NEW SPAKE FORMALISMS AND SECURITY PROOFS IN THE BELLARE-ROGAWAY GAME-PLAYING MODEL

Yeo Jie Xuan, Isaac[1], Tricia Chia Kee Ann[1], and Ruth Ng Ii-Yung[2]

[1] Raffles Institution (Junior College), 10 Bishan Street 21, Singapore 574013
[2] DSO National Laboratories, 12 Science Park Drive, Singapore 118225

Abstract. PAKE (Password-Authenticated Key Exchange) is a family of cryptographic schemes which allows users with low-entropy passwords to establish a high-entropy session key. This paper provides a deeper security analysis into the PAKE schemes by Abdalla and Pointcheval (2005), written in the Bellare-Rogaway Game-Playing Model. We convert the asymptotic syntax used in the original paper to a modern concrete one and provide an additional proof which formalizes a result left implicit in the original work. In particular, an insecure strawman scheme is presented in the paper and we provide new definitions and a precise reduction to accurately measure the advantage of any adversary against it. We follow this with a brief security analysis of their scheme SPAKE1, conjecturing that it is secure against all concrete adversaries. This enables website and application developers to evaluate and compare the security of different competing PAKE schemes when choosing one to implement.

1 Introduction

Nowadays, many applications rely on passwords for user authentication. In some use-cases, two users select a common password and must use it to establish high-entropy session key for end-to-end encryption. Usually, the password is a weak one that people can easily remember. This makes naïve attempts at session-key generation vulnerable to dictionary and other offline attacks by adversaries on the server or the network, compromising the entire session's data. PAKEs (Password-Authenticated Key Exchange) is a type of cryptographic scheme which solves this problem and seeks to establish strong session keys from weak passwords, thus allowing the communications to remain secure.

According to [5], PAKEs allow secure session keys to be exchanged over the network with the only prerequisite shared secret between the two parties involved being the (weak) password. Another defining property of PAKEs is that they also minimise the ability of an adversary, be it an eavesdropper or man-in-the-middle, to perform brute-force attacks on the communications between the two parties (like dictionary attacks) to obtain the password. PAKE thus provides strong security while making use of only low-entropy passwords, which is extremely useful in the real world.

To formally prove the security of such schemes, then, we employ the game-playing model, first introduced by Bellare and Rogaway in 2004 [5]. In recent years, this has become a popular approach to writing security proofs as this is a precise way of stating and concretising the success of an adversary, and thus this model will be used later in this paper. One way of concretely proving the security/insecurity of a scheme is by reducing the scheme to a known hard problem, which we will show later. Another use of the model demonstrated in section **??** is formally describing an adversary and measuring its advantage, i.e. how well it performs.

In the game-playing model, the user is the "gamemaster" and the adversary is the "player". The user sets up the scheme, which is the "game" that the adversary is playing against, and the adversary interacts with the user through oracles, which are the only ways to access game

state. The adversary wins the game if it is able to reach the adversarial goal (e.g. guessing the password, finding the key); the user wins otherwise.

1.1 Contributions

This paper will focus on fully analysing the SPAKE schemes provided in [1]. The main contributions of this paper are as follows:

- full analysis of the strawman scheme that the original paper glossed over - we gave a mathematical proof of its insecurity by constructing an adversary and precisely measuring its advantage in the game-playing model (in section 6.2)
- conversion of several results from the original paper from asymptotic approach to the more modern concrete approach - the latter is more widely-used currently due to the way it simplifies implementation and provides accurate security bounds to measure security/insecurity of a scheme (shown in section 7.1)

2 Syntax and definitions

As mentioned above, we adapt the PAKE security model of Abdalla and Pointcheval [1] to the concrete variant of the Bellare-Rogaway game-playing model [4]. In doing so, we borrow from the notation of [2].

2.1 PAKE syntax

We say P is a PAKE (Password-Authenticated Key Exchange) scheme if it satisfies the following properties for any user U:

$\mathsf{P.ks}$ is the set of all possible session keys exchanged.
$\mathsf{P.Exec}^*$ executes a protocol between two users, where each user provides their password as input and receives the secret key as output.

For correctness, the outputs that two users recieve should always be equal given that their inputs (i.e. the password given) are equal. Additionally, the output of $\mathsf{P.Exec}$ to each user must always be an element of $\mathsf{P.ks}$.

2.2 Notation for describing PAKE over the network

Here we define for some PAKE scheme P some notation to aid in the description of games and adversaries later on. We say that $\mathcal{U}$ is the set of all users, and for any $U_1, U_2 \in \mathcal{U}$:

$\mathsf{P.Send}(U_2, (U_1, M))$ denotes U_1 sending M to U_2;
$\mathsf{P.Exec}(U_1, U_2)$ denotes $\mathsf{P.Exec}^*$ between U_1 and U_2, where U_1 and U_2 both input their password, returning the session transcript; and
$\mathsf{P.sks}$ is a mapping between users and session keys. Whenever a new session key (which should be the same for honest execution) K is established for user U, $\mathsf{P.sks}[U] = K$.

