

The adversary $\mathcal{A}_{SESPAKE_{send}}$ uses the adversary $\mathcal{A}_{SESPAKE}$ as a black box and works as translator between him and the oracles up to request to the oracle $\mathcal{O}_{auth}$. If for the adversary $\mathcal{A}_{SESPAKE}$ the request to the oracle $\mathcal{O}_{auth}$ was made for *Send*-session ($s_{auth} \in Send$), then $\mathcal{A}_{SESPAKE_{send}}$ outputs the same bit as $\mathcal{A}_{SESPAKE}$ outputs. If the request was made for *Exec*-session ($s_{auth} \in Exec$), then $\mathcal{A}_{SESPAKE_{send}}$ outputs a random bit independently of the oracle $\mathcal{O}_{auth}$ response. Denote by $SUCC_{SESPAKE}$, $SUCC_{SESPAKE_{send}}$ and $SUCC_{SESPAKE_{exec}}$ the events that occur if a task of the false authentication was solved successfully by the adversaries $\mathcal{A}_{SESPAKE}$, $\mathcal{A}_{SESPAKE_{send}}$ and $\mathcal{A}_{SESPAKE_{exec}}$ respectively.

We have:

$$\begin{aligned}\Pr[\, SUCC_{SESPAKE_{send}} \,] = \\ = \Pr[\, SUCC_{SESPAKE} \mid s_{auth} \in Send \,] \cdot \Pr[\, s_{auth} \in Send \,] + \\ + \frac{1}{2} \cdot \Pr[\, s_{auth} \in Exec \,], \quad (3.5)\end{aligned}$$

where $\Pr[\, s_{auth} \in Send \,]$ ($\Pr[\, s_{auth} \in Exec \,]$) is the probability for the adversary to make a request to the oracle $\mathcal{O}_{auth}$ for *Send*-session (*Exec*-session).

Remember that $\Pr[\, s_{auth} \in Send \,] + \Pr[\, s_{auth} \in Exec \,] = 1$.

The adversary $\mathcal{A}_{SESPAKE_{exec}}$ works in the same way but he outputs the same bit as $\mathcal{A}_{SESPAKE}$ outputs if there was a request for *Exec*-session, and outputs a random bit if there was a request for *Send*-session:

$$\begin{aligned}\Pr[\, SUCC_{SESPAKE_{exec}} \,] = \\ = \Pr[\, SUCC_{SESPAKE} \mid s_{auth} \in Exec \,] \cdot \Pr[\, s_{auth} \in Exec \,] + \\ + \frac{1}{2} \cdot \Pr[\, s_{auth} \in Send \,], \quad (3.6)\end{aligned}$$

Let us bound the advantage for the adversary $\mathcal{A}_{SESPAKE_{exec}}$.

Lemma 3.19. *The following inequality holds:*

$$\mathbf{Adv}\,(\mathcal{A}_{SESPAKE_{exec}}) \leqslant 2q_H \cdot \mathbf{Adv}_{\mathrm{CDH}}(t + \tau(2q_{exec} + q_{send}) + 3\tau) + O\left(\frac{q_H + q_{\mathrm{HMAC}}}{2^n}\right).$$

Proof. Suppose $\mathcal{A}_{SESPAKE_{exec}}$ is an adversary with parameters $t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}$ who solves the task of the false authentication with the advantage $\mathbf{Adv}\,(\mathcal{A}_{SESPAKE_{exec}})$. We will construct an adversary $\mathcal{A}_{\mathrm{CDH}}$ solving the computational Diffie-Hellman problem based on the adversary $\mathcal{A}_{SESPAKE_{exec}}$ and will bound his success probability $\mathbf{Adv}\,(\mathcal{A}_{\mathrm{CDH}})$.

Suppose the adversary $\mathcal{A}_{\mathrm{CDH}}$ needs to solve the CDH problem for the arbitrary points Q_1 and Q_2. We will use the adversary $\mathcal{A}_{SESPAKE_{exec}}$ as a black box.

