International Journal of Wireless and Mobile Computing.
Special Issue on Security of Computer Network and Mobile Systems.
Volume 2, Number 1, pages 4–13, Inderscience, 2007.

A Security Solution for IEEE 802.11's Ad-hoc Mode: Password-Authentication and Group-Diffie-Hellman Key Exchange

Emmanuel Bresson[1], Olivier Chevassut[2], and David Pointcheval[3]

[1] Cryptology Department, CELAR, 35174 Bruz, France,
http://www.di.ens.fr/~bresson, Emmanuel.Bresson@polytechnique.org.
[2] Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA,
http://www.itg.lbl.gov/~chevassu, OChevassut@lbl.gov.
[3] École normale supérieure, 75230 Paris Cedex 05, France,
http://www.di.ens.fr/~pointche, David.Pointcheval@ens.fr.

Abstract. The IEEE 802 standards ease the deployment of networking infrastructures and enable employers to access corporate networks while traveling. These standards provide two modes of communication called infrastructure and ad-hoc modes. A security solution for the IEEE 802.11's infrastructure mode took several years to reach maturity and firmware are still been upgraded, yet a solution for the ad-hoc mode needs to be specified. The present paper is a first attempt in this direction. It leverages the latest developments in the area of password-based authentication and (group) Diffie-Hellman key exchange to develop a *provably-secure* key-exchange protocol for IEEE 802.11's ad-hoc mode. The protocol allows users to securely join and leave the wireless group at time, accommodates either a single-shared password or pairwise-shared passwords among the group members, or at least with a central server; achieves security against dictionary attacks in the ideal-hash model (i.e. random-oracles). This is, to the best of our knowledge, the first such protocol to appear in the cryptographic literature.

Keywords: Password-based authentication, Group key exchange, Diffie-Hellman, Provable security.

1 Introduction

Wireless technology enables us to use our laptops on the couch at home or in hotel rooms, and gives us flexibility in where and when we work in the business environments. This technology makes it very easy to connect devices [34]. We only have to insert a wireless card into our laptop to establish radio link communications with fixed access points through which we talk to other devices and access the Internet. In this *infrastructure mode*, our laptop joins the network by discovering a wireless access point and negotiating with it the necessary temporal keys. The IEEE 802.11 working group defined the mechanisms for negotiating these keys via the Wi-Fi Protected Access (WPA) and the 802.11i standards [34, 35]. A simpler communication infrastructure for users that do not need broadband connectivity is to transmit data by means of the devices themselves. This networking infrastructure allows rapid developments and minimizes costs since wireless access points do not need to be deployed.

Wireless devices have the ability to operate an *ad-hoc mode* as specified by the IEEE 802.11 standards; however, the Wi-Fi Protected Access (WPA) protocol does not currently provide a security solution for it. The first solution that comes to mind is to maintain a group key for each sending device and to distribute the key to all the other devices using pairwise keys established by WPA. This solution, however, becomes impractical for groups of more than ten devices as the number of keys grows exponentially in the number of devices. Another solution is to develop a *provably-secure* group-Diffie-Hellman protocol for the IEEE 802.11i standard. As wireless technology

matures and bugs are fixed, the *infrastructure* and *ad-hoc modes* will complement each other to bring the Internet where broadband communication infrastructures do not currently exist [25, 28].

Wireless networks provide security researchers with the opportunity to develop *provably-secure* cryptographic technologies that will play an essential role in the deployment of broadband communication infrastructures. The present paper is a first attempt in this direction. It develops a *provably-secure* password-based group-Diffie-Hellman key-exchange protocol for IEEE 802.11's ad-hoc mode, by extending the work of Bresson et al. in three ways [15].

Its first contribution is to accommodate a single-shared password or pairwise-shared passwords among the group members or at least with a central server; unlike Bresson et al. who only allow for a single-shared password. Passwords are indeed frequently shared between users and taken advantage of to exchange a group session key. For example, a Bluetooth piconet is set-up once the devices have exchanged a group session via a two-party Diffie-Hellman key exchange protocol whose flows are encrypted using pairwise-shared passwords [12]. A piconet is also limited to eight devices.

Our second contribution in this paper is efficiency by allowing users to securely join and leave the wireless group at any time —the so-called *dynamic case*—. The dynamic group Diffie-Hellman key exchange using public-key cryptography was dealt with in [17], yet the password-based group Diffie-Hellman key exchange was not dealt with in the literature. We have revisited the scenario of Bresson et al. to enable users devices to join and leave the group as they move from one wireless domain to the next. Providing this "dynamic" feature in a secure way is not easy, but of primary importance to wireless networks. Consequently, the password-based protocol needs to dynamically update the group session key so that entering and leaving users do not gain access to previously exchanged messages. The group session key can not also be static but needs to be refreshed at regular intervals to prevent cryptanalysis [10]. Dynamicity in the membership and single-shared password among the group members seem two counter-intuitive notions at first. How can a user be forced to leave when everyone in the group shares the same password? When a user leaves the remaining group members exchange a new password by —as in the case of conference meeting— writing it on the board or —as in the case of home networking— keying the new password in each of the users' devices. The latter scenario is made more practical using pairwise-shared passwords among the group members.

Our third contribution is a more meaningful security result since security against dictionary attacks is achieved in the ideal-hash model (i.e. random oracles); unlike Bresson et al. who achieve it in both the ideal-cipher and ideal-hash models [15]. We have leveraged these researchers' formalization, wherein group members are modelled as oracles and the attacks of the adversary through queries to these oracles, to reach this cryptographic result. The protocol is a one-mask *Group Open Diffie-Hellman Key Encrypted* (GOKE) —in the sense of [13, 23, 24]— since not all the flows of the original group Diffie-Hellman key exchange are encrypted but only the down-flow. The protocol indeed minimizes the use of the "encryption" function (i.e. a mask-generation function), and is provably-secure in the ideal-hash model and under the standard Computational Diffie-Hellman assumption (CDH). The ideal-hash assumption is easier for engineers to implement than the ideal-cipher model since engineers just have to replace it with a straightforward construction from SHA-1 [5]. The end result is a

secure password-based authenticated group Diffie-Hellman protocol well-suited to the IEEE 802.11i standard.

Organization of the paper. Our paper is organized as follows. In the remainder of this section we summarize the related work. In Section 2, we recall the formalization proposed by Bresson et al. to model security against dictionary attacks in the group setting, and defer the reader to their paper [15] for further details on this model. In Section 3, we present the intractability assumptions upon which the security of the protocol is based. In Section 4, we describe the password-authenticated Diffie-Hellman Group Open Key Exchange (GOKE) protocol, while Section 5 is devoted to the security analysis. We finally conclude the paper.

