The Fifth IEEE International Conference on Mobile and Wireless Communications Networks (MWCN '03).
(october 27 - 29, 2003, Singapore)

Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices

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

[1] Cryptology department – CELAR, 35170 Bruz, France
Emmanuel.Bresson@polytechnique.org
[2] Lawrence Berkeley National Lab. – Berkeley, CA, USA [†]
{OChevassut,AEssiari}@lbl.gov
[3] École normale supérieure – Paris, France
David.Pointcheval@ens.fr

Abstract. Wireless networking has the power to fit the Internet with wings, however, it will not take off until the security technological hurdles have been overcome. In this paper we propose a *very efficient* and *provably-secure* group key agreement well suited for unbalanced networks consisting of devices with strict power consumption restrictions and wireless gateways with less stringent restrictions. Our method meets practicability, simplicity, and strong notions of security.

1 Introduction

Wireless technology has become more pervasive as Internet electronic and commerce transactions on mobile devices have gained in popularity. Institutions and industries are hankering for the power and flexibility of wireless networks, but many are postponing rollouts in strategic areas until they are convinced that their systems are not at risk. The security technologies currently deployed to protect the Internet against attacks are not fully applicable to the wireless Internet since the traditional Internet does not typically place constraints on available power consumption or bandwidth. The nodes in a wireless network are usually mobile and have computation bandwidth capabilities that place severe restrictions when designing cryptographic protocols. Storage limitation on the other hand is becoming less of an issue as many add-on memory cards are now widely available.

In the present paper we have focused on computing applications involving clusters of mobile devices [4, 15, 17]. The Wired Equivalent Privacy (WEP) protocol, which is part of the IEEE 802.11 standard, specifies how to protect the traffic between mobile devices and access points (i.e. gateways) using *pre-established* session keys without specifying how the keys are established. This lack of proper key-establishment scheme has opened the door to various attacks [6]. Our contribution in this paper is a *provably-secure* authenticated group key-exchange scheme based on public-key cryptography that can complement the WEP protocol. Schemes based on symmetric cryptography have obvious performance advantages over public-key cryptography, but suffer from a complex key management; they require trust in the entire network as a device moves from one domain to another. Other schemes based on public-key technology trade less computation for more communication rounds, but are still too costly to be practical for wireless networks that involve low-power computing devices [1, 2, 9, 10].

Our key-exchange scheme allows a cluster of mobiles and one wireless gateway to dynamically agree on a session key. It shifts the computational burden to the gateway and provides mobile devices with the ability to perform most of the public-key cryptographic computations off-line.

[*] Work done while at ENS, Paris.

[†] The authors are 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-AC03-76SF00098. This document is report LBNL-52542. Disclaimer available at http://www-library.lbl.gov/disclaimer.

© World Scientific Publishing 2003.

This scheme can furthermore be combined with a group Diffie-Hellman key exchange scheme [9] to cover wireless environments involving more than one gateway [12]. A mobile device would only perform one public-cryptographic computation as it moves from one wireless domain to the other.

The paper is organized as follows. In Section 3 we present a scheme that achieves "implicit" authentication and in Section 5 we discuss enhancements to achieve "explicit" authentication.

2 Modeling Unbalanced Wireless Networks

Wireless Nodes. The wireless-communication system we model is a set $\mathcal{C}$, of N wireless-capable mobile devices (also called *clients*), and a wireless gateway (also called *server* or base station). We assume the clients and the server do not deviate from the algorithm they are expected to execute. We consider a nonempty subset of $\mathcal{C}$ which we call the *wireless client group* $\mathcal{G}_c$ that consists of the clients communicating with the server. The server S has the special role of adding and removing clients from the group $\mathcal{G}_c$. (In practice, this server covers an entire wireless region called a cell or domain and, thus, it never leaves, hence its special role.) Each mobile U, as well as the base station, also holds a long-lived key LL_U which is a pair of matching public/private keys.

Abstract Interface. We define the basic structure of a group key agreement scheme for unbalanced wireless networks. The scheme GKE consists of four algorithms:

- The *key generation algorithm* $\mathsf{GKE.KGen}(1^\ell)$ is a probabilistic algorithm which on input a security parameter 1^ℓ, provides each client U_i in $\mathcal{C}$ and the base station with a long-lived key.
- The *setup algorithm* $\mathsf{GKE.Setup}(\mathcal{J})$ is an interactive protocol which on input of a set of clients $\mathcal{J}$, sets the wireless client group to be $\mathcal{G}_c = \mathcal{J}$ and provides each client U in $\mathcal{G}_c$ with a secret value sk shared with the base station.
- The *join algorithm* $\mathsf{GKE.Join}(\mathcal{J})$ is an interactive protocol which on input of a set of clients $\mathcal{J}$, updates the wireless client group $\mathcal{G}_c$ to be $\mathcal{G}_c \cup \mathcal{J}$, and provides each client U in $\mathcal{G}_c$ with a new shared secret value sk.
- The *remove algorithm* $\mathsf{GKE.Remove}(\mathcal{J})$ is an interactive protocol which on input of a subset $\mathcal{J}$ of the wireless client group $\mathcal{G}_c$, updates the latter to be $\mathcal{G}_c \backslash \mathcal{J}$, and provides each client U in $\mathcal{G}_c$ with a new shared secret value sk.

