Ephemeral Pairing on Anonymous Networks*

Jaap-Henk Hoepman

Nijmegen Institute for Computing and Information Sciences (NIII)
Radboud University Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, the Netherlands
jhh@cs.ru.nl

Abstract. The *ephemeral pairing problem* requires two or more specific physical nodes in a wireless broadcast network, that do not yet know each other, to establish a short-term relationship between them. Such short-lived pairings occur, for example, when one pays at a check-out using a wireless wallet. This problem is equivalent to the *ephemeral key exchange* problem, where one needs to establish a high-entropy shared session key between two nodes given only a low bandwidth authentic (or private) communication channel between the pair, and a high bandwidth shared broadcast channel.
We study this problem for truly anonymous broadcast networks, discuss certain impossible scenarios and present several protocols depending on the type of communication channel between the nodes.

Keywords: *Authentication, identification, pairing, key exchange, anonymous networks.*

1 Introduction

The *ephemeral pairing problem* (introduced in [Hoe04]) consists of establishing a short-term relationship between two or more specific physical nodes in a wireless broadcast network that do not yet know each other. Ephemeral pairings occur, for example, when one pays at a check-out using a wireless wallet. As opposed to paying with a smart card by inserting it into a specific terminal, using a wireless connection (like Bluetooth[1] or IrDA[2]) gives no guarantee that two physical nodes that want to communicate with each other are actually talking to each other. Without any countermeasures one might end up paying for the customer at the check-out next to you.

To achieve such short-lived pairings, we do not wish to rely on any secret information shared a priori among the nodes. For the large scale systems where we expect the ephemeral pairings to play a part, such a secure initialisation might be costly and carry a huge organisational burden. Instead, we allow the nodes in the system to exchange *small* amounts of information reliably and/or privately.

* Id: pairing.tex,v 2.6 2005/01/05 13:13:33 jhh Exp
[1] See http://www.bluetooth.com.
[2] See http://www.irda.org.

Two typical application scenarios may help to understand the issues involved.

Consider for example the case where someone wishes to pay using a wireless wallet at a checkout counter of a large supermarket. There are many checkouts, and many customers paying simultaneously. To pair the wallet with the right checkout counter, the counter could generate a small random number every time a new customer arrives, and show this number on its display. Subsequently, the customer enters this number on his wallet to initiate the pairing. In this case, the customer knows the source of the random number and uses the number displayed by the counter at which it wants to pay. In other words, the random number is *authentic*. However, eavesdroppers may also be able to read the number from the display, and hence the number is not *private*. If eavesdropping is made impossible, the communication is both authentic and private.

As an example for the private case[3] consider the following. Instead of a display at each counter, the supermarket installs a single ticket dispenser ahead of all the counters (similar to systems used to assign waiting numbers to customers in e.g., a large post office). The ticket dispenser provides the customer with a ticket on which the random number is printed, together with the number of the counter to pay at. Authenticity of the information cannot be assumed, for instance because attackers may reinsert old or forged tickets in the machine. However, the information can be considered private (provided the user does not drop the ticket on the floor right after entering it on his wallet).

Because the devices are human operated, and the operators are involved in the exchange of the information, the numbers of bits that can be transferred is low (comparable to the size of typical passwords), and certainly much less than the number of bits required for strong cryptographic keys. Therefore, one cannot expect to be able to use such private or authentic communication mechanisms (called channels from now) to establish cryptographic keys directly. We do note that typically the numbers transferred over these channels are machine generated. Several realistic methods for implementing such private or authentic low bandwidth channels exist [Hoe04].

In more abstract terms then, this problem can be phrased as an *ephemeral key exchange* (denoted by φKE) problem: given a low bandwidth authentic (or private) communication channel between two nodes, and a high bandwidth broadcast channel, can we establish a high-entropy shared secret session key between the two nodes without relying on any a priori shared secret information? Here, the low bandwidth channel models the (implicit) authentication and limited information processing capabilities of the users operating the nodes.