1.1 Related Work

The cryptographic literature on designing secure protocols for password-authenticated key exchange is quite voluminous. Protocols for two-party Diffie-Hellman key exchange [18] have been proposed and refined for over a decade. The seminal work in this area is the *Encrypted Key Exchange* (EKE) protocol proposed by Bellovin and Merritt in the early 90's [7, 8]. Security researchers, however, were only recently able to come up with formal arguments to support the security of the complete suite of *EKE* protocols [4, 1, 2, 11, 13, 14, 23, 24]. Instantiations for the encryption primitive are either password-keyed symmetric ciphers [2, 6, 14] or mask-generation functions computed as the product of the message with the hash of a password [1, 11, 13, 23, 24]. Recently, Abdalla et al. proposed an original mask-generation function computed as the product of the message with a constant value raised to the power of the password [4]; this new mask generation function alleviates the need of a full-domain hash function in the group. Security researchers also provided constructions secured in the standard model based on general computational assumptions, the Decisional Diffie-Hellman assumption (using a variant of the Cramer-Shoup encryption scheme), or even strong computational assumptions; however, these construction are not efficient enough for practical use [20, 22]. Engineers are now given a suite of secure protocols to choose from depending on their security requirements (ideal-cipher model vs. ideal-hash model) and the constraints of their software (one-flow or two-flows of the Diffie-Hellman key exchange are encrypted).

In the light of these recent developments it was natural to provide engineers with a suite of secure password-authenticated group Diffie-Hellman protocols. Password-based protocols for the group setting have not been studied as extensively as the two-party case. Bresson et al. [15] adapted the formal model of Bellare et al. [2] and defined in it the execution of a password-authenticated group Diffie-Hellman protocol. The protocol is the original group Diffie-Hellman key exchange [33] with the flows randomized and encrypted using a symmetric cipher keyed with a single password shared among the group members [15]. The protocol, however, does not allow the parties to join and leave the group at any time which is a feature of prime importance to the IEEE 802.11 standards since users join and leave a group as they move from one wireless realm to another. Bresson et al.' protocol is secure against dictionary attacks in both the ideal-cipher and random-oracle models.

The present paper extends the work of Bresson et al. [15] to allow dynamicity in the membership, pairwise-shared passwords, and stronger security results.

2 The Model

Players. The players belongs to a nonempty set $\mathcal{U}$ of n users who can participate in the group Diffie-Hellman key exchange protocol P. A player $U_i \in \mathcal{U}$ may have many *instances* called oracles involved in distinct executions of P. The players also share a low-entropy secret pw taken from a small dictionary $\mathsf{Password}$ of size N. This password pw follows a certain distribution $\mathcal{D}_{pw}$ (uniform or not) in the $\mathsf{Password}$ set. The probability $\mathcal{D}_{pw}(q)$ to be in the most probable set of q passwords is denoted as follows:

$$\mathcal{D}_{pw}(q) = \max_{\substack{P \subseteq \mathsf{Password} \\ \#P \leq q}} \left\{ \Pr_{pw \in \mathcal{D}_{pw}} [pw \in P] \right\},$$

where $\mathcal{D}_{pw}(q) = q/\#\mathsf{Password} = q/N$ when the distribution is uniform.

Queries. The adversary $\mathcal{A}$ and its interactions with the players are modeled by the following queries:

- $\mathsf{Execute}(\mathcal{U})$: The adversary gets access to honest executions of the protocol. $\mathcal{A}$ gets back the protocol flows of an honest execution of the protocol P between the players.
- $\mathsf{Send}(U_i, m)$: $\mathcal{A}$ sends a message to the oracle U_i and gets back the response oracle U_i generates in processing the message m according to P. $\mathcal{A}$ initialize the protocol using the $\mathsf{Send}(U_1, \mathtt{Start})$-query and gets back the flow the first player should send out to the second player.
- $\mathsf{Reveal}(U_i)$: This query is only available to $\mathcal{A}$ if oracle U_i holds a session key. $\mathcal{A}$ gets back the session key hold by an oracle.
- $\mathsf{Test}(U_i)$: The query can be only asked once by $\mathcal{A}$, and is answered by flipping a coin b and forwarding the value of $\mathsf{Reveal}(U_i)$ if $b = 1$ or a random value if $b = 0$. The query is only available to $\mathcal{A}$ if U_i is Fresh (see below).

Dealing separately with the $\mathsf{Execute}$ and Send-queries is especially significant in the password-based setting. In effect the number of $\mathsf{Execute}$-queries reflects the number of off-line attempts the adversary can make to guess the password, while the number of Send-queries reflects the (on-line) attacks, thus the number of passwords, he may have tried. Security against dictionary attacks is achieved when the security bound does not depend on the number of $\mathsf{Execute}$-queries, but tightly on $\mathcal{D}_{pw}(q_s)$, where q_s is the number of Send-queries.

Security Notions. The notion of **AKE Security** needs to be achieved in the setting of game $\mathbf{Game}^{\mathsf{ake}}(\mathcal{A}, P)$:

- $\mathbf{Game}^{\mathsf{ake}}(\mathcal{A}, P)$ is initialized by providing coin tosses to $\mathcal{A}$, all U_i, and then:
  1. Provide each player with a password pw distributed in $\mathsf{Password}$,
  2. Initialize any U_i with $sk_{U_i} \leftarrow \text{NULL}$,
  3. Initialize adversary $\mathcal{A}$ with 1^ℓ and oracle access to all U_i,
  4. Run adversary $\mathcal{A}$ and answer queries made by $\mathcal{A}$,
  5. At the end of the game, $\mathcal{A}$ outputs its guess b' for the bit b involved in the Test-query.
- **AKE Security:** In an execution of P, we say that $\mathcal{A}$ *wins* if it asks a single Test-query to a Fresh player U and correctly guesses the bit b used in $\mathbf{Game}^{\mathsf{ake}}(\mathcal{A}, P)$. An oracle U_i is said to be Fresh (or holds a Fresh key sk) if Π_i^t has computed a

session key $sk \neq \text{NULL}$ and neither Π_i^t nor one of its partners has been asked for a Reveal-query. The **AKE advantage** is denoted $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$, where the probability space is over all the random coins of the adversary and all the oracles.

3 The Computational Assumptions

Computational Diffie-Hellman Assumption (CDH). Let's $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order q. A (t, ϵ)-CDH-attacker for $\mathbb{G}$ is a probabilistic Turing machine Δ running in time t that given the triplet (g, g^x, g^y), can find g^{xy} with probability greater than ϵ. We denote this probability $\mathsf{Succ}_{\mathbb{G}}^{\mathsf{cdh}}(\Delta)$:

$$\mathsf{Succ}_{\mathbb{G}}^{\mathsf{cdh}}(\Delta) = \Pr_{x,y}[\Delta(g^x, g^y) = g^{xy}].$$

Trigon Group Computational Diffie-Hellman Assumption (TG-CDH). Let $n \in \mathbb{N}$ be a parameter denoting the number of participants that can join the group, I_n be $\{1, \ldots, n\}$, $\mathcal{P}(I_n)$ be the set of all subsets of I_n, and $\mathcal{T}_n$ be the trigon subset of $\mathcal{P}(I_n)$ as described below, which does not contain I_n:

$$\begin{aligned}\mathcal{T}_n &= \bigcup_{j=1}^{j=n} L_j \text{ with } L_j = \bigcup_{k=1}^{k=j} \left\{ \{i \,|\, 1 \leq i \neq k \leq j, i \neq k\} \right\} \\ &= \big\{\{\}\big\} \cup \big\{\{2\},\{1\}\big\} \cup \big\{\{2,3\},\{1,3\},\{1,2\}\big\} \\ &\quad \cup \big\{\{2,3,4\},\{1,3,4\},\{1,2,4\},\{1,2,3\}\big\} \ldots\end{aligned}$$