Adversary. The adversary $\mathcal{A}$ is neither a client, nor the server, and in our formalization it is given enormous capabilities to closely model its abilities in the real life: the adversary can tap on the wire to eavesdrop, delete, delay, insert, replay, modify messages. We model these capabilities through the following queries:

- The adversary $\mathcal{A}$ has the ability to send arbitrary messages to the base station S using the $\mathsf{SendServer}$-query. This query on input a message m returns the message that S would have produced in processing the message. If the message is not a valid one, the query returns a special symbol $\perp$. The query $\mathsf{SendServer}(\mathtt{setup}, \mathcal{J})$ (resp. $\mathsf{SendServer}(\mathtt{join}, \mathcal{J})$ and $\mathsf{SendServer}(\mathtt{remove}, \mathcal{J})$) return the flows output by the base station when initiating a $\mathsf{Setup}(\mathcal{J})$ (resp. $\mathsf{Join}(\mathcal{J})$ and $\mathsf{Remove}(\mathcal{J})$) depending on the scheme.
- The adversary $\mathcal{A}$ has the ability to send arbitrary messages to clients using a Send-query. This query on input a client U_i and a message m returns the message U_i would have produced in processing the message. If the message is not a valid one, a special symbol $\perp$ is returned. The queries $\mathsf{Send}(\mathtt{setup}, \mathcal{J})$, $\mathsf{Send}(\mathtt{join}, \mathcal{J})$ and $\mathsf{Send}(\mathtt{remove}, \mathcal{J})$, respectively, return the flows output by the client when initiating a $\mathsf{Setup}(\mathcal{J})$, $\mathsf{Join}(\mathcal{J})$ and $\mathsf{Remove}(\mathcal{J})$, respectively, depending on the scheme.

– Known-key attacks are modeled by the Reveal-query. This query allows the adversary $\mathcal{A}$ to learn the value of a particular session key sk, if of course the attacked player (client or server) has already computed the key. This query on input a player returns sk to the adversary but not the player internal state.
– Forward-secrecy is modeled through the Corrupt-query. This query allows $\mathcal{A}$ to learn the value of long-lived keys. This query on input a player returns to the adversary the value of the long-lived keys. Forward-secrecy means that loss of a LL-key does not compromise the semantic security of previously-distributed session keys.

The adversary has, unlike in the wired environment, true and full control over the communication medium. The adversary can cut off mobiles, modify the mobile network topology and make mobiles disappear and reappear continuously. The adversary's abilities to modify the network topology are modeled through the following queries:

– The $\mathsf{Send}/\mathsf{SendServer}(\texttt{setup}, \mathcal{J})$-queries provide the adversary the capability to bring together a pool of players.
– The $\mathsf{Send}/\mathsf{SendServer}(\texttt{remove}, \mathcal{J})$-queries provide the adversary the ability to make clients disappear.
– The $\mathsf{Send}/\mathsf{SendServer}(\texttt{join}, \mathcal{J})$-query provide the adversary the ability to make clients reappear or simply bring more clients together.

3 Key Agreement

This section provides a method accommodating group key agreement to mobiles lacking the computational resources to perform multiple on-line computation in a finite cyclic group (such as modular exponentiation), but with enough computational resources to perform symmetric-cryptographic operations. A key agreement for wireless networks is designed to provide a collection of heterogeneous wireless-capable devices with a group session key to be used to set up a security association within which messages multicast over the wireless link are cryptographically protected.

We present a method accommodating group key agreement to mobiles lacking the computational resources to perform any on-line computation in a finite cyclic group (such as modular exponentiation), but can only do symmetric cryptographic operations. In the following, we do not explicitly separate the pre-computation part, but the reader will easily make this distinction between data that can be computed before having received anything and data that cannot. It is in the same vein as [16] in that implicit authentication of the server is provided by proving its ability to decrypt, and the implicit authentication of the mobiles is done through signatures. Signatures and encryptions, which have to be computed by the low-power devices, can actually be precomputed. Thereafter, very few computations have to be performed on-line.

3.1 Protocol preliminaries

The session-key space SK associated to this method is $\{0,1\}^\ell$ equipped with a uniform distribution, where ℓ is a security parameter. Arithmetic is in a finite cyclic group $\mathbb{G} = \langle g \rangle$ of ℓ-bit prime order q. Both g and q are publicly known. There are also three hash functions $\mathcal{H} : \{0,1\}^\star \to \{0,1\}^\ell$, $\mathcal{H}_0 : \{0,1\}^\star \to \{0,1\}^{\ell_0}$, where ℓ_0 needs not be equal to ℓ, and $\mathcal{H}_1 : \{0,1\}^{\ell_1} \times \mathbb{G} \to \{0,1\}^{\ell_0}$, where ℓ_1 is the maximal bit-length of a counter c used to prevent replay attacks.