The adversary $\mathcal{A}_{\mathrm{CDH}}$ starts up the adversary $\mathcal{A}_{SESPAKE_{exec}}$ and simulates a protocol execution, where he uses as u_1 and u_2 (a response to query to the oracle $\mathcal{O}_{exec}$) the points

$(Q_1 + \alpha P) - Q_{PW}$ и $(Q_2 + \beta P) + Q_{PW}$, where α and β are random values. Here the adversary uses as a session key the value of the random function $H'(u_1, u_2)$ unknown and inaccessible for computing by the adversary $\mathcal{A}_{SESPAKE_{KA}}$ for queries to the oracles $\mathcal{O}_{check}$ and $\mathcal{O}_{auth}$. Then he chooses one of the q_H queries to the oracle $\mathcal{O}_H$ (denote this query by R) made by the adversary $\mathcal{A}_{SESPAKE_{exec}}$ and computes the value T by the following way:

$$T = R - \beta Q_1 - \alpha Q_2 - \alpha\beta P,$$

where the corresponding α and β are chosen using the values u_1 and u_2 from the queries to the oracle $\mathcal{O}_{\text{HMAC}}$ and from the responses of the oracle $\mathcal{O}_H$. The probability that there will be the same keys for different α and β is $O\left(\frac{1}{2^n}\right)$.

If the adversary $\mathcal{A}_{SESPAKE_{exec}}$ computes athe value $R = \text{CDH}(u_1 + Q_{PW}, u_2 - Q_{PW})$ correctly, then T is required value of $\text{CDH}(Q_1, Q_2)$. Thus the advantage for the constructed adversary $\mathbf{Adv}\,(\mathcal{A}_{\text{CDH}})$ is $\Pr[T = \text{CDH}(Q_1, Q_2)]$.

Denote by A an event that occurs if there is a value used to compute the required $\text{CDH}(Q_1, Q_2)$ among the queries to the oracle $\mathcal{O}_H$.

Then we have:

$$\mathbf{Adv}_{\text{CDH}}(t + \tau(2q_{exec} + q_{send}) + 3\tau) \geqslant \mathbf{Adv}\,(\mathcal{A}_{\text{CDH}}) = \frac{1}{q_H} \cdot \Pr[\ A\ ].$$

Consider the advantage $\mathbf{Adv}\,(\mathcal{A}_{SESPAKE_{exec}}) = 2 \cdot \Pr[\ SUCC_{SESPAKE_{exec}}\ ] - 1$ of the adversary $\mathcal{A}_{SESPAKE_{exec}}$.

$$\begin{aligned}\Pr[\ SUCC_{SESPAKE_{exec}}\ ] = \underbrace{\Pr[\ SUCC_{SESPAKE_{exec}} \mid A\ ]}_{\leqslant 1} \cdot \Pr[\ A\ ] + \\ + \Pr\left[\ SUCC_{SESPAKE_{exec}} \mid \overline{A}\ \right] \cdot \underbrace{(1 - \Pr[\ A\ ])}_{\leqslant 1} \leqslant \\ \leqslant \Pr[\ A\ ] + \Pr\left[\ SUCC_{SESPAKE_{exec}} \mid \overline{A}\ \right]. \quad (3.7)\end{aligned}$$

Consider the case when the adversary needs to guess a bit b upon the condition that there is no value $\text{CDH}(u_1 + Q_{PW}, u_2 - Q_{PW})$ among the queries to the oracle $\mathcal{O}_H$, and bound his success probability $\Pr\left[\ SUCC_{SESPAKE_{exec}} \mid \overline{A}\ \right]$.