Let also $\mathsf{GDH}_{\mathcal{T}_n}$ be the set:

$$\begin{aligned}\mathsf{GDH}_{\mathcal{T}_n} &= \{\mathcal{D}_{\mathcal{T}_n}(x_1, \ldots, x_n) \mid x_1, \ldots, x_n \in_R \mathbb{Z}_q\}, \\ \text{where } \mathcal{D}_{\mathcal{T}_n}(x_1, \ldots, x_n) &= \left\{ \left(J, g^{\prod_{j \in J} x_j} \right) \,\middle|\, J \in \mathcal{T}_n \right\}.\end{aligned}$$

A (t, ϵ)-TG-CDH$_n$-attacker for $\mathbb{G}$ is a probabilistic Turing machine Δ running in time t that given a group Diffie-Hellman trigon $\mathcal{D} = \mathcal{D}_{\mathcal{T}_n}(x_1, \ldots, x_n) \in \mathsf{GDH}_{\mathcal{T}_n}$, can find $g^{x_1 \cdots x_n}$ with probability greater than ϵ. We denote this probability $\mathsf{Succ}_{\mathbb{G}}^{\mathsf{tgcdh}_n}(\Delta)$:

$$\mathsf{Succ}_{\mathbb{G}}^{\mathsf{tgcdh}_n}(\Delta) = \Pr_{x_1,\ldots,x_n}[\Delta(\mathcal{D}_{\mathcal{T}_n}(x_1, \ldots, x_n)) = g^{x_1 \ldots x_n}].$$

Relation between TG-CDH and CDH. The TG-CDH problem has been used previously in the literature and shown to be equivalent to the CDH problem [15, 16]. The TG-CDH is random self-reducible which means that an instance $\mathcal{D}_{\mathcal{T}_n}(x_1, \ldots, x_n)$ can easily be transformed in an instance $\mathcal{D}_{\mathcal{T}_n}(x_1\alpha_1, \ldots, x_n\alpha_n)$, by exponentiating the appropriate elements in the trigon (less than $n^2/2$ exponentiations). The solution of the first one is $S = g^{x_1 \cdots x_n}$, while the solution of the second one is $S' = g^{x_1\alpha_1 \ldots x_n\alpha_n} = S^{\alpha_1 \ldots \alpha_n}$.

4 The Password-Based GOKE

In this section we present the protocol for password-based authenticated group open key exchange (GOKE).

L_1	g			
L_2	g^{x_2}	g^{x_1}		
L_3	$g^{x_2x_3}$	$g^{x_1x_3}$	$g^{x_1x_2}$	
L_4	$g^{x_2x_3x_4}$	$g^{x_1x_3x_4}$	$g^{x_1x_2x_4}$	$g^{x_1x_2x_3}$

Fig. 1. The trigon structure $\mathcal{T}_4$.

4.1 Preliminaries

Let $\mathcal{H}_0$, $\mathcal{H}_1$ and $\mathcal{H}_2$ denote three hash functions, and f denotes the encryption function. (This latter function is in practice instantiated by a mask-generation function computed as the product of the message with a constant value raised to the power of the password [4].) We will also use the transformation ψ which takes as input a k-vector $X = \{X_1, \ldots, X_k\}$ in $\mathbb{G}^k$, an element $x \in \mathbb{Z}_q$, and returns a $k+1$-vector $Y = \{Y_1, \ldots, Y_{k+1}\}$, where for $i = 1, \ldots, k-1, Y_i = X_i^x, Y_k = X_k$, and $Y_{k+1} = X_k^x$. For example, the $n-th$ sequential call is computed as $X_{i+1} = \psi(X_i, x_i)$ (starting at $X_1 = \{g\}$).

4.2 Algorithm

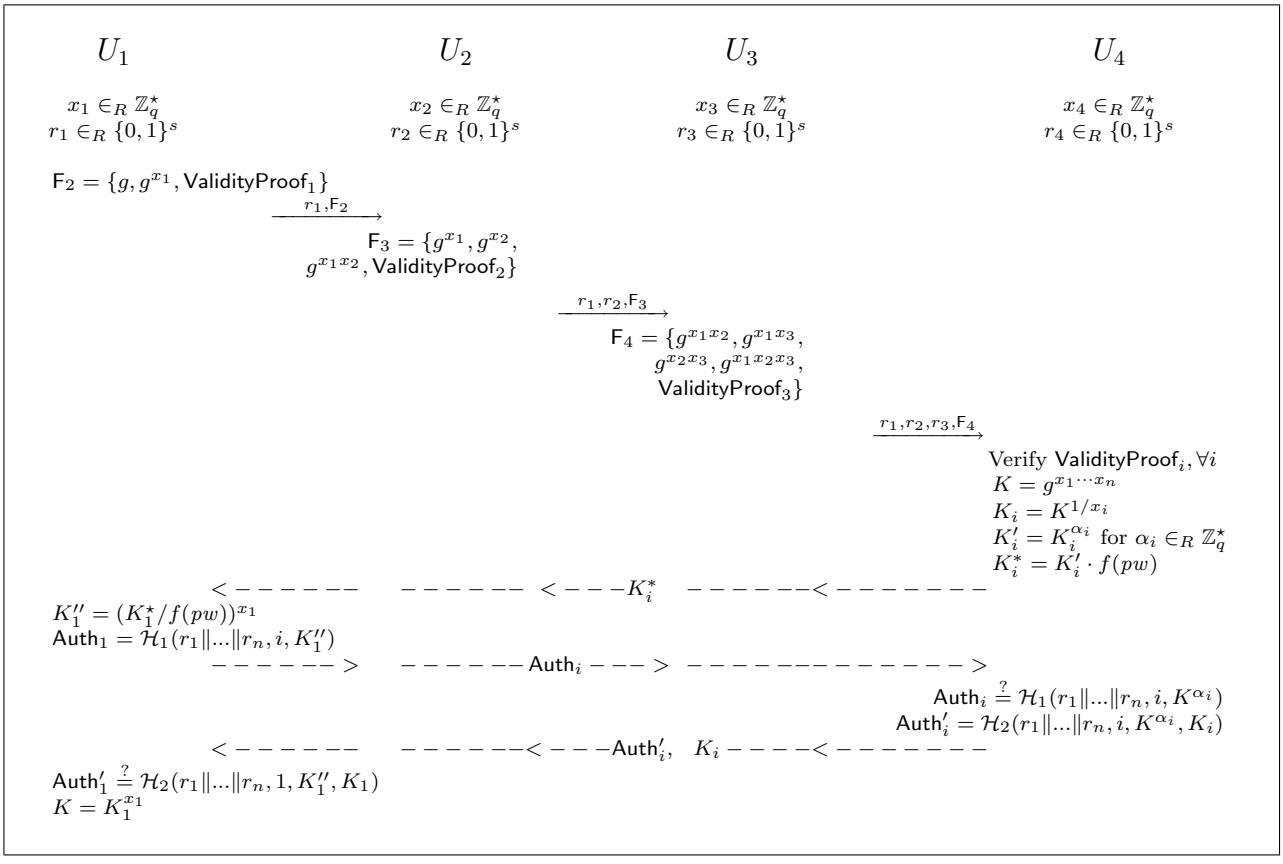

Fig. 2. An honest execution of the password-authenticated Diffie-Hellman Group Open Key Exchange (GOKE) protocol with four players $\mathcal{U} = \{U_1, U_2, U_3, U_4\}$. The session key is $sk_i = \mathcal{H}_0(r_1 \| \ldots \| r_n, K)$.