We consider a signature scheme $\mathsf{SIGN} = (\mathsf{SIGN.KGen}, \mathsf{SIGN.Sig}, \mathsf{SIGN.Ver})$. Each client U_i holds a pair of signing private/public key (SK_i, PK_i) which are the output of the key generation

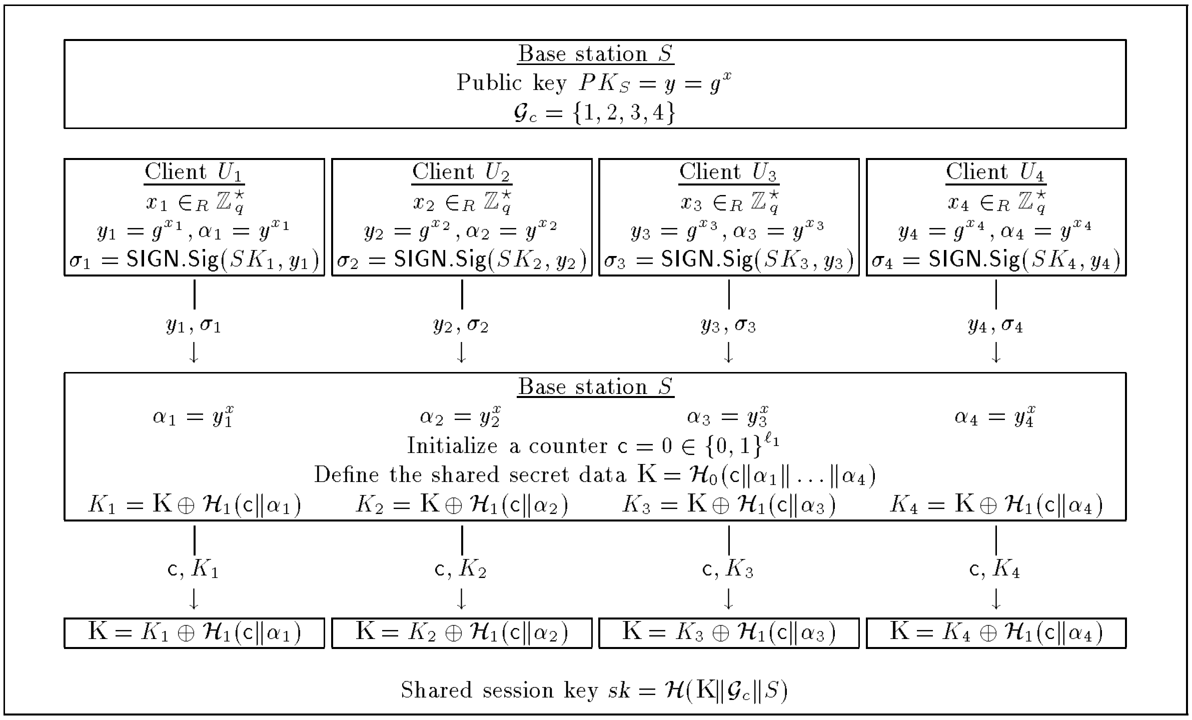

Fig. 1. An execution of the Setup algorithm with the five devices U_1, U_2, U_3, U_4 and S.

signature scheme algorithm $\mathsf{SIGN.KGen}$. One would probably argue that when dealing with low-power computing mobiles, special low-cost [19] or on-line/off-line [13, 23] signature schemes are required. However, the messages to be signed are in our setting known at pre-computation time and, thus, a mobile does not have to compute anything on-line to send its contribution.

3.2 Algorithms

Key Generation The algorithm $\mathsf{GKE.KGen}$, on input the set of clients $\mathcal{C}$ and a security parameter ℓ, performs the following steps:

1. Run $\mathsf{SIGN.KGen}(1^\ell)$ for each client U_i in $\mathcal{C}$ to provide each client with a pair (SK_i, PK_i) of signing/verifying keys;
2. Choose $x \in_R \mathbb{Z}_q^\star$ and set the Server's private/public keys to be: $(SK_S, PK_S) = (x, g^x)$. One denotes $y = g^x$.

Setup The algorithm $\mathsf{GKE.Setup}$, on input a set of client-devices $\mathcal{J}$, performs the following steps (see also Figure 1):

1. Set the wireless client group $\mathcal{G}_c$ to be the input set $\mathcal{J}$.
2. Each client $U_i \in \mathcal{G}_c$ chooses at random a value $x_i \in \mathbb{Z}_q$ and *precomputes* $y_i = g^{x_i}$, $\alpha_i = PK_S^{x_i} = y^{x_i}$ as well as a signature σ_i of y_i, under the private key SK_i.
3. Each client U_i sends (y_i, σ_i) to S.
4. For each $i \in \mathcal{G}_c$, the server S checks the signature σ_i using PK_i, and if they are all correct, computes the values $\alpha_i = y_i^x$.
5. The server S initializes the counter $\mathsf{c} = 0$, as a bit-string of length ℓ_1 and computes the shared secret value:
$$\mathrm{K} = \mathcal{H}_0(\mathsf{c}\|\{\alpha_i\}_{i \in \mathcal{G}_c})$$
and sends to each client U_i the values c and $K_i = \mathrm{K} \oplus \mathcal{H}_1(\mathsf{c}\|\alpha_i)$.