There are numerous applications that require a solution to the ephemeral pairing problem, e.g., connecting two laptops in a business meeting, exchanging electronic business cards using PDAs, buying electronic tickets at a box office and checking them at a venue, unlocking doors using a wireless token (making sure the right door is unlocked), etc. In most of these applications, indeed by the

[3] This example is admittedly slightly more contrived — indeed authentic communication appear to occur more naturally in real applications.

very nature of the problem, the cooperation of the user/owner of the device is required to successfully establish a pairing. For instance, the user may be asked to select a pattern from a list that corresponds to the pattern shown on the remote device. Alternatively, the user may be asked to copy and enter a passcode. We stress that in all these cases, the number of bits that can be handled by the user is very limited and that these bits do not have enough entropy to secure the communications directly. This is the main motivation for the model and the problem statement of establishing a high bandwidth secure communication channel between two nodes using only a low bandwidth authentic/private channel.

1.1 State of the art

Stajano and Anderson [SA99] introduced the (long-lived) pairing problem that occurs when separate devices need to establish a long term relationship that allow one of the devices to exert control over the other (e.g., a remote control and the corresponding TV set). These pairings are supposed to exists over prolonged periods of time, and therefore the setup of such a pairing is allowed to be quite involved.

In [Hoe04] it was shown that solutions to the ephemeral pairing problem can sometimes be based on Encrypted Key Exchange (EKE) [BM92] protocols, suggesting a relationship between these two problems. An extended discussion on this, and a review of the state of the art regarding EKE is also presented there. The solutions of [Hoe04] only apply to non-anonymous broadcast networks.

In this paper we study ephemeral pairing on anonymous broadcast networks. The difference between the anonymous and non-anonymous case is the following. In the non-anonymous case, participants only need to receive messages from a identified sender. If no such message is received, or if the adversary forged a sender identity (which is detected in subsequent stages of the protocol), the protocol simply aborts. In the anonymous case, participants may receive many messages, and it is not a priori clear which messages are intended for them. Therefore, they have to collect all messages they receive, and based on their content decide which message to accept (if any). This influences the design of the protocols. One could, in principle, use the 'secure' point-to-point channel for coordinating a session identifier directly, but in order to save this limited resource only for the purpose of key-exchange we choose not to do so. We note that the power of the adversary is the same in both models (it can arbitrarily change the source and destination in the non-anonymous model, and doesn't have to in the anonymous case).

Balfanz *et al.* [BSSW02] study essentially the same problem, but assume that the low-bandwidth communication channel is large enough to pre-authenticate public keys (either by sending whole keys, or hashes of these keys), that can subsequently be used in a standard public key authentication protocol. Gehrmann *et al.* [GMN04, GN04] describe the ISO/IEC standards for manual authentication of wireless devices, that allow for smaller bandwidth on the communication channel, but require the channel to be private.

A more rigorous and formal treatment of the security of EKE protocols was initiated by Lucks [Luc97], and expanded on by several authors [BMP00, BPR00, Sho99, CK01, GL03]. Due to space constraints the security proofs in this paper are informal, but will be based on Bellare *et al.* [BPR00] in the full version.

1.2 Summary of results

In this paper we present several ephemeral key exchange protocols for completely anonymous broadcast networks, for different combinations of the point-to-point communications channels between the two nodes. These are presented in Sect. 4. Before that, we describe the model in Sect. 2, and present some impossibility results for certain types of point-to-point channels in Sect. 3. We conclude with directions for further research in Sect. 5.

2 Model and problem statement

Consider n physically identifiable nodes communicating over a public and insecure broadcast network, each attended by a human operator. The operators (and/or the nodes they operate) can only exchange *small* amounts of information reliably and/or in private. The ephemeral pairing problem requires two or more nodes (to be determined by their operators) to securely establish a shared secret.

As discussed in [Hoe04], this problem can be seen in more abstract terms as an ephemeral key exchange (φKE) problem. Consider Alice and Bob, connected through a high bandwidth broadcast network. In this paper, the broadcast network is completely anonymous. Alice and Bob also share a low bandwidth communication channel over which they can exchange at most η bits of information per message. This channel is either

authentic, meaning that Bob is guaranteed that a message he receives actually was sent by Alice (but this message may be eavesdropped by others), or

private, meaning that Alice is guaranteed that the message she sends is only received by Bob (but Bob does not know the message comes from Alice).