The protocol consists of an up-flow and a down-flow as depicted on figure 2. In the up-flow, upon receiving i values $\mathsf{F}_i = \{F_{i1}, \ldots, F_{ii}\}$, in addition to a knowledge proof $\mathsf{ValidityProof}_i$, see below, player U_i (for $i < n$) chooses a random exponent x_i in $[1, q-1]$, computes $\mathsf{F}_{i+1} = \psi(\mathsf{F}_i, x_i)$ as the values to be forwarded to the next player.

At the same time, in order to prove consistency in the generated messages, which is required for the security analysis, the player also sends $W_i = g^{x_i}$ together with a flow-dependent, signature of knowledge for x_i and the correct computation of the new values from the ones he received. Player U_i computes the following knowledge proof $\mathsf{ValidityProof}[x_i : W_i = g^{x_i} \ \wedge \ \mathsf{F}_{i+1} = \psi(\mathsf{F}_i, x_i)](\mathsf{F}_i)$. This player then forwards all the received values.

Technical details. It is easy to compute the needed $\mathsf{ValidityProof}$ (Non-Interactive Zero-Knowledge Proofs of Knowledge), by a Schnorr's like non-interactive version of the proof of knowledge [31] using the Fiat-Shamir paradigm [19], in the random oracle model [5], as follows. Recall player U_i has received in the up-flow F_i a vector $(X_1, \ldots, X_i)$ and holds a fresh, randomly chosen Diffie-Hellman exponent x_i for the group key exchange. In order to generate the proof of validity for $\mathsf{F}_{i+1} = \psi(\mathsf{F}_i) = (Y_1, \ldots, Y_{i+1})$, the player chooses one random number r in $\mathbb{Z}_q^\star$ and computes $T_k = X_k^r$, for $k = 1, \ldots, i-1$. Then the random oracle is invoked to produce a value $c = \mathcal{H}(T-1, \ldots, T_k, \mathsf{F}_i, \mathsf{F}_{i+1})$ and the player uses r to compute $s = c - x_i r$. The output $\mathsf{ValidityProof}_i$ is made of $(T_1, \ldots, T_k, s, \mathsf{F}_i)$. The verification is simply done by checking that $T_k = X_k^s Y_k^c$ for all $k \in [1, i-1]$ and $T_i = X_i^s Y_{i+1}^c$ (note that c is easily retrieved by invoking the random oracle again).

Formally speaking, we assume player U_1 to receive $\mathsf{F}_1 = \{g\}$ and $\mathsf{ValidityProof}_1$ being the empty string. This means that each player receives all the values computed by previous players, together with their proofs of validity. He thus has to check all of them before computing his flow.

In the down-flow, upon receiving F_n of length n (as well as the proof $\mathsf{ValidityProof}_n$), the last player U_n chooses a random exponent x_n and computes $(K_1, \ldots, K_n, K) = \psi(\mathsf{F}_n, x_n)$. We thus have $K = g^{x_1 \cdots x_n}$ as the last component. And for each $i = 1, \ldots, n$, we have $K_i = K^{1/x_i}$. Then the player U_n chooses randomly and independently $n-1$ elements $\alpha_i \in \mathbb{Z}_q^\star$ and sets $K_i' = K_i^{\alpha_i}$ for $i < n$. These values are used for the authentication. More precisely, let pw_i be the password common to U_n and U_i (in case of a single-shared password, we have $pw_i = pw$ for all i); U_n sends to every other player U_i the challenge $K_i^\star = K_i' \cdot U^{pw_i}$.

The player authenticates itself to the last player as follows. Upon receiving a challenge $K_i^\star$, each player U_i raises the "unmasked" challenge to the power of its private exponent x_i, and sends the resulting hash value as its authenticator: $\mathsf{Auth}_i = \mathcal{H}_1(r_1 \| \ldots \| r_n, i, K_i'')$ where $K_i'' = (K_i^\star / U^{pw_i})^{x_i}$. The last player in turn authenticates itself to the others as follows. He first waits for having received *all* the authenticators Auth_i from the others. The authenticator Auth_i is verified in a straightforward way: $\mathsf{Auth}_i \stackrel{?}{=} \mathcal{H}_1(r_1 \| \ldots \| r_n, i, K^{\alpha_i})$. One notices that K_i'' should be equal to K^{α_i}. If all the received authenticators are valid the last player sends its own authenticator Auth_i' to every player U_i as well as the value K_i to be used to compute the session key. The value Auth_i' is computed as $\mathcal{H}_2(r_1 \| \ldots \| r_n, i, K^{\alpha_i}, K_i)$. The last player then terminates, accepting $sk = \mathcal{H}_0(r_1 \| \ldots \| r_n, K)$ as its session key.

The protocol terminates successfully once each player has checked the validity of the authenticators: $\mathsf{Auth}_i' = \mathcal{H}_2(r_1 \| \ldots \| r_n, i, K_i'', K_i)$. If the authenticator is valid the player terminates, accepting $sk = \mathcal{H}_0(r_1 \| \ldots \| r_n, K_i^{x_i})$ as its session key.

4.3 Rationale

Two ingredients are essential from the security view point: proofs of validity and key confirmation steps before the last player communicates the K_i's. Otherwise, an adversary could send non-valid flows and then introduce redundancy which could help to perform a dictionary attack: assume that $\mathcal{A}$ tries to impersonate U_1, and thus sends incorrect values $\{g, U^r\}$ to U_2. The latter playing honestly will send $\{g^{x_2}, U^r, U^{rx_2}\}$. Knowing r, the adversary learns U^{x_2}. On top of that, $\mathcal{A}$ also tries to impersonate the group controller and sends $K_2^* = U^s$, for a known s. The authenticator sent back by U_2 is derived from $(U^{x_2})^{s-pw}$, a value easily tested by the adversary, for all the passwords: a dictionary attack.

4.4 Practical considerations

The GOKE protocol can be used with different passwords shared between each group member U_i and the last member U_n in the group. GOKE can also be made more efficient, and therefore more practical, by allowing U_n to verify all the proofs of knowledge once rather than by letting each player U_i verifies the proof of knowledge one at a time. This enables U_n to batch the verification of the knowledge proofs and to speed-up the verification phase by a factor 2 [3, 26].