6. Each client U_i (and S) recovers the shared secret value K and the session key sk as described below, and accepts:

$$\mathrm{K} = K_i \oplus \mathcal{H}_1(\mathsf{c} \| \alpha_i) \quad \text{and} \quad sk = \mathcal{H}(\mathrm{K} \| \mathcal{G}_c \| S).$$

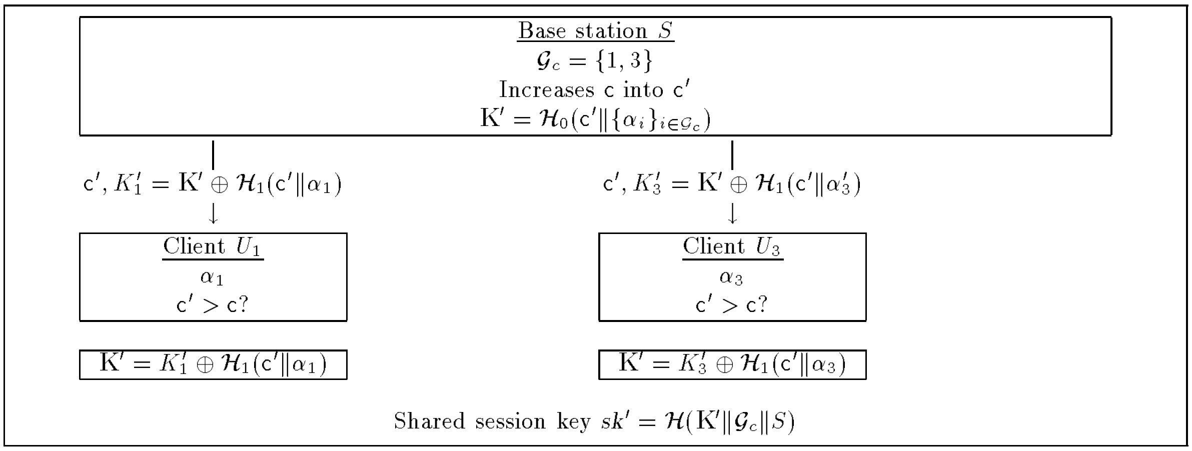

Fig. 2. An execution of the Remove algorithm with the two devices U_2 and U_4 disappearing due to a wireless link failure.

Remove The algorithm GKE.Remove, on input the set $\mathcal{J}$ of disappearing client-devices, performs the following steps (see also Figure 2):

1. Update the wireless client group $\mathcal{G}_c = \mathcal{G}_c \backslash \mathcal{J}$.
2. The server S operates as in the Setup phase. It increases the counter c and computes the shared secret value $\mathrm{K} = \mathcal{H}_0(\mathsf{c} \| \{\alpha_i\}_{i \in \mathcal{G}_c})$.
3. Then it sends to each client $U_i \in \mathcal{G}_c$ the values c and $K_i = \mathrm{K} \oplus \mathcal{H}_1(\mathsf{c} \| \alpha_i)$.
4. Each client $U_i \in \mathcal{G}_c$ already holds the value $\alpha_i = g^{xx_i}$, and the old counter value. So it first checks that the new counter is greater than the old one, and simply recovers the secret shared value K and the session key sk as described below, and accepts:

$$\mathrm{K} = K_i \oplus \mathcal{H}_1(\mathsf{c} \| \alpha_i) \quad \text{and} \quad sk = \mathcal{H}(\mathrm{K} \| \mathcal{G}_c \| S).$$

Join The algorithm GKE.Join, on input the set of appearing client-devices $\mathcal{J}$, performs the following steps (see also Figure 3):

1. Update the wireless client group $\mathcal{G}_c = \mathcal{G}_c \cup \mathcal{J}$.
2. Each appearing client $U_j \in \mathcal{J}$ had chosen at random a value $x_j \in \mathbb{Z}_q$ and *precomputed* $y_j = g^{x_j}$, $\alpha_j = PK_S^{x_j}$ as well as a signature σ_j of y_j, under the private key SK_j.
3. Each appearing client $U_j \in \mathcal{J}$ sends the values (y_j, σ_j) to the device server S.
4. The server S checks the incoming signatures and if correct, operates as in the Setup phase, with an increased counter c: it computes the shared secret value:

$$\mathrm{K} = \mathcal{H}_0(\mathsf{c} \| \{\alpha_i\}_{i \in \mathcal{G}_c}).$$

5. Then it sends to each client $U_i \in \mathcal{G}_c$ the values c and $K_i = \mathrm{K} \oplus \mathcal{H}_1(\mathsf{c} \| \alpha_i)$.
6. Each client $U_i \in \mathcal{G}_c$ already holds the value $\alpha_i = g^{xx_i}$, and the old counter value (set to zero for the new ones). So it first checks that the new counter is greater than the old one, and simply recovers the secret shared value K and the session key sk as described below, and accepts:

$$\mathrm{K} = K_i \oplus \mathcal{H}_1(\mathsf{c} \| \alpha_i) \quad \text{and} \quad sk = \mathcal{H}(\mathrm{K} \| \mathcal{G}_c \| S).$$