2.3 Password frequency distributions

We define a password to be a bitstring whose length is at most L, where L is an integer which all practical passwords do not exceed in length. We then define a password frequency distribution function to be a function $f : \bigcup_{i=1}^{L} \{0,1\}^i \to [0,1]$, such that its sum over all possible passwords is exactly 1. We then define a function $\text{rank}(f, n)$ that takes as input a password frequency distribution f and an integer n, and returns the password with the n-th highest frequency (if there is more than one, they are sorted in lexicographical order). Finally, we define a algorithm $\text{draw}(f)$ that returns each possible password pw with probability $f(\text{pw})$ for any password frequency distribution f.

3 DLP dictionary problem (DLP-DICT)

A *dictionary attack* is a common attack against many cryptographic schemes that involves guessing common keys or plaintexts. It can be very effective when the key is low-entropy. PAKEs aim to defeat such attacks, but a weak PAKE scheme that we will be studying later lends itself quite handily to such an attack. As such we will now be looking at a game $\text{G}^{\mathsf{dlp\text{-}dict}}_{\mathbb{G},f,N}$ (where $\mathbb{G}$ is a group, N is a group element and f is a password distribution function that returns the password frequency) that specifically captures that situation, and one specific adversary who has a high advantage playing this game given certain conditions are fulfilled.

As seen in figure 1a, the adversary is given N^{pw} where pw is a password drawn randomly according to the distribution f. The adversary succeeds if the value of $N^{\text{pw}'}$ it obtained is equal to the actual one given by the game (N^{pw}), and it can use the oracle FREQ given to access the n-th most common password.

We also define the advantage of an adversary $\mathcal{D}$ against this game as the probability of it winning the game; that is,

$$\mathbf{Adv}^{\mathsf{dlp\text{-}dict}}_{\mathbb{G},f,N}(\mathcal{D}) = \Pr\left[\text{G}^{\mathsf{dlp\text{-}dict}}_{\mathbb{G},f,N}(\mathcal{D})\right].$$

Now, we will give a simple example of an adversary that performs a dictionary attack, i.e. it just tries passwords in order of frequency. This adversary can perform well in this game given f is mildly skewed i.e. given moderately high variance; in the lemma below we will state exactly its advantage in terms of f.

Game $\text{G}^{\mathsf{dlp\text{-}dict}}_{\mathbb{G},f,N}(\mathcal{D})$

$\text{pw} \leftarrow \text{draw}(f)$
$\text{pw}' \leftarrow \mathcal{D}^{\text{FREQ}}(N^{\text{pw}})$
return $N^{\text{pw}'} = N^{\text{pw}}$

Oracle $\text{FREQ}(n)$

return $\text{rank}(f, n)$

(a) Game for DLP-DICT problem

Adversary $\mathcal{D}^{\text{FREQ}}_{q}(N^{pw})$

$\text{pw}' \leftarrow 0, n \leftarrow 1$
while $N^{\text{pw}'} \neq N^{\text{pw}}$ **and** $n \leq q$,
  $\text{pw}' \leftarrow \text{FREQ}(n)$
  $n \leftarrow n + 1$
if $N^{\text{pw}'} \neq N^{\text{pw}}$ **and** $n > q$
  then $\text{pw}' \leftarrow \text{FREQ}(q + 1)$
return pw'

(b) An adversary against the DLP-DICT problem

Lemma 1. *We define a family of adversaries parameterised by q in figure 1b. Then*

$$\mathbf{Adv}^{\mathsf{dlp\text{-}dict}}_{\mathbb{G},f,N}(\mathcal{D}_q) = \sum_{i=1}^{q+1} f(\mathrm{rank}(f,i)).$$

Proof. Notice that if the selected pw is among the first $q+1$ highest-frequency passwords, the adversary is guaranteed to return that password: if it is among the first q, then it will be checked and returned in the **while**-loop, and if it is the $q+1$-th password, then it will just be returned without checking after the **while**-loop (as shown in the **if** branch). Since, by definition, the probability that the password is the first in that list is given by $\mathrm{rank}(f,1)$, the probability of it being second is given by $\mathrm{rank}(f,2)$, and so forth, and these events are all disjoint (i.e. mutually exclusive), the probability any of the events occurring is simply given by the sum of all their probabilities. □

4 The security of PAKE

We will now present a framework with which we can formally prove the security of PAKE schemes, as previously laid out by [2].

4.1 Intuition

Intuitively, we expect that a secure PAKE satisfies these four security properties:

Offline dictionary resistance Offline dictionary resistance is the property that even if communications between the two parties are leaked, it is computationally infeasible to recover the shared password, meaning that it is impractical to launch a brute-force dictionary attack on any leaked communications using current or near-future technology.

Forward secrecy Forward secrecy is achieved by a PAKE scheme when, in the event that the password is leaked, it is still computationally infeasible to recover previous session keys that were exchanged using it.

Known-session security The keys exchanged using a known-session secure PAKE scheme will appear to be completely unrelated; in other words, even if one session key is compromised, the attacker should not be able to compromise other keys given only that information.