It is also straightforward to allow users to join and leave the group in the course of the GOKE protocol [17]. When one or more players are added to the group, the last player U_n initiates a sequence of up-flows, starting from him (rather than from the first player): he simply re-computes $\psi(\mathsf{F}_n, x'_n)$ with a new, fresh exponent x'_n then the protocol continues up to the last joining player, who becomes the subsequent group controller. Note that the values $\psi(\mathsf{F}_n, x_n)$ had been previously used in the authentication, however they were sent in the clear (that is, were not masked with the password), so no additional information is leaked when performing the join operation. Similarly, when players leave the group, the group controller sends authenticator to the remaining players, by using fresh, new exponents. Here again, since the authentication procedures are done pairwise and independently, the leaving players cannot gain any useful information.

F_1	g	g^{x_1}			
F_2	g^{x_2}	g^{x_1}	$g^{x_1x_2}$		
F_3	$g^{x_2x_3}$	$g^{x_1x_3}$	$g^{x_1x_2}$	$g^{x_1x_2x_3}$	
F_4	$g^{x_2x_3x_4}$	$g^{x_1x_3x_4}$	$g^{x_1x_2x_4}$	$g^{x_1x_2x_3}$	$g^{x_1x_2x_3x_4}$

Fig. 3. Up-Flows received by the first 4 players.

5 The Security Analysis

In this section we prove that the GOKE protocol is secure against dictionary attacks under the Group Computational Diffie-Hellman assumption and in the random-oracle model. (At first we do not, like Bresson et al., addresses concurrent executions of the protocol.)

Theorem 1. *Let $\mathcal{A}$ be an attacker against the scheme described in Figure 2, in which the password is drawn from a dictionary of size N according to distribution $\mathcal{D}_{pw}$. Let*

q_H and q_s be the number of Hash and Send*-queries the adversary is allowed to make, respectively. Then we have:*

$$\mathsf{Adv}^{\mathsf{ake}}(\mathcal{A}) \leq 12\mathcal{D}_{pw}(q_s) + 4(q_H^2 + q_H)\mathsf{Adv}^{\mathsf{cdh}}_{\mathbb{G}}(t' + \tau_{\mathbb{G}}) \\ +4q_H\mathsf{Adv}^{\mathsf{tgcdh}}_{\mathbb{G}}(t' + q_s\tau_{\mathbb{G}}) + \frac{2q_s}{2^\ell} + \frac{2q_s^2}{2^s}$$

where $t' \leq t + q_s n\tau_{\mathbb{G}} + \frac{64q_Hq_s^2\ln(4q_s/\epsilon)}{\epsilon^3}$ for a value ϵ satisfying $\mathsf{Adv}^{\mathsf{ake}}(\mathcal{A}) \leq 2\epsilon$, and $\tau_{\mathbb{G}}$ is the time needed for computing an exponentiation in $\mathbb{G}$.

Proof. We consider in the following sequence of games the event S defined as $b = b'$, where b is the underlying, random bit used in the Test-query and b' is the bit returned by the adversary. The semantic security aims to make $\Pr[\mathsf{S}]$ negligibly close to $1/2$.

Game $\mathbf{G}_0$: This is the real game, in which we make every player performing the actions as specified in the protocol (exponentiations of received flows, hashing, etc.). Also we simulate the random oracles in a classical way, maintaining lists of already asked values, together with the corresponding answers. In this game, we have by definition:

$$\Pr[\mathsf{S}_0] = \frac{\mathsf{Adv}^{\mathsf{ake}}(\mathcal{A}) + 1}{2}.$$

Game $\mathbf{G}_1$: In this game, the simulated authenticators and the final session key are computed using private random oracles $\mathcal{H}'_1, \mathcal{H}'_2$ and $\mathcal{H}'_0$ in place of $\mathcal{H}_1, \mathcal{H}_2$ and $\mathcal{H}_0$ respectively. More precisely, we do not compute the values of K''_i (neither K) anymore and set:

$$\mathsf{Auth}_i = \mathcal{H}'_1(r_1\|...\|r_n, i),$$
$$\mathsf{Auth}'_i = \mathcal{H}'_2(r_1\|...\|r_n, i)$$
$$\text{and } sk = \mathcal{H}'_0(r_1\|...\|r_n).$$

We stress that these values are perfectly random for the adversary, since the random oracles are private to the simulator. Thus the probability for the adversary of successfully forging an authenticator is at most $q_s/2^\ell$, unless a collision appeared on the SID defined as $r_1\|...\|r_n$ (which is bounded by $q_s^2/2^s$.) Excepted these bad cases, the semantic security of the session key is now perfect (since it is computed using a random oracle the adversary has not access to):

$$\Pr[\mathsf{S}_1] = \frac{1}{2} + \frac{q_s}{2^\ell} + \frac{q_s^2}{2^s}.$$

Moreover, the two games are indistinguishable unless the attacker asks one of the "bad" values $(r_1\|...\|r_n, i, K^{\alpha_i})$ to the random oracle[1], which event is denoted AskH. The remaining of the proof is thus devoted to upper-bound the probability of such event.

$$|\Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_0]| \leq \Pr[\mathsf{AskH}_1].$$

Game $\mathbf{G}_2$: We show how to simulate the instances of the players without knowing the exponents. We assume we are given a TG-CDH-instance $\mathcal{D}(a_1, \ldots, a_n)$. Since we have access to the extractors associated to any proof of knowledge (as explained in the appendix A), we can properly deal with the cases the received values have been

[1] More precisely, if he asks either $(r_1\|...\|r_n, i, K^{\alpha_i})$ to $\mathcal{H}_1$ or $(r_1\|...\|r_n, i, K^{\alpha_i}, K_i)$ to $\mathcal{H}_2$ or $(r_1\|...\|r_n, K)$ to $\mathcal{H}_0$.

built by the adversary. We make use of this functionality to embed the instance $\mathcal{D}$ in the protocol flows: We simulate the protocol by answering each $\mathsf{Send}(U_i, \mathsf{F}_i)$-query by extracting all exponents put in it by the adversary (for the players he is controlling); we then "remove" these exponents, and obtain a line of the trigon. Taking the next line, we then "reintroduce" the adversary's exponents. On top of that, we randomize the process by using multiplicative random self-reducibility (with known randomizing exponents) to process multiple queries asked to the same player: this essentially adds at most n exponentiations per Send-query (the $\mathsf{Execute}$-queries are simulated by simply randomizing the instance $\mathcal{D}$). The last call (if asked to player U_n) provides us with values K_i, that we use to compute the challenges K_i^*. Recall that we do not have to compute the values K_i'' nor K, then the remaining of the protocol (authentication flows) is easily simulated.

The following (basic) lemma shows that such a simulation is always possible:

Lemma 2. *For each $\mathsf{Send}(U_i, \mathsf{F}_i)$-query asked to a player instance U_i such that the number of simulated players (i.e. players not under $\mathcal{A}$'s control) up to index i is equal to t, the answer is derived from the line L_t in instance $\mathcal{D}$.*

The proof uses a simple induction on t. Also we introduce the notation $t = \phi(i)$; this simply means that player U_i is simulated using the line $L_{\phi(i)}$ of the instance $\mathcal{D}$. In other words, U_i implicitly makes use of the exponent $a_{\phi(i)}$.