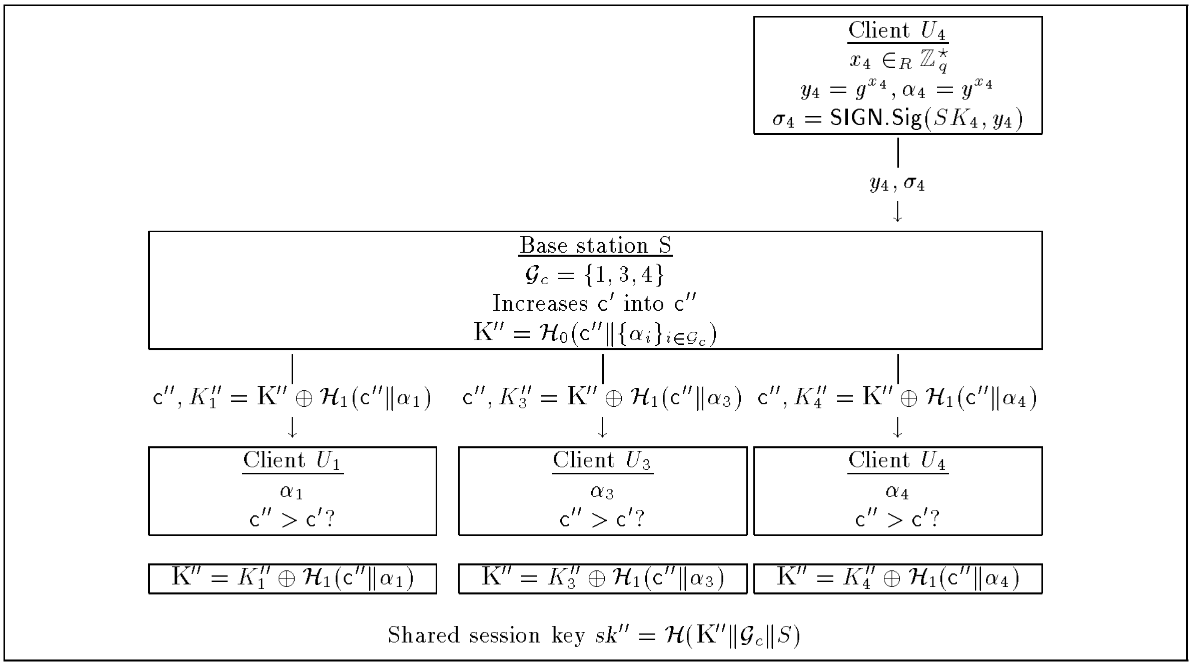

Fig. 3. An execution of the JOIN algorithm with one device (U_4) (re)appearing (using either the same pair (y_4, σ_4) or a new one).

3.3 Pseudo-Random Functions

When engineers choose a protocol for key exchange, they take into its security, computation and communication efficiency, and easy of integration. Since they do not face the same computing environment, they may choose to use a different means to generate the session key. The computation of K with $\mathcal{H}_0(\mathsf{c} \| \{\alpha_i\}_{i \in \mathcal{G}_c})$ is just a way to generate, deterministically, a random string. They may want to implement a version of the above protocol wherein the base station simply draws the value K with any pseudo-random generator. This version would exhibit a similar security result and proof as the one presented in the next section (a security proof is straightforward to derive from the one presented below).

3.4 Efficiency

The protocol presented in this paper is very efficient, since almost everything can be precomputed off-line for the clients, while achieving a strong level of security. The amount of memory available on the clients may provide a trade-off:

– by storing many distinct triples $(y_i, \sigma_i, \alpha_i)$ one increases the security level, but one hashing and one XOR have to be performed on-line;
– by storing many $\mathcal{H}_1(\mathsf{c} \| \alpha_i)$, for each $(y_i, \sigma_i, \alpha_i)$, for several values of the counter, one increases efficiency, since only one XOR has to be performed on-line.

4 Security Analysis

In this section we present the security definitions and show that our protocol achieves them. The security of our protocol is formulated as a function of the amount of resources the adversary expends: the time of computing and the number of queries the adversary makes to the protocol participants.

4.1 Notions of Security

Freshness. The freshness notion captures the intuitive fact that a session key is not "obviously" known to the adversary. A device U is said to be **Fresh**, in the current operation execution, (or holds a **Fresh** sk) if the following two conditions are satisfied. First, nobody in $\mathcal{C}$ has ever been asked for a $\mathsf{Corrupt}$-query from the beginning of the game (during the lifetime of the α_i's). Second, in the current operation execution, U has accepted and neither U nor its partners have been asked for a Reveal-query.

AKE Security. The semantic security of the session key is modeled via an additional query, called the Test-query. This query is only made available if the player is **Fresh**. This can only be asked once during the entire attack. When such a query is asked, a bit b is privately flipped, and the adversary $\mathcal{A}$ gets back either the session key if $b = 1$, or a random string of same length if $b = 0$. When $\mathcal{A}$ terminates, it outputs a single bit b'. Semantic security formally means that $\mathcal{A}$ does not learn *any* information about sk and thus, has no advantage in guessing the bit b. Hence, we define:

$$\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = \left| \Pr_{b'}[b' = 1 | b = 1] - \Pr_{b'}[b' = 1 | b = 0] \right| = 2 \Pr_{b,b'}[b = b'] - 1$$

and say that protocol P is an $\mathcal{A}$-**secure AKE** if $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A})$ is negligible for all probabilistic polynomial time adversary $\mathcal{A}$.

Signature Scheme The security notion for a signature scheme is that it is computationally infeasible for an adversary to produce a valid forgery σ with respect to any message m under a (adaptively) chosen-message attack (CMA). It is (t, q, ϵ)-**CMA-secure** if there is no adversary $\mathcal{A}$ which can get a probability greater than ϵ in mounting an existential forgery under a CMA-attack within time t, after q signing queries. We denote this probability ϵ as $\mathsf{Succ}_{\mathsf{SIGN}}^{\mathsf{cma}}(\mathcal{A})$.