Online dictionary secrecy A PAKE scheme achieves online dictionary secrecy if it only allows an attacker to check at most one password per execution of the protocol in a single session with a single user.

4.2 Security game

Now, we will formalise the above intuitive notions of security with a security "game". While the four security properties are convenient for checking that a scheme is secure, we will see later that simply fulfilling the above properties does not necessarily mean that the scheme is secure. Recalling [2], for some PAKE protocol P, and a table pws where $\mathsf{pws}[U] = \perp$ for all U initially, we have:

Game $\mathrm{G}^{\mathrm{pake}}_{\mathsf{P},f}(\mathcal{A})$

$\mathsf{revealed} \leftarrow \emptyset$
$\mathsf{flag} \leftarrow \mathsf{false}$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\textsc{Send},\textsc{Execute},\textsc{Reveal},\textsc{Test},\textsc{Freq}}(f)$
return $b' = b$

Oracle $\textsc{Execute}(U_1, U_2)$

if $\mathsf{pws}[U_1] = \perp$ **and** $\mathsf{pws}[U_2] = \perp$ **then**
  $\mathsf{pws}[U_1] = \mathrm{draw}(f)$
if $\mathsf{pws}[U_1] = \perp$ **then**
  $\mathsf{pws}[U_1] = \mathsf{pws}[U_2]$
if $\mathsf{pws}[U_2] = \perp$ **then**
  $\mathsf{pws}[U_2] = \mathsf{pws}[U_1]$
return $\mathsf{P.Exec}(U_1, U_2)$

Oracle $\textsc{Send}(U_2, (U_1, x))$

return $\mathsf{P.Send}(U_2, (U_1, x))$

Oracle $\textsc{Reveal}(U)$

$\mathsf{revealed} \leftarrow \mathsf{revealed} \cup \{U\}$
return $\mathsf{P.sks}[U]$

Oracle $\textsc{Test}(U)$

if flag **or** $U \in \mathsf{revealed}$ **then return** $\perp$
$\mathsf{revealed} \leftarrow \mathsf{revealed} \cup \{U\}$
$\mathsf{flag} \leftarrow \mathsf{true}$
let $K_0 \xleftarrow{\$} \mathsf{P.ks}$
let $K_1 = \mathsf{P.sks}[U]$
return K_b

Oracle $\textsc{Freq}(n)$

return $\mathrm{rank}(f, n)$

Intuitively, the goal of the adversary is to guess whether a given bitstring is a real session key or not. The adversary wins if and only if they can successfully tell whether a key is real, as shown in the game, and the success of the adversary would prove the scheme's insecurity due to its inability to select a truly random session key. To this end, they will be given some oracles as shown above, which they may use to interact with the game any number of times. The oracle $\textsc{Execute}(U_1, U_2)$ establishes a common password between 2 users according to a password frequency distribution, $\textsc{Send}(U_2, (U_1, x))$ sends a message from a user (denoted as U_1) to another user (denoted as U_2), $\textsc{Reveal}(U)$ denotes revealing a session key, $\textsc{Test}(U)$ returns either the real or random session key for the adversary to guess, and $\textsc{Freq}(n)$ returns the password with the n-th highest frequency in the distribution.

Now, let us formally define $\Pr[\mathrm{G}^{\mathrm{pake}}_{\mathsf{P},f}(\mathcal{A})]$ to be the probability that our adversary A wins the game. Then the advantage of A is

$$\mathbf{Adv}^{\mathrm{pake}}_{\mathsf{P},f}(\mathcal{A}) = 2\Pr\left[\mathrm{G}^{\mathrm{pake}}_{\mathsf{P},f}(\mathcal{A})\right] - 1$$

This formula accounts for the 50% probability that A wins by guessing the key randomly, and scales it to be between 0 and 1. Under the above assumptions, we say that the PAKE scheme is secure if the advantage of any practical adversary is negligible and it is extremely difficult for the adversary to guess whether the key is real.

5 SPAKE protocols

Here, we present two similar protocols which we will be studying in the diagram below, both of which are taken from [1]. The strawman scheme is not named in the original work, so for our purposes we will be naming it SPAKE?. The diagram illustrates parts of the protocols, SPAKE? and SPAKE1, that are exactly the same, and the differences between them are then shown in different types of boxes below that.

In each SPAKE protocol, Alice first picks a random number x from $\mathbb{N}$ and computes X, which is defined to be $g^x M^{\mathrm{pw}}$. Then Bob does the same, except using another random