Due to randomness of the H function, data obtained in response to the query to the oracle $\mathcal{O}_H$ with the values that are not equal to $\text{CDH}(u_1 + Q_{PW}, u_2 - Q_{PW})$ are random and thus consist no information about the session keys. So the adversary needs to solve the following task: to define using the responses of the oracles $\mathcal{O}_{check}$ and $\mathcal{O}_{auth}$ only what key was used to compute the HMAC value. As for the adversary keys seem equal for cases $b = 0$ and $b = 1$, he can guess the right bit value with probability close to 1 to an accuracy of $O\left(\frac{1}{2^n}\right)$ using the following strategy only: choose a random key, compute HMAC using this key and compare obtained values with responses of the oracles $\mathcal{O}_{check}$ and $\mathcal{O}_{auth}$. I.e. if HMAC is computed with the chosen key is equal to responses of both oracles $\mathcal{O}_{check}$ and $\mathcal{O}_{auth}$, then the adversary outputs $b' = 1$, and vice versa, if HMAC is computed with chosen key is equal to the responses of either oracle $\mathcal{O}_{check}$ or the oracle $\mathcal{O}_{auth}$, then the adversary outputs $b' = 0$.

Approximate success probability is explained with the probability to mistake because of the function H randomness.

Then

$$\Pr\left[\ SUCC_{SESPAKE_{exec}} \mid \overline{A}\ \right] = \frac{1}{2} + O\left(\frac{q_H + q_{\text{HMAC}}}{2^n}\right).$$

Taking into account (3.7), we have

$$2 \cdot \Pr\left[\ A\ \right] \geqslant \mathbf{Adv}\left(\mathcal{A}_{SESPAKE_{exec}}\right) - O\Big(\frac{q_H + q_{\text{HMAC}}}{2^n}\Big).$$

Thus

$$\mathbf{Adv}_{\text{CDH}}(t + \tau(2q_{exec} + q_{send}) + 3\tau) \geqslant \frac{1}{2q_H} \cdot \left(\mathbf{Adv}\left(\mathcal{A}_{SESPAKE_{exec}}\right) - O\Big(\frac{q_H + q_{\text{HMAC}}}{2^n}\Big)\right).$$

□

By the law of total probability we have

$$\begin{aligned}\Pr\left[\ SUCC_{SESPAKE}\ \right] = \\ = \Pr\left[\ SUCC_{SESPAKE} \mid s_{auth} \in Send\ \right] \cdot \Pr\left[\ s_{auth} \in Send\ \right] + \\ + \Pr\left[\ SUCC_{SESPAKE} \mid s_{auth} \in Exec\ \right] \cdot \Pr\left[\ s_{auth} \in Exec\ \right]. \quad (3.8)\end{aligned}$$

Transform (3.5) и (3.6):

$$\begin{aligned}\Pr\left[\ SUCC_{SESPAKE} \mid s_{auth} \in Send\ \right] \cdot \Pr\left[\ s_{auth} \in Send\ \right] = \\ = \Pr\left[\ SUCC_{SESPAKE_{send}}\ \right] - \frac{1}{2} \cdot \Pr\left[\ s_{auth} \in Exec\ \right], \quad (3.9)\end{aligned}$$

$$\begin{aligned}\Pr\left[\ SUCC_{SESPAKE} \mid s_{auth} \in Exec\ \right] \cdot \Pr\left[\ s_{auth} \in Exec\ \right] = \\ = \Pr\left[\ SUCC_{SESPAKE_{exec}}\ \right] - \frac{1}{2} \cdot \Pr\left[\ s_{auth} \in Send\ \right]. \quad (3.10)\end{aligned}$$

Using (3.9) and (3.10) in the right of the expression (3.8), we get finally:

$$\Pr\left[\ SUCC_{SESPAKE_{send}}\ \right] = \Pr\left[\ SUCC_{SESPAKE}\ \right] - \Pr\left[\ SUCC_{SESPAKE_{exec}}\ \right] + \frac{1}{2}.$$

or

$$\begin{aligned}\mathbf{Adv}_{SESPAKE_{send}}(t, q_{exec}, q_{send}, q_H, q_{\text{HMAC}}) = \\ = \mathbf{Adv}_{SESPAKE}(t, q_{exec}, q_{send}, q_H, q_{\text{HMAC}}) - \\ - \mathbf{Adv}_{SESPAKE_{exec}}(t, q_{exec}, q_{send}, q_H, q_{\text{HMAC}}).\end{aligned}$$