Computational Diffie-Hellman Assumption A (t, ϵ)-CDH attacker in $\mathbb{G}$ is a probabilistic machine Δ running in time t such that

$$\mathsf{Succ}_{\mathbb{G}}^{\mathsf{cdh}}(\Delta) = \Pr_{x_1, x_2} [\Delta(g^{x_1}, g^{x_2}) = g^{x_1 x_2}] \geq \epsilon$$

where the probability is taken over the random values x_1 and x_2. The CDH-Problem is (t, ϵ) intractable if there is no (t, ϵ)-attacker in $\mathbb{G}$. The CDH-assumption states that is the case for all polynomial t and any non-negligible ϵ.

4.2 Our result

The security of our protocol is measured as the probability that an adversary can get some (partial) information on the key. This probability is denoted $\mathsf{Adv}_P^{\mathsf{ake}}$ and depends on the number of messages sent on the network.

Security Theorem. Let $\mathcal{A}$ be an adversary against the Authenticated Key Exchange (AKE) security of our protocol P, making at most q_s active requests, and asking at most q_H queries to the hash oracles ($\mathcal{H}_0$ and $\mathcal{H}_1$). Let N denote the total number of low-power devices. Then we have:

$$\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) \leq 2N \cdot \mathsf{Succ}_{\mathsf{SIGN}}^{\mathsf{cma}}(t, q_s) + 2 q_s q_H \cdot \mathsf{Succ}_{\mathbb{G}}^{\mathsf{cdh}}(t).$$

The above theorem shows that the security of our protocol is based on the intractability of the well-studied computational Diffie-Hellman problem (CDH) and on the security of the signature scheme (CMA) to prevent existential forgeries under adaptive chosen message attacks.

Proof. We incrementally define a sequence of games $\mathbf{G}_0$ through $\mathbf{G}_3$, in which we simulate the protocol and consider the adversary attacking the simulated protocol. The simulation is such that in the last game, $\mathcal{A}$'s advantage is trivially 0. In each game we denote by b the bit involved in the Test-query and by b' the guess output by $\mathcal{A}$. We refer in game $\mathbf{G}_i$ the event S_i as being $b = b'$. We denote by N the size of $\mathcal{C}$ and by q_s the total number of Send-queries asked to the players.

Game $\mathbf{G_0}$. This is the real attack, in which the server is given $y = g^x$, and each client-device is given a pair of signing/verification key, and randomly chooses the x_i's to compute the y_i's. We thus have:

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

Game $\mathbf{G_1}$. We refer to Forge as the event that $\mathcal{A}$ asks for a $\mathsf{SendServer}(m')$-query, such that the verification of the signature is correct and m' was not previously output by a client as an answer to another Send-query. In other words, $\mathcal{A}$ is sending a message it has built by itself, after having seen at most q_s correct signatures (of a specific format).

In that case, we abort the game and fix b' randomly. The games $\mathbf{G}_1$ and $\mathbf{G}_0$ are identical as long as Forge does not occur. By guessing the impersonated client, one easily gets:

$$|\Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_0]| \leq \Pr[\mathsf{Forge}] \leq N \cdot \mathsf{Succ}^{\mathsf{cma}}_{\mathsf{SIGN}}(t, q_s).$$

Game $\mathbf{G_2}$. In that game, we are given a Diffie-Hellman triple $(A = g^\alpha, B = g^\beta, C = g^\gamma)$ with the values α and β (and thus $\gamma = \alpha\beta \bmod q$), and define $x \leftarrow \alpha$, $y \leftarrow A = g^\alpha$. Furthermore, any random exponent x_i is defined by $\beta + \delta_i \bmod q$, and $y_i \leftarrow Bg^{\delta_i}$. As a consequence, α_i is set to CA^{δ_i}. This simulation is still perfect, as soon as a new random δ_i is drawn for any new x_i:

$$\Pr[\mathsf{S}_2] = \Pr[\mathsf{S}_1].$$

Game $\mathbf{G_3}$. In this game, we do exactly as above, except that any hash value involving an α_i (either $\mathcal{H}_0(\mathsf{c}\|\{\alpha_i\}_{i\in\mathcal{G}_c})$ or $\mathcal{H}_1(\mathsf{c}\|\alpha_i)$), asked by the players or the server are answered independently from the random oracles. Since the same hash queries, asked by the adversary, are still answered by querying the random oracles, some inconsistency may occur. Such an inconsistency is discovered by the adversary if such a hash query is asked by the adversary, event which we denote by AskH:

$$|\Pr[\mathsf{S}_3] - \Pr[\mathsf{S}_2]| \leq \Pr[\mathsf{AskH}].$$

Such an event AskH means that some α_i (among at most q_s, since at most q_s valid signatures have been produced, and thus at most q_s values for y_i's which each leads to one α_i) appears in the list of the hash queries. Since we do not need anymore α, β and C either for the simulation (they were just required in Game $\mathbf{G}_2$ for simulating K and the K_i), we are now just given A and B. By guessing the α_i instance (and thus the δ_i) that has been asked by the adversary, and the corresponding hash-query (and thus the α_i), one extracts $C = \alpha_i A^{-\delta_i}$:

$$\Pr[\mathsf{AskH}] \leq q_H q_s \times \mathsf{Succ}^{\mathsf{cdh}}_{\mathbb{G}}(t).$$