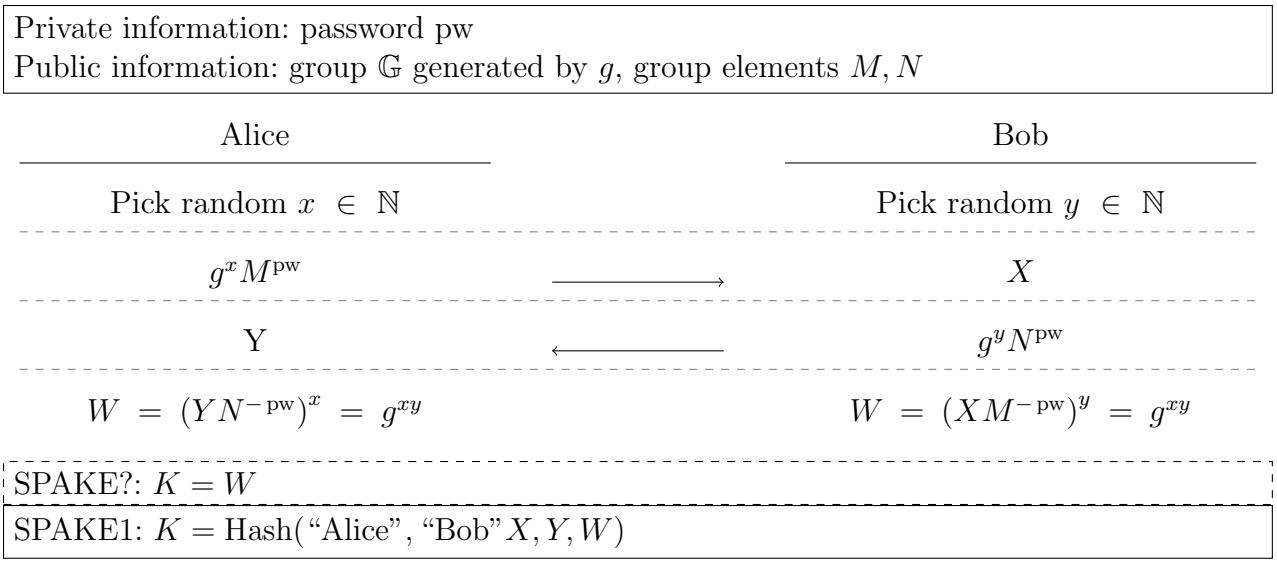

Fig. 2: Illustration of the various SPAKE protocols

number y. At the end, both Alice and Bob compute W, from which K is directly derived as shown above. We will later be bounding the advantage of any adversary against SPAKE? to show the scheme's insecurity and will also be providing a conjecture and intuition for the security of SPAKE1.

6 SPAKE?

At first glance, SPAKE? is similar to the Diffie-Hellman Key Exchange scheme and seems to fulfil the four security properties, and hence should be secure; intuitively it achieves all four security properties (since DHKE does as well). However, we soon realise the scheme is still not secure, even after satisfying the four security properties. We will now show why this is so and describe an adversary playing the SPAKE? game, again with reference to [1], in which SPAKE? was less formally referred to as an extreme case of a protocol with no use of random oracles. We shall, however, formally describe the adversary in more detail than the original paper for a better understanding of such a scheme's problems.

6.1 Definition

The two main functions of SPAKE? that must be implemented are SPAKE?.Exec and SPAKE?.Send. In Exec, the function simply executes the protocol between two simulated users as in figure 2. SPAKE?.Send is a state machine simulating only one user responding to the inputs given to it, according to the same protocol. For brevity, we do not list the pseudocode, here, but it can be found in appendix A.

6.2 (In)security of SPAKE? Scheme

This scheme looks quite similar to the Diffie-Hellman Key Exchange algorithm, which is thought to be very secure. However, as we shall soon see, there exists an adversary with high advantage against this game. Formally, we say that:

Theorem 1. *Given a group $\mathbb{G}$ with a public group element N, and a password frequency f, if there exists an adversary $\mathcal{B}$ against* DLP-DICT*, then there exists an adversary $\mathcal{A}$ against* SPAKE?*.*

$$\mathbf{Adv}^{\text{pake}}_{\mathsf{SPAKE?},f,N}(\mathcal{A}) \geq \left(1 - |\mathbb{G}|^{-1}\right)\mathbf{Adv}^{\mathsf{dlp\text{-}dict}}_{\mathbb{G},f,N}(\mathcal{B}).$$

Proof. Let e be the identity element of group $\mathbb{G}$. We will now provide a concrete example of such an adversary $\mathcal{A}$ built using $\mathcal{B}$ in figure 3. Here, the adversary works as a man-in-the-middle, by intercepting the first message from user 1, and altering it en route to user 2. Then, the adversary compromises K_1 and K_2, the session keys of U_1 and U_2. Thus the adversary is able to compute $N^{\text{pw}} = Y(K_1/K_2)r^{-1}$. Then, it will use $\mathcal{B}$ to attempt to brute-force pw from N^{pw} and use it to establish a new session key with a user. Then it will call TEST on that session, check whether its response is equal to the new session key it established, and return the corresponding bit (0 for not equal and 1 for equal).

For explanatory purposes, we will present the entire SPAKE? game with its oracles and the adversary inlined in 4. This provides the big picture of the adversary's attempt to reach its goal and break the scheme's security.