We easily see that the simulation is perfectly indistinguishable from the previous game, if conditioned to the success probability of correctly extracting the adversary's exponents. Then we get (according to section A and letting $\epsilon = \Pr[\mathsf{AskH}_1]$):

$$\Pr[\mathsf{AskH}_2] \geq \Pr[\mathsf{AskH}_1]/2.$$

However, the times of the simulation is a bit more expensive:

$$t_2 \leq t + q_s n \tau_{\mathbb{G}} + \frac{64 q_H q_s^2 \ln(4q_s/\epsilon)}{\epsilon^3},$$

for any $\epsilon \geq \Pr[\mathsf{AskH}_1]/2$. The former contribution is for the random-self reduction, while the latter is for the extraction (see appendix A.)

We now derive two games, in order to analyze the probability of passive or active attacks. We first analyze active attacks.

Game $\mathbf{G}_3$: In this game, we show how to upper-bound the probability of event AskH. To do so, we modify the way the challenges K_i^* are computed, in such a way that the password is not used anymore and becomes information-theoretically hidden to the adversary. Indeed, we note that the challenges K_i^* are independent, random elements in $\mathbb{G}$, so we just simulate them by choosing a random exponent α_i and sets $K_i^* = U^{\alpha_i}$. Intuitively we study what $\mathcal{A}$ can do/learn on the authentication flows between U_n and U_i.

Several cases may then appear, each of them leads to a specific bounds. These cases correspond to the attacks the adversary can mount against the 2-party authentications (between the last player and another one). We stress that these cases are disjoint because we focus on the first occurrence of the "bad" hash-query; thus the global upper-bound is just the sum. Much more important, the simulation is identical in these three cases, so we do not need to condition by the probability of having correctly "guessed" the case. For evaluating AskH, we postpone the choice of the password after the answer of the adversary.

Case AskH^1*: the value* K_i^* *is sent by the simulator, but* U_i *has not been simulated.* In this case, $\mathcal{A}$ is likely to know the underlying x_i so we just notice the following. If AskH^1 happens, the query K^{α_i} asked by $\mathcal{A}$ is equal to $(U^{\alpha_i - pw})^{x_i}$, but, as said above, the password can be chosen at the very end of the protocol. The probability of such an event is thus upper-bounded by $\mathcal{D}_{pw}(q_s)$.

Case AskH^2*: the value* K_i^* *is sent by the simulator, and* U_i *has been simulated as well.* As described before, the challenges K_i^* are simulated by random elements U^{α_i}, and the password is chosen at the very end, being thus independent from $\mathcal{A}$'s view. Our goal in this last case is to show that, even if $\mathcal{A}$ controls some player U_j (knowing x_j), it cannot learn useful information from the $U_i \leftrightarrow U_n$ authentication. More precisely, because $\mathcal{A}$ does not know $a_{\phi(i)}$, he cannot ask the query K_i'' without (implicitly) breaking the CDH problem. The following lemma formally proves this.

Lemma 3. *If there is a query* $\mathsf{DH}(K_i^*/U^{pw}, W_i)$ *for a player* U_i *not under* $\mathcal{A}$*'s control, one can solve the* CDH *problem.*

Proof. It is easy to see that since $K_i^*/U^{pw} = U^{\alpha_i - pw}$. Thus if we retrieve (with probability at least $1/q_H$) a value $Z = \mathsf{DH}(K_i^*/U^{pw}, W_i)$, then we can compute $\mathsf{DH}(U, W_i) = Z^{1/(\alpha_i - pw)}$. Here $W_i = g^{a_{\phi(i)}}$.

Case AskH^3*: the values* K_i^* *and* Auth_i' *are sent by the adversary.* Of course one cannot prevent the adversary to test a password by sending to the players' instances many challenges he built by himself —and for which he probably knows the discrete logarithm. The following lemma shows, however, that $\mathcal{A}$ cannot try more than one password when sending a challenge, then his probability of success reduces to $1/N$ (N being the number of passwords) for each Send-query. More precisely, if he tries to distinguish the simulated authenticators from the true ones by testing two different passwords π and π', and asking the corresponding queries K^{α_i} to $\mathcal{H}_1$, the solution to a Diffie-Hellman computational problem can be recovered.

Lemma 4. *Let* K^* *be a challenge sent by the adversary. Let us assume that the* $\mathcal{H}$*-list contains at least two queries* $\mathsf{DH}(K^*/U^{\pi}, W_i)$ *and* $\mathsf{DH}(K^*/U^{\pi'}, W_i)$*, for* $\pi \neq \pi'$*, then one can solve the* CDH *problem in time* $t + \tau_{\mathbb{G}}$*.*

Proof. Remind that W_i denotes the quantity $g^{a_{\phi(i)}}$. We first notice that all flows sent in the protocol are of the form $g^{\prod_{j \in J} x_j}$; this is due to the fact that each flow is sent with a ValidityProof proof. The consequence is that the discrete logarithms of U in any of these values remains unknown (otherwise one could easily get $\log_g U$ by removing the exponents embedded by $\mathcal{A}$). Now let us assume that there exist two queries Z and Z' in the $\mathcal{H}$-list, satisfying:

$$Z = \mathsf{DH}(K^*/U^{\pi}, W_i) \text{ and } Z' = \mathsf{DH}(K^*/U^{\pi'}, W_i).$$

By division, we get $Z/Z' = \mathsf{DH}(U, W_i)^{\pi' - \pi}$. Since $\pi \neq \pi'$ and the group order is a prime, we can set $w = (\pi' - \pi)^{-1} \mod q$. It follows that $\mathsf{DH}(U, W_i) = (Z/Z')^w$. As noticed above, this value cannot be computed from the flows themselves. Thus, unless the adversary can solve this Diffie-Hellman instance, he can test at most one password when sending a challenge K^*.

If the adversary forges an authenticator Auth_i', one shows that $\mathcal{A}$ cannot guess the password better than at random since neither the players' authenticators nor the

challenges provide information on it; indeed, all these values are simulated by pure random strings or elements. Therefore the probability that $\mathcal{A}$ makes a $\mathcal{H}_2$-query that helps to distinguish the games is upper-bounded by $\mathcal{D}_{pw}(q_s)$.

This leads to $\Pr[\mathsf{AskH}_3^3] \leq 2 \times \mathcal{D}_{pw}(q_s) + q_H^2 \times \mathsf{Succ}_{\mathbb{G}}^{\mathsf{cdh}}(t + \tau_{\mathbb{G}})$.

Gathering the three cases, we obtain (for active attacks):

$$\Pr[\mathsf{AskH}_3] \leq 3 \times \mathcal{D}_{pw}(q_s) + (q_H^2 + q_H) \times \mathsf{Succ}_{\mathbb{G}}^{\mathsf{cdh}}(t + \tau_{\mathbb{G}}).$$

Game $\mathbf{G}_4$: It remains to deal with passive attacks. In a passive eavesdropping, we note that all exponents are successively embedded in the protocol flows using the ψ function (that is, $\phi(i) = i$), and that the session key is derived from the TG-CDH secret (by the random self-reduction known to the simulator.) Also we note that the challenges K_i^* as well as the values K_i can be properly simulated, without any additional material (these values are included in the last line of the trigon instance).