Using the lemma we get the following inequality:

$$\begin{aligned}\mathbf{Adv}_{SESPAKE_{send}}(t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}) \geqslant \\ \geqslant \mathbf{Adv}_{SESPAKE}(t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}) - \\ - 2q_H \cdot \mathbf{Adv}_{\mathrm{CDH}}(t + \tau(2q_{exec} + q_{send})) - O\left(\frac{q_H + q_{\mathrm{HMAC}}}{2^n}\right).\end{aligned}$$

Now we will construct an adversary «with warning» $\mathcal{A}_{SESPAKE,w}$ based on the adversary «without warning» $\mathcal{A}_{SESPAKE}$ using him as a black box. For the intelligent adversary «with warning» it is more logical to make a request to the oracle $\mathcal{O}_{auth}$ for the participant of the *Send*-session, as advantage of the adversary $\mathcal{A}_{SESPAKE_{exec}}$ is comparable with the success probability to solve the CDH problem. Therefore, we can consider the adversary $\mathcal{A}_{SESPAKE_{send}}$ instead of the adversary $\mathcal{A}_{SESPAKE}$. In the process of attack the adversary «with warning» working as a translator between the oracles and the adversary $\mathcal{A}_{SESPAKE_{send}}$ chooses one of the *Send*-session initiated by $\mathcal{A}_{SESPAKE_{send}}$ uniformly at random and makes a request to the oracle $\mathcal{O}_{auth}$ for the participant of the chosen session.

If the adversary $\mathcal{A}_{SESPAKE_{send}}$ makes a request to the oracle $\mathcal{O}_{check}$ for the participant of the session chosen by $\mathcal{A}_{SESPAKE,w}$, then the adversary $\mathcal{A}_{SESPAKE,w}$ returns to $\mathcal{A}_{SESPAKE_{send}}$ the first part of the oracle $\mathcal{O}_{auth}$ response. If the adversary $\mathcal{A}_{SESPAKE_{send}}$ makes a request to the oracle $\mathcal{O}_{check}$ for the participant of another session, then the adversary $\mathcal{A}_{SESPAKE,w}$ just transmits this request.

If the adversary $\mathcal{A}_{SESPAKE_{send}}$ makes a request to the oracle $\mathcal{O}_{auth}$ for the participant of the session chosen by $\mathcal{A}_{SESPAKE,w}$, then the adversary $\mathcal{A}_{SESPAKE,w}$ returns to $\mathcal{A}_{SESPAKE_{send}}$ the second part of the oracle $\mathcal{O}_{auth}$ response and outputs as a result the same bit that the adversary $\mathcal{A}_{SESPAKE_{send}}$ outputs. If the adversary $\mathcal{A}_{SESPAKE_{send}}$ makes a request to the oracle $\mathcal{O}_{auth}$ for the participant of another session, the adversary $\mathcal{A}_{SESPAKE,w}$ chooses a random key, computes M' using this key and sends it to the adversary $\mathcal{A}_{SESPAKE_{send}}$. As a results he outputs a random bit.

For the constructed adversary $\mathcal{A}_{SESPAKE,w}$ the probability that the chosen session coincides with the session the adversary $\mathcal{A}_{SESPAKE_{send}}$ made a request to the oracle $\mathcal{O}_{auth}$ for is $\frac{1}{q_{send}}$. Thus the advantage for the adversary $\mathcal{A}_{SESPAKE,w}$ is smaller than the advantage of the adversary $\mathcal{A}_{SESPAKE_{send}}$ at most in q_{send}:

$$\mathbf{Adv}_{SESPAKE,w}(t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}) \geqslant \frac{1}{q_{send}} \cdot \mathbf{Adv}_{SESPAKE_{send}}(t, q_{exec}, q_{send}, q_H, q_{\mathrm{HMAC}}).$$

From this we get a required bound. □

3.9 Some comments and explanations

In this section some constructive features of the *SESPAKE* protocol and its influence on the properties are discussed.