Recall the definition of the advantage of an adversary against the PAKE from section 4.2. This gives us

$$\begin{aligned}
\mathbf{Adv}^{\text{pake}}_{\mathsf{SPAKE?},f,N}(\mathcal{A}) &= 2\Pr\left[\mathrm{G}^{\text{pake}}_{\mathsf{SPAKE?},f,N}(\mathcal{A})\right] - 1 && (1)\\
&= 2\Pr[\mathrm{G}_0] - 1 && (2)\\
&= 2\Pr[b' = b] - 1 && (3)\\
&= 2(\Pr[b = 1]\Pr[b' = 1|b = 1] + \Pr[b = 1]\Pr[b' = 0|b = 0]) - 1 && (4)\\
&= 2\left(\frac{1}{2}\Pr[b' = 1|b = 1] + \frac{1}{2}\Pr[b' = 0|b = 0]\right) - 1 && (5)\\
&= \Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0] && (6)
\end{aligned}$$

The reason equations 2 and 3 are equal is that G_0 returns 1 if (and only if) $b' = b$ as seen in the last line of G_0. Then, manipulating probabilities reduces the expression to the difference between the probabilities of two simple cases in equation 6: when the adversary guesses 1 when challenge bit b is 1 and, and when the adversary guesses 1 when the challenge bit is 0. We will now explore these two cases.

Let us first consider the case where $b = 1$, i.e. the session key returned by TEST is real. Then the success of $\mathcal{A}$ depends entirely on the success of $\mathcal{B}$. If $\mathcal{B}$ succeeds, then $\mathcal{A}$ is guaranteed to succeed as well, and if $\mathcal{A}$ does not succeed, then it implies $\mathcal{B}$ did not succeed either. Note that $\mathcal{B}$ succeeds if it can find either pw or a value such that $N^{\text{pw}} \equiv N^{\text{pw}'}$, which it can use to establish a new session key; later on, testing if $K = K'$ will definitely return true. However, if $\mathcal{B}$ fails, then b' is guaranteed to be 0. Therefore,

$$\Pr[b' = 1|b = 1] = \Pr\left[\mathrm{G}^{\mathsf{dlp\text{-}dict}}_{\mathbb{G},f,N}(\mathcal{B})\right] = \mathbf{Adv}^{\mathsf{dlp\text{-}dict}}_{\mathbb{G},f,N}(\mathcal{B}).$$

Now we turn our attention to the case where $b = 0$, i.e. the session key returned by TEST is randomly generated. As explained above, if $\mathcal{B}$ fails, b' is immediately set to 0. If $\mathcal{B}$ succeeds, then one would expect $\mathcal{A}$ to return $b' = 0$. However, if the random group element K_0 chosen

in the line $K_0 \xleftarrow{\$} \mathbb{G}$ happens to be exactly the same as K', then the test $K' = K$ will still succeed, resulting in b' being set to 1. Since there are $|\mathbb{G}|$ group elements being chosen from, the probability of this collision is simply $|\mathbb{G}|^{-1}$. Then the overall probability of b' being set to 1 while b is 0 is given by the probability that $\mathcal{B}$ succeeds *and* $K' = K$. This yields

$$\Pr[b' = 1 | b = 0] = \Pr\left[\mathrm{G}^{\mathsf{dlp\text{-}dict}}_{\mathbb{G},f,N}(\mathcal{B})\right] \cdot |\mathbb{G}|^{-1} = |\mathbb{G}|^{-1}\mathbf{Adv}^{\mathsf{dlp\text{-}dict}}_{\mathbb{G},f,N}(\mathcal{B}).$$

□

Adversary $\mathcal{A}^{\textsc{Send},\textsc{Execute},\textsc{Reveal},\textsc{Test},\textsc{Freq}}$

$U_1 \xleftarrow{\$} \mathcal{U}$
$U_2 \xleftarrow{\$} \mathcal{U} \setminus \{U_1\}$
$r \xleftarrow{\$} \mathbb{G} \setminus \{e\}$
$\textsc{Execute}(U_1, U_2)$
$(U', X) \leftarrow \textsc{Send}(U_1, (U_2, \mathbf{start}))$
$(U', Y) \leftarrow \textsc{Send}(U_2, (U_1, Xg^r))$
$\textsc{Send}(U_1, (U_2, Y))$
$K_1 \leftarrow \textsc{Reveal}(U_1)$
$K_2 \leftarrow \textsc{Reveal}(U_2)$
$N^{\mathrm{pw}} \leftarrow K_1 K_2^{-1} r^{-1} Y$
$\mathrm{pw} \leftarrow \mathcal{B}^{\textsc{Freq}}(N^{\mathrm{pw}})$
if $\mathrm{pw} = \perp$ **then**
  return 0
$(U', Y) \leftarrow \textsc{Send}(U_1, (U_2, \mathbf{start}))$
$x \xleftarrow{\$} \mathbb{G} \setminus \{e\}; X \leftarrow g^x M^{\mathrm{pw}}$
$K \leftarrow (Y N^{-\mathrm{pw}})^x$
$K' \leftarrow \textsc{Test}(U_2)$
return $K = K'$

Fig. 3: Adversary against SPAKE?