In such a game, the probability that $\mathcal{A}$ asks the value $(r_1\|...\|r_n, K)$ to the random oracle $\mathcal{H}_0$ is easily related to the probability of solving the TG-CDH-problem for the instance $\mathcal{D}$:

$$\Pr[\mathsf{AskH}_4] \leq q_H \times \mathsf{Succ}_{\mathbb{G}}^{\mathsf{tgcdh}}(t_3 + \tau_{\mathbb{G}}).$$

Putting all together, we obtain the claimed result. □

6 Acknowledgment

The second author is supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, Mathematical Information and Computing Sciences Division, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This document is report LBNL-57432. Disclaimer available at `http://www-library.lbl.gov/disclaimer`.

The third author has been supported in part by France Telecom R&D through the contract CIDRE, between France Telecom R&D and Ecole Normale Supérieure, as well as by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

References

1. M. Abdalla, O. Chevassut and D. Pointcheval. One-time Verifier-based Encrypted Key Exchange. In *International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, pages 47–64, Jan 2005.
2. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In B. Preneel, editor, *Proc. of Eurocrypt '00*, LNCS 1807, pages 139–155. Springer-Verlag, 2000.
3. M. Bellare, J. A. Garay, and T. Rabin. Fast Batch Verification for Modular Exponentiation and Digital Signatures. In *Eurocrypt '98*, LNCS 1403, pages 236–250. Springer-Verlag, Berlin, 1998.
4. M. Abdalla and D. Pointcheval. Simple Password-Based Encrypted Key Exchange Protocols. In *CT – RSA '05*, LNCS, pages 191–208. Springer-Verlag, Berlin, 2005.
5. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.
6. M. Bellare and P. Rogaway. The AuthA Protocol for Password-Based Authenticated Key Exchange. Contributions to IEEE P1363. March 2000.
7. S. M. Bellovin and M. Merrit. Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks. In *Proc. of the Symposium on Security and Privacy*, pages 72–84. IEEE, 1992.
8. S. M. Bellovin and M. Merrit. Augmented Encrypted Key Exchange: A Password-based Protocol Secure against Dictionary Attacks and Password File Compromise. In *Proc. of the 1st CCS*, pages 244–250. ACM Press, New York, 1993.

9. D. Boneh. The Decision Diffie-Hellman Problem. In *Third Algorithmic Number Theory Symposium*, LNCS 1423, pages 48–63. Springer-Verlag, 1998.
10. N. Borisov, I. Goldberg, and D. Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. In *Mobicom*, 2001.
11. V. Boyko, P. MacKenzie, and S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In B. Preneel, editor, *Proc. of Eurocrypt '00*, LNCS 1807, pages 156–171. Springer-Verlag, 2000.
12. The Official Bluetooth Membership Site. Available at http://www.bluetooth.org .
13. E. Bresson, O. Chevassut, and D. Pointcheval. New Security Results on Encrypted Key Exchange. *Proc. of International Workshop on Practice and Theory in Public Key Cryptography (PKC)*. Springer, February 2004.
14. E. Bresson, O. Chevassut, and D. Pointcheval. Security Proofs for an Efficient Password-based Key Exchange. *Proc. of 10th ACM Conference on Computer and Communications Security*. ACM Press, Nov 2003.
15. E. Bresson, O. Chevassut, and D. Pointcheval. Group Diffie-Hellman Key Exchange Secure against Dictionary Attacks. In Y. Zheng, editor, *Proc. of Asiacrypt '2002*. Springer, December 2002.
16. E. Bresson, O. Chevassut, and D. Pointcheval. The Group Diffie-Hellman Problems. In H. Heys and K. Nyberg, editors, *Proc. of SAC '2002*, LNCS. Springer-Verlag, August 2002.
17. E. Bresson, O. Chevassut, D. Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange: The Dynamic Case. In *Proc. of Asiacrypt 01*, pages 290–309, Dec 2001.
18. W. Diffie and M. Hellman. New Directions In Cryptography. In *IEEE Transactions on Information Theory*, volume IT-22(6), pages 644–654, November 1976.
19. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *Crypto '86*, LNCS 263, pages 186–194. Springer-Verlag, Berlin, 1987.
20. O. Goldreich and Y. Lindell. Session-Key Generation using Human Passwords Only. In J. Kilian, editor, *Proc. of Crypto '01*, LNCS 2139, pages 408–432. Springer-Verlag, August 2001.
21. M. Jakobsson and S. Wetzel. Security Weaknesses in Bluetooth. In *Proc. of the RSA Cryptographer's Track (RSA CT '01)*, LNCS 2020, pages 176–191. RSA Data Security, Springer-Verlag, 2001.
22. J. Katz, R. Ostrovsky, and M. Yung. Efficient Password-Authenticated Key Exchange using Human-Memorable Passwords. In *Proc. of Eurocrypt '01*, LNCS 2045, pages 475–494. Springer-Verlag, May 2001.
23. P. MacKenzie. More Efficient Password Authenticated Key Exchange. In D. Naccache, editor, *RSA Conference '01*, LNCS 2020, pages 361–377. Springer-Verlag, 2001.
24. P. D. MacKenzie. The PAK suite: Protocols for Password-Authenticated Key Exchange. Technical Report 2002-46, DIMACS, 2002.
25. Motorola Inc, Mobile Mesh Networks Technology. Available at http://www.meshnetworks.com .
26. D. M'Raïhi and D. Naccache. Batch Exponentiation – A Fast DLP-based Signature Generation Strategy. In *Proc. of the 3rd CCS*, pages 58–61. ACM Press, New York, 1996.
27. M. Naor and O. Reingold. Number-Theoretic Constructions of Efficient Pseudo-Random Functions. In *Proc. of 38th FOCS*, pages 458–467. IEEE, 1997.
28. PacketHop Inc, Mobile Mesh Networking Solutions. Available at http://www.packethop.com .
29. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
30. A. Sahai. Non-Malleable Non-Interactive Zero-Knowledge and Chosen-Ciphertext Security. In *Proc. of the 40th FOCS*. IEEE, New York, 1999.
31. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto '89*, LNCS 435, pages 235–251. Springer-Verlag, Berlin, 1990.
32. V. Shoup. OAEP Reconsidered. In J. Kilian, editor, *Proc. of Crypto' 01*, LNCS 2139, pages 239–259. Springer-Verlag, 2001.
33. M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman Key Distribution Extended to Groups. In *ACM Conference on Computer and Communications Security*, March, 1996.
34. Wi-Fi Alliance http://www.wi-fi.org .
35. N. Winget, R. Housley, D. Wagner, and J. Walker. Security Flaws in 802.11 Data Link Protocols. Communications of the ACM, Vol. 46, No. 5, May, 2003.

A Proofs of Validity and Extractors

As shown in section 4, an adversary could make calls with invalid inputs, we thus need to make sure this cannot happen: we require *proofs of validity*. We then denote

by $\mathsf{NIZKProof}(X, Y)$ a non-interactive zero-knowledge proof of knowledge and membership for x, such that $Y = \psi(X, x)$, where X is a k-vector. Such proofs can be efficiently done in the random-oracle model [5], with simple indistinguishable simulation under the decisional Diffie-Hellman assumption, and with the additional property of simulation-soundness [30]. Furthermore, we can extract x with overwhelming probability, using an improvement of the forking lemma [29], as we now explain with more details.