3.9.1 The sessions number limitation for one password

Informally the *SESPAKE* protocol is secure if the best way to find a session key (or some information about it) is the testing a password by force of interaction with network participant ("online"-guessing a password). An inadmissible alternative for the protocol security is a practical brute force that doesn't required such interaction ("offline"-selection).

For the "online"-guessing every unsuccessful attempt leads to disconnection caused by the validation failure according to the $SESPAKE_{KC}$ protocol. The absence of the fail attempts limitation increases a probability of "online"-guessing the password. That is the reason, why there must be a limitation of fail connections in a row and a limitation of fail connections for the particular password. Also we should to insert a limitation on total connections number: successful and unsuccessful connections. These limitations are very important part of the protocol.

Let us observe the influence of these limitations on the bound of the theorem 3.11. Thus, for example, in a summand $\frac{(2q_{exec}+q_{send})^2}{q}$ the value q_{exec} is majorized with the admissible number of sessions for the protocol, and q_{send} is majorized with the admissible number of the fail connections for some password. The admissible number of the fail connections is usually set to several tens, and the admissible number of sessions is majorized with the value of order $10^7 - 10^9$. In this case, if $q \approx 2^{256}$, then

$$\frac{(2q_{exec} + q_{send})^2}{q} \leqslant (3 \cdot 10^9)^2/2^{256} \leqslant 2^{64}/2^{256} = 2^{-192}.$$

3.9.2 Processing of small order point case

In this section we consider the reasons for the special case processing when $\frac{m}{q}Q_B = 0_E$ (or $\frac{m}{q}Q_A = 0_E$) at generation of the key. In this case «special processing» uses the z_b flag and the point P as Q_B, if $\frac{m}{q}Q_B = 0_E$.

A protocol where this case is not processed in a special way can be exposed with the "online"-selection the passwords at wrong using the counters of fail connections number. We will now explain this thesis.

Let the counters of fail connections number are decreased at start of the protocol $SESPAKE_{KC}$, and a case $\frac{m}{q}Q_B = 0_E$ is not processed in the mentioned way. Then the adversary imitating A sends to B the points $u_1 = X - Q_{PW'}$, where $\frac{m}{q}X = 0_E$, for different PW' and measures a time that B needs to respond. After B responds the adversary disconnects. So, B does not decrease his counter of fail connections number, and measured time gives the adversary a criterion to find a right password — for wrong password the time will be essentially smaller, as the operation of scalar point multiplication (to obtain src) becomes degenerative.

Difference in times to check the condition $\frac{m}{q}Q_B = 0_E$ for cases $Q_B = 0_E$ and $Q_B \neq 0_E$ is inessential, as cofactor is sufficiently low on practice. It is explained by fact that a characteristic p of the base field is chosen for optimization as low as possible to ensure that the values of the parameter q meets the requirements ($2^{254} < q < 2^{256}$ or $2^{508} < q < 2^{512}$) set in the document [6]. Thus the values q and p become close, it means that a cofactor value m/q is low too.

The analysis method described above becomes unusable if participants decrease their counters at the start of the work.

Additionally notice that if the adversary imitates a side B he can manipulate a value UKM. But he can not set it to 0 to obtain the degeneracy of scalar point multiplication because A can detect it.

3.9.3 Pseudorandomness of the points Q_1, Q_2, Q_3 and P

The points Q_1, Q_2, Q_3 and P must be chosen in such a way that a multiplicity of any point under any other point would be unknown, and a run time for computing this multiplicity would be comparable with a run time for solving a discrete logarithm problem in the group of used elliptic curve. We can achieve it by choosing the points according to the principle of proved pseudorandomness [7]. So, a choice of elliptic curve points in the Weierstrass form can be made by the following way: generate a random string s and set $x = H(s)$ until the value $x^3 + ax + b$ becomes a quadratic residue; set $y = \sqrt{x} \mod p$. Here as H we can use, for example, the GOST R 34.11-2012 hash-function [5].