Game G_0

$\mathsf{revealed} \leftarrow \emptyset$
$\mathsf{pws} \leftarrow \emptyset$
$\mathsf{flag} \leftarrow \mathsf{false}$
$b \xleftarrow{\$} \{0, 1\}$
$U_1 \xleftarrow{\$} \mathcal{U}; U_2 \xleftarrow{\$} \mathcal{U} \setminus \{U_1\}$
$r \xleftarrow{\$} \mathbb{G} \setminus \{e\}$
$\textsc{Execute}(U_1, U_2)$
$(U', X) \leftarrow \textsc{Send}(U_1, (U_2, \mathbf{start}))$
$(U', Y) \leftarrow \textsc{Send}(U_2, (U_1, Xg^r))$
$\textsc{Send}(U_1, (U_2, Y))$
$K_1 \leftarrow \textsc{Reveal}(U_1); K_2 \leftarrow \textsc{Reveal}(U_2)$
$N^{\mathrm{pw}} \leftarrow K_1 K_2^{-1} r^{-1} Y$
$\mathrm{pw} \leftarrow \mathcal{B}^{\textsc{Freq}}(N^{\mathrm{pw}})$
if $\mathrm{pw} = \perp$ **then**
  $b' \leftarrow 0$
else
  $(U', Y) \leftarrow \textsc{Send}(U_1, (U_2, \mathbf{start}))$
  $x \xleftarrow{\$} \mathbb{G} \setminus \{e\}; X \leftarrow g^x M^{\mathrm{pw}}$
  $K' \leftarrow (Y N^{-\mathrm{pw}})^x$
  if flag **or** $U_2 \in \mathsf{revealed}$ **then**
    return $\perp$
  $\mathsf{revealed} \leftarrow \mathsf{revealed} \cup \{U_2\}$
  $\mathsf{flag} \leftarrow \mathsf{true}$
  $K_0 \xleftarrow{\$} \mathbb{G}$
  $K_1 \leftarrow \mathsf{SPAKE?.sks}[U_2]$
  $K \leftarrow K_b$
  $b' \leftarrow K' = K$
return $b' = b$

Fig. 4: Inlined SPAKE? game

We note that this man-in-the-middle adversary for SPAKE? poses a risk to practical applications, since a real-world adversary could do this by having either user install malware

on their computer. Using our result from 3, a more concrete lower bound on the advantage of such an adversary can be given:

Corollary 1. *Given a group $\mathbb{G}$ with a public group element N where exponentiation can be performed in $O(\tau)$, and a password frequency f, there exists a $q\tau$-time adversary $\mathcal{A}$ against* SPAKE? *such that*

$$\mathbf{Adv}^{\text{pake}}_{\mathsf{SPAKE?},f}(\mathcal{A}) \geq \left(1 - |\mathbb{G}|^{-1}\right) \sum_{i=1}^{q+1} f(\text{rank}(f, i)).$$

Proof. If we substitute $\mathcal{B}$ in our adversary above for $\mathcal{D}_q$ for some positive integer q, (defined in section 3), then lemma 1 gives the advantage of $\mathcal{D}_q$ which can just be plugged in.

Since the only non-constant-time component of $\mathcal{A}$ is the line where it runs $\mathcal{B}$ — instantiated in this case to $\mathcal{D}_q$ — it is guaranteed to make up to q exponentiations in $\mathbb{G}$. Therefore this adversary has a time complexity of $O(q\tau)$ where τ is the time complexity of an exponentiation in $\mathbb{G}$. □

7 SPAKE1

For this section, we will be providing an intuition and conjecture for the security of SPAKE1.

7.1 Security

Since it is very similar to SPAKE?, the formal definition of SPAKE1 can be found in appendix B. Notice that the only difference between this scheme and SPAKE? is the final hashing of the session key (see figure 2 above). Intuitively, the reason this prevents the attack against SPAKE? is because if we assume the hash is "good", then it destroys all structure, crucially including the group structure. This means that the adversary cannot exploit algebraic relationships between session keys; no matter how he tampers with communications, they will not be related in any predictable way.

What exactly is a "good" hash? [7] lists some common properties of a strong hash function such as preimage resistance and collision resistance. However, the ultimate ideal hash is given by a so-called *random oracle*. First formally stated in [3], this model assumes the existence of a function that associates each unique input with a truly random output, and hence there is no way to predict its output, or relationships between outputs even if the inputs are closely related. So, the attack is mitigated by using a hash function behaving like a random oracle.

However, the hashing of the session key does not only prevent this attack, but it also prevents *all other attacks*. By *reducing* the problem down to known hard problems, it can be shown that breaking SPAKE1 is itself a hard problem.

A theorem bounding SPAKE1's security was first stated and proven in [1]. Here we have adapted it in the concrete style rather than the asymptotic approach used in the paper, and as such certain bounds had to be adapted and certain variables had to be concretised from the original, necessitating re-proving of the theorem; we are however fairly certain of this new bound. Therefore, it is instead given here as a conjecture whose proof is an area for future work to expand on.