A.1 Main ideas

Each proof $\mathsf{NIZKProof}(X, Y)$ contains a Non-Interactive Zero-Knowledge Proof of Knowledge that the sender "knows" the exponent used to build the element. In the Random Oracle Model (ROM), a truly random hash function is used to construct such proofs. In order to be able to use the property of "proof of knowledge", we need to build *extractors of knowledge*. However such extractors must be constructed sequentially, otherwise the complexity grows exponentially with the number of players n. To achieve such situation, we require the hash function to be called on the input the previously produced proofs of knowledge. In other words, and very informally speaking, when querying the random oracle to build proof number n, the $(n-1)$-th proof itself must be part of the query: this ensures that, in a complete list of $\mathsf{ValidityProof}$, each call to the random oracle is made *after* the previous proof of knowledge has been computed. To propagate this property over all the executions of the GOKE protocol, we restrict our scenario to **non-concurrent executions**. As a consequence, the construction of knowledge extractors can be properly chained and our proof complexity does not explode.

A.2 Details

We now show that from a correct proof (submitted to the random oracle when building further proofs of validity) we can extract the underlying exponent, with non-negligible probability. We then consider that the adversary has constructed a correct

$$\mathsf{NIZKProof}(X, Y) = \mathsf{ValidityProof}[x : Y = \psi(X, x)](\mathcal{Z}),$$

where $\mathcal{Z}$ is itself a proof of knowledge for a correct computation of X. We emphasize that there may be several rounds of communication between the computation of this correct proof and the moment it is queried to the random oracle.

Let us assume that the adversary $\mathcal{A}$ is able to produce such a proof with probability $\nu = \epsilon + 1/2^k$, where k is the output size of the random oracle, and thus measures the probability of correctness by chance.

We denote by $\mathcal{S}$ the set of choices for the random coins ω of $\mathcal{A}$ and the random oracle $\mathcal{H}$ (restricted to queries which include $\mathcal{Z}$). Moreover for each of them we denote by $\mathsf{Ind}(\omega, \mathcal{H})$ the index of the $\mathcal{H}$-query which corresponds to the *crucial query*: the one in the final proof. With probability ϵ, this index is between 1 and q_H.

By replaying the adversary $\mathcal{A}$ with the same random tape ω, and by simulating another random oracle $\mathcal{H}'$ that is the same as $\mathcal{H}$ up to the $\mathsf{Ind}(\omega, \mathcal{H})$-th query (excluded), we can extract the value x whose knowledge is proved. What remains to show is that this strategy succeeds with overwhelming probability (over the possible choices of $(\omega, \mathcal{H})$) and after a polynomially bounded number of replays.

For any i, we define the set of the random values which leads to index i: $\mathcal{S}_i = \{(\omega, \mathcal{H}) \in \mathcal{S} | \mathsf{Ind}(\omega, \mathcal{H}) = i\}$. We define the good indices i, in $I = \{i | \Pr[\mathcal{S}_i|\mathcal{S}] \geq \alpha/2q_H\}$, for an appropriate α we will define later. Finally, we define the good beginnings:

$$\Omega_i = \{(\omega, \mathcal{H}) | \Pr_{\mathcal{H}'}[(\omega, \mathcal{H}') \in \mathcal{S}_i | \mathcal{H}' \equiv_i \mathcal{H}] \geq \alpha^2\epsilon/4q_H\},$$

where the relation $\mathcal{H}' \equiv_i \mathcal{H}$ means that we restrict the choice of $\mathcal{H}'$ to the random oracles providing the same answers as $\mathcal{H}$ for the first i queries. For any $i \in I$, using the Bayes' law:

$$\begin{aligned}
&\Pr[(\omega, \mathcal{H}) \notin \Omega_i | (\omega, \mathcal{H}) \in \mathcal{S}_i] \\
=& \Pr[(\omega, \mathcal{H}) \in \mathcal{S}_i | (\omega, \mathcal{H}) \notin \Omega_i] \\
&\times \Pr[(\omega, \mathcal{H}) \notin \Omega_i] / \Pr[(\omega, \mathcal{H}) \in \mathcal{S}_i] \\
<& \alpha^2\epsilon/4q_H \times 1/(\alpha\epsilon/2q_H) = \alpha/2.
\end{aligned}$$

And thus, $\Pr[\Omega_i|\mathcal{S}_i] \geq 1 - \alpha/2$. Furthermore, since all the subsets $\mathcal{S}_i$ are disjoint,

$$\sum_{i \in I} \Pr[\mathcal{S}_i|\mathcal{S}] = 1 - \sum_{i \notin I} \Pr[\mathcal{S}_i|\mathcal{S}] \geq 1 - q_H \times \alpha/2q_H = 1 - \alpha/2,$$

and consequently

$$\begin{aligned}
&\Pr_{\omega, \mathcal{H}}[\exists i \in I, (\omega, \mathcal{H}) \in \Omega_i \cap \mathcal{S}_i | \mathcal{S}] \\
=& \Pr[\cup_{i \in I}, \Omega_i \cap \mathcal{S}_i | \mathcal{S}] \\
=& \sum_{i \in I} \Pr[\Omega_i \cap \mathcal{S}_i | \mathcal{S}] = \sum_{i \in I} \Pr[\Omega_i | \mathcal{S}_i] \cdot \Pr[\mathcal{S}_i | \mathcal{S}] \\
\geq& (1 - \alpha/2) \cdot \sum_{i \in I} \Pr[\mathcal{S}_i | \mathcal{S}] \geq (1 - \alpha/2)^2 \geq 1 - \alpha.
\end{aligned}$$

Then, when $\mathcal{A}$ outputs a valid proof, with probability greater than $1 - \alpha$, $i = \mathsf{Ind}(\omega, \mathcal{H}) \in I$ and $(\omega, \mathcal{H}) \in \Omega_i \cap \mathcal{S}_i$. In this case, we know that we can have a second success with probability greater than $\alpha^2\epsilon/4q_H$. And thus, after $-4q_H(\ln \alpha)/\alpha^2\epsilon$ replays, the probability of failure is less than α: the probability of success is more than $1 - \alpha$.

Globally, the probability of extraction is more than $1 - 2\alpha$, within time bounded by $4q_H(\ln 1/\alpha)/\alpha^2\epsilon$.

A.3 Chaining the Extractors

In fact, we built a *black-box extractor* in the sense that extracting a witness for a given proof of knowledge is feasible independently from the other proofs submitted by the adversary. If we suppose that the advantage of the adversary is greater than ϵ, it means that it can complete an attack with probability greater than ϵ (independently of its success since an incomplete game cannot lead to any advantage). Now let us denote by q_s the number of Send-queries (active attacks) the adversary is allowed to make. By taking $\alpha = \epsilon/4q_s$, all the extractions are successful with probability greater than $1 - \epsilon/2$, and thus there is at least one failure with probability bounded by $\epsilon/2$. Excluding these executions just reduces by one half the advantage of the adversary: we can assume that all the proofs built by the adversary are given with the witness.