If the multiplicity of some point Q_i under P is known, then the adversary that imitates a server can get a criterion to control the wrong passwords (thus he can find a password in "offline" mode). In this case he needs to interact with a client just once. We describe the potential action sequence. For short we consider parameters UKM and $\frac{m}{q}$ equal to 1.

If the Q_{ind} multiplicity under P is known, then Q_s multiplicity under P is also known: let $Q_s = \gamma P$. Let $z = F(PW, salt, 2000)$. Then the adversary sends the message $u_2 = \beta P + z'\gamma P$ in response to the message $u_1 = \alpha P - z\gamma P$, where β and z' are values chosen by the adversary at will. The next message sent by a client is a constant string ciphered on the key $K = H_{256}\left(\alpha\left(\beta + z'\gamma - z\gamma\right)P\right)$ unknown by the adversary. Correctness of decoding this message is a criterion to control wrong session keys and the relation

$$K = H_{256}\left(\left(\beta + z'\gamma - z\gamma\right) \cdot \left(u_1 + z\gamma P\right)\right),$$

gives a possibility to try a key. The adversary does not know just a dependent on the password value z at the right.

If the Q_s multiplicity under P is unknown, then the adversary needs to know α, as the key will be computed by formula $K = H_{256}\left(\alpha\beta P + \alpha r' Q_s - \alpha r Q_s\right)$.

3.9.4 A secret parameter

From the description of the protocol we can easily see that a client authenticates using not PW but Q_{PW}. A server does not know that Q_{PW} is generated from the password PW.

But there is a sense to consider a modified protocol: the server doesn't store Q_{PW} but generates Q_{PW} from PW at the start of the authentication procedure. This modification has a sense if the participants are equal, i.e, for example, participants are the custom workstations. In this case a password is entered by users every time when they want to connect according to the protocol. But the modified protocol has some operational differences from the original one. Also this modification can influence on the properties of the used mechanisms upon some conditions.

Notice that if a server stores Q_{PW}, then he has no need to compute Q_{PW} at the work start. It can be excused in the case when the server is a device with the limited computing resources (e.g., if a server is a functional key carrier).

The keeping by the server a point Q_{PW} instead of PW is excused also if a password PW is used by the client not for interaction according to the considered protocol only. In this case if the adversary gets an access to server's secret parameter (password PW), then he knows the secret parameters of all systems that uses the same password PW. If the secret parameter is a point Q_{PW}, the adversary just gets a criterion to find PW.

The participant should protect stored parameters PW and Q_{PW} using technical means and organizational measures.

4 Conclusion

We propose a new password-authenticated key exchange protocol *SESPAKE* and analyze its cryptographic properties. The *SESPAKE* protocol includes a key agreement step and an authentication step without key diversification. For this protocol we propose to extend the original indistinguishability-based model. It allows to prove the protocol security under two different threats: the threat of distinguishing a generated session key from a random string and the threat of the false authentication.

This protocol is the first *PAKE* protocol without key diversification for a full version of which a full security proof has been obtained. Also the paper contains a comparative review of the some well-known protocols with the proposed one considering efficiency and security properties. Additionally there is a detailed explanation of the reasons of all protocol elements.

The SESPAKE protocol is one of the most efficient *PAKE* protocols as it needs only 4 computations of multiple points in the group of elliptic curve points and 2 group elements and 2 hash values exchanges.

5 Acknowledgements

The authors are very grateful to Vladimir Popov, Vasiliy Nikolaev, Lolita Sonina for their valuable discussions and suggestions concerning this work.

References

[1] J. Schmidt. Requirements for PAKE schemes, Internet-Draft, October 13, 2015. draft-irtf-cfrg-pake-reqs-01.