Conjecture 1. Given any adversary $\mathcal{A}$ against SPAKE1 which makes at most q_{send}SEND queries (of which q_{start} send the message **start**), q_H hash queries, and q_{exe} EXECUTE queries, there exist adversaries $\mathcal{B}$ and $\mathcal{C}$ against CDH over some group $\mathbb{G}$ respectively such that $\mathcal{B}$ runs in $O(4t + (q_{\text{start}} + q_H)\tau)$ and $\mathcal{C}$ runs in $O(t + (2q_{\text{exe}} + 3)\tau)$ where τ is the time complexity of a single exponentiation in $\mathbb{G}$, and

$$\mathbf{Adv}^{\mathsf{pake}}_{\mathsf{SPAKE1},f,N}(\mathcal{A}) \leq 2\left(\frac{q_{\text{send}}}{2^L} + \sqrt[6]{2^{14-2L}\mathbf{Adv}^{\mathsf{cdh}}_{\mathbb{G}}(\mathcal{B}) + \frac{2^{15}q_H^4}{2^{2L}|\mathbb{G}|}}\right) + 2\left(\frac{(q_{\text{exe}} + q_{\text{send}})^2}{2|\mathbb{G}|} + q_H\mathbf{Adv}^{\mathsf{cdh}}_{\mathbb{G}}(C)\right).$$

The CDH problem, its game, and its advantage function are quite well-known; but for completeness they can be found in appendix C. Therefore, given that CDH is a well-known hard problem, an adversary against SPAKE1 that is bounded by an adversary against CDH has a low advantage, making the scheme secure.

8 Conclusion

In this paper, we have precisely measured the advantage of any adversary against SPAKE?, a strawman scheme provided in [1], using the Bellare-Rogaway game-playing model. We have also written everything in the concrete approach, which better aids implementation and accuracy. The results of this paper would be useful for people who want to implement PAKE schemes (like application and website designers), and future researchers of PAKE, to better evalulate and compare the security of competing PAKE schemes and make a more informed decision when choosing a scheme to implement.

8.1 Future work

Future work could explore the full security proof of the SPAKE1 theorem (conjecture 1), study the issue of session concurrency for SPAKE1, and also prove the security of SPAKE2, the third scheme provided in [1]. A comparison of the practical usage of SPAKE1 and SPAKE2 could then be made.

Acknowledgements

We would like to thank our mentor, Dr. Ruth Ng, for taking a lot of time off her busy schedule to guide us in our project for the year. We would also like to thank Choo Jia Guang and Ti Yan Bo for helping us out in the project when we encountered issues along the way, and DSO for providing this opportunity and making this project possible. Finally, we would like to thank the SP20 interns for their emotional support and friendship during our internship.

References

[1] Michel Abdalla and David Pointcheval. "Simple Password-Based Encrypted Key Exchange Protocols". In: *Topics in Cryptology – CT-RSA 2005*. Ed. by Alfred Menezes. Vol. 3376. Lecture Notes in Computer Science. San Francisco, CA, USA: Springer, Heidelberg, Germany, 2005, pp. 191–208. DOI: 10.1007/978-3-540-30574-3_14.

[2] Mihir Bellare, David Pointcheval, and Phillip Rogaway. *Authenticated Key Exchange Secure Against Dictionary Attacks*. Cryptology ePrint Archive, Report 2000/014. https://eprint.iacr.org/2000/014. 2000.

[3] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *ACM CCS 93: 1st Conference on Computer and Communications Security*. Ed. by Dorothy E. Denning et al. Fairfax, Virginia, USA: ACM Press, 1993, pp. 62–73. DOI: 10.1145/168588.168596.

[4] Mihir Bellare and Phillip Rogaway. "The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs". In: *Advances in Cryptology – EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. Lecture Notes in Computer Science. St. Petersburg, Russia: Springer, Heidelberg, Germany, 2006, pp. 409–426. DOI: 10.1007/11761679_25.

[5] Steven M. Bellovin and Michael Merritt. "Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks". In: *1992 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 1992, pp. 72–84. DOI: 10.1109/RISP.1992.213269.

[6] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[7] Phillip Rogaway and Thomas Shrimpton. "Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance". In: *Fast Software Encryption – FSE 2004*. Ed. by Bimal K. Roy and Willi Meier. Vol. 3017. Lecture Notes in Computer Science. New Delhi, India: Springer, Heidelberg, Germany, 2004, pp. 371–388. DOI: 10.1007/978-3-540-25937-4_24.

A Formal definition of SPAKE?

Here a formal definition of SPAKE? in pseudocode is given as described in 6.1. Below we defined Env to be a map where $\mathsf{Env}[U] = \perp$ for all U initially.