In this last game, since none of $\mathcal{H}_0(\mathsf{c}\|\{\alpha_i\}_{i\in\mathcal{G}_c})$ or $\mathcal{H}_1(\mathsf{c}\|\alpha_i)$ is used more than once, because of the incremental counter, that has to be checked by the players before accepting, the advantage of any adversary is exactly 0. The adversary has indeed no information about any K, and thus about any session key. This concludes the proof. □

Note that with specific groups, and specific signature schemes, this security result can be improved:

- In the random oracle model [3], a model that we already assume, many signatures can be simulated (such as the Schnorr's signature [21, 22], that also relies on the discrete logarithm problem [18]), and thus no private key needs to be known when simulating participants. We can thus suppress the factor N.
- In some groups, the decisional Diffie-Hellman problem is easy (e.g. in weak curves, granted Weil pairing), then $\Pr[\mathsf{AskH}]$ can be shown less than $\mathsf{Succ}^{\mathsf{cdh}}_{\mathbb{G}}(t + q_H q_s \mathcal{O}(1))$.

As a consequence, in specific environments, the security result becomes

Theorem 1. *Let $\mathcal{A}$ be an adversary against the AKE security of our protocol P, asking at most q_s Send-queries and q_H queries to the hash oracles. Let N denote the total number of low power devices. Then we have:*

$$\mathsf{Adv}^{\mathsf{ake}}_{P}(\mathcal{A}) \leq 2 \cdot \mathsf{Succ}^{\mathsf{cma}}_{\mathsf{SIGN}}(t, q_s) + 2 \cdot \mathsf{Succ}^{\mathsf{cdh}}_{\mathbb{G}}(t + q_s q_H \mathcal{O}(1)).$$

5 Mutual Authentication and (Partial) Forward Secrecy

U_i		U_j
	Protocol GKE which outputs	
comes up with $\mathrm{K} = K_{U_i}$		comes up with $\mathrm{K} = K_{U_j}$
		$\mathsf{Auth}_{U_j} \leftarrow \mathcal{H}(K_{U_j} \Vert j)$
		$sk_{U_j} \leftarrow \mathcal{H}(K_{U_j} \Vert 0 \Vert \mathcal{G}_c \Vert S)$
	$\xleftarrow{auth_{U_j}}$	
$\mathsf{Auth}_{U_j} \stackrel{?}{=} \mathcal{H}(K_{U_i} \Vert j)$		
$sk_{U_i} \leftarrow \mathcal{H}(K_{U_i} \Vert 0 \Vert \mathcal{G}_c \Vert S)$		

Fig. 4. U_j-to-U_i authentication in GKE. The shared session key is $sk = \mathcal{H}(\mathrm{K} \Vert 0 \Vert \mathcal{G}_c \Vert S)$.

Mutual authentication ensures each player that all other parties did actually compute the same key. Our protocol can be modified to achieve this goal. The modification presented on Figure 4 requires that each low-power device computes N hashings and sends one flow to the server S. This computational overhead is tolerable only if N does not get too large, but for larger values of N this overhead can also be kept to a minimum by performing mutual authentication through the server. Each client authenticates to the server which then in turn authenticates to each client only after all clients have been authenticated. This approach has the attractive advantage of being not only provably secure, in the random-oracle model, but to also add little overhead to the original protocol.

About forward-secrecy, it is clear that as soon as the long-term key x of the server is leaked, all the session keys can be recovered, since all the α_i can easily be computed from the y_i and x. Therefore, no forward-secrecy exists when the server long-term key is revealed. However, the long-term keys of the low-power devices (the signing keys) are used for implicit authentication only, and not for hiding the session key. Therefore, the leakage of clients' long-term keys do not reveal anything about previous session keys. Furthermore, strong (partial) forward-secrecy (where any internal data is revealed in case of corruption, *i.e.* the signing key, but also the x_i, y_i

and α_i) is also achieved if the x_i's and α_i's are erased as soon as they are not useful (the client has left from the group). As a consequence, no information about previous session keys can be found in the memory of the low-power devices. We claim these considerations make sense since the server can be reasonably seen as more reliable that the mobile devices.

Related Work. The question whether public-key cryptographic technology could be implemented on low-power end devices has been addressed in the context of session-key establishment [15, 16, 24–26] and signing on a (contact-free) smart-card[13, 14, 19, 23]. For example, the public-based variant of the Kerberos protocol has been adapted to run on a low-power computing devices [24], and methods for converting any traditional signature schemes into an efficient on-line/off-line one have emerged [13, 23]. The current *de facto* standard for securing the electronic transactions ("e-Commerce") between a client and a server over the Internet uses elliptic curve cipher suites to run on low-power devices [5], and has evolved into a protocol to secure "m-Commerce" [20].

Attempts to design secure two-party key-establishment protocols for the mobile environment have been made [7, 16, 24–26], but despite the apparent simplicity of designing such protocols many proposed schemes were later found to be flawed. A first attempt to design key exchange protocols for a cluster of mobiles was made by Asokan et al. [1] in the context of *ad hoc* mobile wireless networks. Their protocols are suited when a small group of powerful mobile devices, like laptops, get together but they become impractical when low-power devices come into play.