[2] M. Abdalla, D. Pointcheval. Simple Password-Based Encrypted Key Exchange Protocols. Proc. of Topics in Cryptology - CT-RSA 2005, LNCS 3376, pp. 191-208, Springer-Verlag.

[3] F. Bao, R.H. Deng, H. Zhu. Variations of Diffie-Hellman problem. In: S. Qing, D. Gollmann, J. Zhou, (eds.) ICICS 2003. LNCS, vol. 2836, pp. 301–312. Springer, Heidelberg (2003).

[4] B. Kaliski. RSA Laboratories. «PKCS 5: Password-Based Cryptography Specification Version 2.0», RFC 2898, September 2000.

[5] V. Dolmatov, , Ed., and A. Degtyarev. «GOST R 34.11-2012: Hash Function», RFC 6986, August 2013.

[6] V. Dolmatov, , Ed., and A. Degtyarev. «GOST R 34.10-2012: Digital Signature Algorithm», RFC 7091, December 2013.

[7] M. Lochter, J. Merkle, J-M. Schmidt, T. Schutze. Requirements for Standard Elliptic Curves. Cryptology ePrint Archive, Report 2014/832.

[8] S. Smyshlyaev, Ed., E. Alekseev, I. Oshkin, V. Popov CRYPTO-PRO. «The Security Evaluated Standardized Password Authenticated Key Exchange (SESPAKE) Protocol», Internet-Draft, November 29, 2015. draft-smyshlyaev-sespake-00.

[9] S. Smyshlyaev, Ed., E. Alekseev, I. Oshkin, V. Popov, S. Leontiev CRYPTO-PRO, V. Podobaev FACTOR-TS, D. Belyavsky TCI. «Guidelines on the cryptographic algorithms, accompanying the usage of standards GOST R 34.10-2012 and GOST R 34.11-2012», RFC draft, August 14, 2015.

[10] M. Abdalla. Reducing The Need For Trusted Parties In Cryptography. HAL Id: tel-00917187 https://tel.archives-ouvertes.fr/tel-00917187. Submitted on 11 Dec 2013

[11] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, Advances in Cryptology – EUROCRYPT 2000, volume 1807 of Lecture Notes in Computer Science, pages 139–155. Springer, May 2000.

[12] M. Bellare and P. Rogaway. The AuthA Protocol for Password-Based Authenticated Key Exchange. Contributions to IEEE P1363. March 2000.

[13] E. Bresson, O. Chevassut, D. Pointcheval. Security Proofs for an Efficient Password-Based Key Exchange. CCS '03 Proceedings of the 10th ACM conference on Computer and communications security.

[14] M. Bellare, A. Boldyreva, A. Desai, D. Pointcheval. Key-Privacy in Public-Key Encryption. In Asiacrypt '01, LNCS 2248, pages 566-582. Springer-Verlag, Berlin, 2001.

[15] V. Boyko, P. MacKenzie, S. Patel. Provably secure password authenticated and key exchange using Diffie-Hellman. In EUROCRYPT 2000 (LNCS 1807), pp. 156-171, 2000.

[16] P. MacKenzie. The PAK Suite: Protocols for password-authenticated key exchange, 2002.

[17] M. Bellare, R. Canetti, H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In 30th Annual ACM Symposium on Theory of Computing, pages 419–428. ACM Press, May 1998.

[18] R. Canetti, S. Halevi, J. Katz, Y. Lindell, P. D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, Advances in Cryptology – EUROCRYPT 2005, volume 3494 of Lecture Notes in Computer Science, pages 404–421. Springer, May 2005.

[19] S. Bellovin, M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. Proceedings of the I.E.E.E. Symposium on Research in Security and Privacy, Oakland, May 1992.

[20] M. Bellare and P. Rogaway. Entity Authentication and key distribution. Advances in Cryptology - Crypto 93 Proceedings, Lecture Notes in Computer Science Vol. 773, D. Stinson ed, Springer-Verlag, 1994.