Procedure SPAKE?.Send$(U_2, (U_1, \mathsf{msg}))$
if $\mathsf{msg} = \mathsf{start}$ **then**
  $x \xleftarrow{\$} \mathbb{G}; X \leftarrow g^x M^{\mathsf{pw}}$
  $\mathsf{Env}[U_1] \leftarrow \perp$
  $\mathsf{Env}[U_2] \leftarrow (x, X)$
  return (U_2, X)
if $\mathsf{Env}[U_2] = \perp$ **then**
  $(A, X) \leftarrow \mathsf{msg}$
  $y \xleftarrow{\$} \mathbb{G}; Y \leftarrow g^y N^{\mathsf{pw}}$
  $W \leftarrow (X/M^{\mathsf{pw}})^y$
  $\mathsf{Env}[U_2] \leftarrow (y, Y)$
  $K \leftarrow W$
  $\mathsf{SPAKE?.sks}[U_2] \leftarrow K$
  return (U_2, Y)
if $\mathsf{Env}[U_2] \neq \perp$ **and** $\mathsf{Env}[U_1] \neq \perp$ **then**
  $(B, Y) \leftarrow \mathsf{msg}$
  $x, X \leftarrow \mathsf{Env}[U_2]$
  $W \leftarrow (Y/M^{\mathsf{pw}})^x$
  $K \leftarrow W$
  $\mathsf{SPAKE?.sks}[U_2] \leftarrow K$

Procedure SPAKE?.Exec(U_1, U_2)
$x \xleftarrow{\$} \mathbb{G}; X \leftarrow g^x M^{\mathsf{pw}}$
$y \xleftarrow{\$} \mathbb{G}; Y \leftarrow g^y N^{\mathsf{pw}}$
$W \leftarrow (X/M^{\mathsf{pw}})^y$
$K \leftarrow W$
$\mathsf{SPAKE?.sks}[U_1] \leftarrow K$
$\mathsf{SPAKE?.sks}[U_2] \leftarrow K$
return $((U_1, X), (U_2, Y))$

B Formal definition of SPAKE1

Below we give a formal description of SPAKE1 as promised in 7.1.

Let Env be a map.
Procedure SPAKE1.Send$(U_2, (U_1, \mathsf{msg}))$
if $\mathsf{msg} = \mathsf{start}$ **then**
  $x \xleftarrow{\$} \mathbb{G}; X \leftarrow g^x M^{\mathsf{pw}}$
  $\mathsf{Env}[U_1] \leftarrow \perp$
  $\mathsf{Env}[U_2] \leftarrow (x, X)$

return (U_2, X)
if $\mathsf{Env}[U_2] = \perp$ **then**
$(A, X) \leftarrow \mathsf{msg}$
$y \xleftarrow{\$} \mathbb{G}; Y \leftarrow g^y N^{\mathsf{pw}}$
$W \leftarrow (X/M^{\mathsf{pw}})^y$
$\mathsf{Env}[U_2] \leftarrow (y, Y)$
$K \leftarrow H(A, U_2, X, Y, W)$
$\mathsf{SPAKE1.sks}[U_2] \leftarrow K$
return (U_2, Y)
if $\mathsf{Env}[U_2] \neq \perp$ **and** $\mathsf{Env}[U_1] \neq \perp$ **then**
$(B, Y) \leftarrow \mathsf{msg}$
$x, X \leftarrow \mathsf{Env}[U_2]$
$W \leftarrow (Y/M^{\mathsf{pw}})^x$
$K \leftarrow H(U_2, B, X, Y, W)$
$\mathsf{SPAKE1.sks}[U_2] \leftarrow K$

Procedure $\mathsf{SPAKE1.Exec}(U_1, U_2)$

$x \xleftarrow{\$} \mathbb{G}; X \leftarrow g^x M^{\mathsf{pw}}$
$y \xleftarrow{\$} \mathbb{G}; Y \leftarrow g^y N^{\mathsf{pw}}$
$W \leftarrow (X/M^{\mathsf{pw}})^y$
$K \leftarrow H(U_1, U_2, X, Y, W)$
$\mathsf{SPAKE1.sks}[U_1] \leftarrow K$
$\mathsf{SPAKE1.sks}[U_2] \leftarrow K$
return $((U_1, X), (U_2, Y))$

C Computational Diffie-Hellman problem (CDH)

Here we will reproduce the original Computational Diffie-Hellman problem as presented in [6], but generalised to any finite cyclic group $\mathbb{G}$ generated by g, as used in section **??**. Note that all operations below are taken in this group.

Given a pair of values (g^u, g^v) where u and v are uniformly randomly chosen from $\mathbb{N}$, find g^{uv}. We also say $\mathsf{CDH}(u, v) = g^{uv}$. The assumption that this problem is hard to solve was first used in the in original DHKE protocol in [6]; we say the advantage of an adversary $\mathcal{A}$ solving this problem $\mathbf{Adv}^{\mathsf{cdh}}_{\mathbb{G}}(\mathcal{A}) = \Pr\left[\mathrm{G}^{\mathsf{cdh}}_{\mathbb{G}}(\mathcal{A})\right]$, where $\mathrm{G}^{\mathsf{cdh}}_{\mathbb{G}}$ is defined as

Game $\mathrm{G}^{\mathsf{cdh}}_{\mathbb{G}}(\mathcal{A})$

$u \xleftarrow{\$} \mathbb{N}$
$v \xleftarrow{\$} \mathbb{N}$
$x \leftarrow \mathcal{A}(g^u, g^v)$
return $x = g^{uv}$