One way to avoid many of the flaws in constructing cryptographic protocols is to design in the framework of provable security. The work of Bresson et al. [9] provides a useful formal model to start from in designing a provably-secure dynamic group-key-agreement protocols, based on a public-key infrastructure. The password-based protocol they provide in [10] which would be well-suited for *ad hoc* networks is too costly to be a practical solution for heterogeneous mobiles. A first step towards the low-power devices is an extension of the Jakobsson and Pointcheval [16] 2-party protocol to groups: an efficient, elegant, and provably secure key exchange protocol for groups of mobile devices.

6 Conclusion

In this paper we presented an efficient key agreement protocol for heterogeneous wireless networks. Our protocol allows a set of heterogeneous mobiles devices to form a secure group and to handle the continuous disappearing and reappearing of mobiles due to communication failures. Our protocol has been proved secure in the random oracle model under the computational Diffie-Hellman assumption.

References

1. N. Asokan and P. Ginzboorg. Key Agreement in Ad-hoc Networks, February 2000. Expanded version of a talk given at the Nordsec '99 workshop.
2. G. Ateniese, M. Steiner, and G. Tsudik. New Multiparty Authentication Services and Key Agreement Protocols. *IEEE J. of Selected Areas in Communications*, Apr. 2000.
3. M. Bellare and P. Rogaway. Entity Authentification and Key Distribution. In *Crypto '93*, LNCS 773, pp. 232–249. Springer, 1993.
4. K. Berket, P. M. Melliar-Smith, and L. E. Moser. The InterGroup Protocols: Scalable Group Communication for the Internet. *3rd Global Internet Mini-Conference*, Nov. 1998.
5. S. Blake-Wilson, V. Gupta, C. Hawk, and B. Moeller. ECC Cipher Suites for TLS. RFC 20296, Feb. 2002.
6. N. Borisov, I. Goldberg, and D. Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. In *ACM MobiCom'01*, 2001.
7. C. Boyd and A. Mathuria. Key Establishment Protocols for Secure Mobile Communications: A Selective Survey. In *ACISP '98*, LNCS 1438, pp. 344–355. Springer, 1998.

8. E. Bresson, O. Chevassut, and D. Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange – The Dynamic Case. In *Asiacrypt '01*, LNCS 2248, pp. 290–309. Springer, Dec 2001.
9. E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In *Eurocrypt '02*, LNCS 2332, pp. 321–336. Springer, May 2002.
10. E. Bresson, O. Chevassut, and D. Pointcheval. Group Diffie-Hellman Key Exchange Secure Against Dictionary Attacks. In *Asiacrypt '02*, LNCS 2501, pp. 497-514. Springer, Dec 2002.
11. E. Bresson, O. Chevassut, A. Essiari, and D. Pointcheval. Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices. In *5th IEEE MWCN*. IEEE, Oct 2003. Full version available from http://www.di.ens.fr/users/pointche.
12. G. D. Crescenzo and O. Kornievskaia. Efficient Kerberized Multicast in a Practical Distributed Setting. In *Information Security Conference (ISC'01)*. Oct 2001
13. S. Even, O. Goldreich, and S. Micali. On-line/Off-line Digital Signatures. In *Crypto '89*, LNCS 435, pp. 263–277, Springer, Aug 1989.
14. M. Girault. Self-Certified Public Keys. In *Eurocrypt '91*, LNCS 547, pp. 490–497. Springer, May 1992.
15. A. Harbitter and D. A. Menace. The Performance of Public Key-Enabled Kerberos Authentication in Mobile Computing Applications. In *ACM CCS'01*, pp. 78–85.
16. M. Jakobsson and D. Pointcheval. Mutual Authentication for Low-Power Mobile Devices. In *Financial Crypto '01*, LNCS 2339, pp. 178–195. Springer, 2001.
17. T. Phan, L. Huang, and C. Dulan. Challenge: Integrating Mobile Wireless Devices Into the Computational Grid. In *MobiCom '02*, pp. 271–278, 2002.
18. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *J. of Cryptology*, 13(3):361–396, 2000.
19. G. Poupard and J. Stern. Security Analysis of a Practical "on the fly" Authentification and Signature Generation. In *Eurocrypt '98*, LNCS 1403, pp. 422–436. Springer, May 1998.
20. Wireless Application Protocol. Wireless Transport Layer Security Specification. WAP TLS, WAP-199 WTLS, February 2000.
21. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto '89*, LNCS 435, pp. 235–251. Springer, 1990.
22. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *J. of Cryptology*, 4(3):161–174, 1991.
23. A. Shamir and Y. Tauman. Improved Online/Offline Signature Schemes. In *Crypto '01*, LNCS 2139, pp. 355–367. Springer, Aug 2001.
24. B. Tung and C. Neuman and M. Hur and A. Medvinsky and S. Medvinsky and J. Wray. Public-Key Cryptography for Initial Authentication in Kerberos. http://www.ietf.org/internet/drafts/draft-ietf-cat-kerberos-pk-init-12.txt, 2001.
25. D. S. Wong and A. H. Chan. Efficient and Mutually Authenticated Key Exchange for Low Power Computing Devices. In *Asiacrypt '01*, LNCS 2248, pp. 272–289. Springer, 2001.
26. F. Zhu, A. H. Chan, D. S. Wong, and R. Ye. Password Authenticated Key Exchange based on RSA for Imbalanced Wireless Network. In *ISC '02*, LNCS 2433, pp. 150–161. Springer, 2002.