[21] J. Bender, M. Fischlin, D. Kugler. Security Analysis of the PACE Key-Agreement Protocol, Proceedings of the 12th International Conference on Information Security, September 07-09, 2009, Pisa, Italy.

[22] D. Harkins. Simultaneous authentication of equals: A secure, passwordbased key exchange for mesh networks. In Sensor Technologies and Applications, 2008. SENSORCOMM '08. Second International Conference on, 839 – 844, aug. 2008.

[23] D. Clarke and F. Hao. Cryptanalysis of the Dragonfly key exchange protocol. IET Information Security, Volume 8, Issue 6, November 2014, 283 – 289.

[24] J. Lancrenon, M. Skrobot. On the Provable Security of the Dragonfly Protocol. Information Security, Volume 9290 of the series Lecture Notes in Computer Science, p. 244-261, 2015.

[25] M. Bellare. Practice-oriented provable-security, Proc. First International Workshop on Information Security (ISW '97), LNCS 1396, SpringerVerlag, 1998, 221-231.

[26] D. Jablon. Strong Password-Only Authenticated Key Exchange. ACM SIGCOMM Computer Communication Review, 1996.

[27] P. MacKenzie. On the security of the SPEKE Password-Authenticated Key Exchange protocol. IACR ePrint Archieve, 2001.

[28] Federal Office for Information Security (BSI): Advanced security mechanism for machine readable travel documents – extended access control (eac), password authenticated connection establishment (pace), and restricted identification (ri) (2008).

[29] F. Hao, S. F. Shahandashti. The SPEKE Protocol Revisited. Security Standardisation Research, 2014.

[30] S. Shin, K. Kobara. Most Efficient Augmented Password-Only Authentication and Key Exchange for IKEv2, 2008.

[31] S. Shin, K. Kobara, H. Imai. Security Proof of AugPAKE. Cryptology ePrint Archive: Report 2010/334, 2010.

[32] S. Shin, K. Kobara. Augmented Password-Authenticated Key Exchange (AugPAKE), the Internet-Draft, July 22, 2015. draft-irtf-cfrg-augpake-04.

[33] F. Hao, P. Ryan. Password Authenticated Key Exchange by Juggling. Proceedings of the 16th International Workshop on Security Protocols, 2008.

[34] M. Abdalla, P. MacKenzie, F. Benhamouda. Security of the J-PAKE Password-Authenticated Key Exchange Protocol, 2015.

6 Appendix

Lemma 6.1. *Let $\pi_{r,N}$ is the probability that there are two equal values among r realization of a random variable uniformly distributed on the set of power N. Then the following inequality holds:*

$$\pi_{r,N} \leqslant \frac{r^2}{2N}.$$

Proof. Let us denote the probability that all r realization are different as $\chi_{r,N}$. It is easy to see that $\pi_{r,N} = 1 - \chi_{r,N}$ and

$$\chi_{r,N} = 1 \cdot \left(1 - \frac{1}{N}\right)\left(1 - \frac{2}{N}\right) \ldots \left(1 - \frac{r-1}{N}\right).$$

We will prove by induction that $\chi_{r,N} \geqslant 1 - \frac{r(r-1)}{2N}$. For $r = 1$ statement is right as $\chi_{1,N} = 1 \geqslant 1$. Let the statement holds for $r \geqslant 1$. The justice of this statement for $r+1$ follows from the relations:

$$\chi_{r+1,N} = \chi_{r,N} \cdot \left(1 - \frac{r}{N}\right) \geqslant \left(1 - \frac{r(r-1)}{2N}\right)\left(1 - \frac{r}{N}\right) = 1 - \frac{1}{N}\left(\frac{r(r-1)}{2} + r\right) + \frac{r^2(r-1)}{2N^2} > 1 - \frac{r(r+1)}{2N}.$$

Thus we conclude that $\pi_{r,N} \leqslant \frac{r(r-1)}{2N} \leqslant \frac{r^2}{2N}$. □