Given these connections, Alice and Bob are required to establish an authenticated and shared σ bits secret (where $\sigma \gg \eta$). They do not share any secrets a priori, and do not have any means to authenticate each other, except through the low bandwidth channel.

The adversary may eavesdrop, insert and modify packets on the broadcast network, and may eavesdrop on the authentic channel or insert and modify packets on the private channel. Note that, by assumption, the adversary cannot insert or modify packets on the authentic channel. Also, the adversary may subvert any number of nodes and collect all the secret information stored there.

Security of our protocols is defined as in the encrypted key exchange model developed by Bellare *et al.* [BPR00], where the adversary is given the task to

distinguish an actual session key from an arbitrary random value for any instance of the protocol run of his choice.

In our analysis we will bound the advantage of the adversary for a particular protocol using s, t and the number of active attacks (denoted by q) performed by the adversary. Here, s and t are the security parameters of the φKE protocol. s roughly corresponds to the size of the session key to be established, and determines the advantage of a passive adversary. t roughly corresponds to the capacity of the channel between two principals, and mostly determines the advantage of an active adversary. Actually, q corresponds to the number of instances that are attacked actively by the adversary (and that involve one or more message insertions or modifications).

We work in the random oracle model, and assume hardness of the Computational Diffie Hellman problem.

We use the following notation throughout the paper. In the description of the protocols, ac is the authentic channel, pc is the private channel, and bc is the broadcast channel. Assignment is denoted by := Receiving messages from the channel or the broadcast network can be done in a blocking fashion (indicated by **receive**) or in a non-blocking fashion (indicated by **on receiving**).

In message flowcharts, $\xrightarrow{m}$ denotes sending m on the private or authentic channel, while $\overset{m}{\Longrightarrow}$ denotes broadcasting m on the broadcast channel. The receiving party puts the message in the indicated variable v at the arrowhead.

3 Impossibility results

In this section we show a few straightforward impossibility results. The φKE problem[4] cannot be solved using only a single uni-directional point-to-point channel between Alice and Bob that is either authentic or private. Even an authentic channel from Alice to Bob and a private channel from Bob to Alice is not strong enough. An authentic channel from Alice to Bob and another private channel from Alice to Bob *is* strong enough however (even though there is no point-to-point channel from Bob to Alice, see Sect.4.3).

Theorem 3.1. *The φKE problem cannot be solved using a single private channel from Alice to Bob and a single authentic channel from Bob to Alice.*

Proof. Suppose there is a protocol solving the φKE problem using a single private channel from Alice to Bob and a single authentic channel from Bob to Alice. Now, instead of Alice, let the adversary start a session with Bob. Because Alice and Bob do not a priori share any secret information, and because the adversary can use the private channel to Bob and the authentic channel from Bob in exactly the same way as Alice does, Bob cannot distinguish the adversary from Alice in this session. This contradicts the requirement that at the end of the session Alice and Bob share a secret session key.

[4] To be more precise, the following results only hold for the general (two-sided) version of the problem, not the one-sided version (see [Hoe04] for the difference between these two).

The following two facts are easy corollaries of this theorem.

Corollary 3.2. *The φKE problem cannot be solved using a single private channel from Alice to Bob.*

Corollary 3.3. *The φKE problem cannot be solved using a single authentic channel from Bob to Alice.*

4 φKE protocols for anonymous networks

In this section we present three φKE protocols. In the first protocol Alice and Bob are connected by a bidirectional private channel. The second protocol covers the case where Alice and Bob are connected by a bidirectional authentic channel. In the third protocol there are two channels, one authentic and the other private, both running from Alice to Bob. All protocols assume an anonymous broadcast network.

In all protocols, Alice and Bob are required to generate a (small entropy) password to be exchanged over the low bandwidth communication channel. We stress that this password is machine generated, at random, at the start of each protocol run, and hence that these passwords can be assumed to statistically independent. It is only the transmission of these passwords over the low bandwidth channel that requires (in the application of this model in a practical setting) human intervention. It is also this same human handling of these passwords that requires them to have only a small number of bits.

The main problem handling an anonymous broadcast network is to ensure that a participant in the protocol can immediately reject messages that are obviously not intended for it. Without such precautions, even honest but ignorant nodes can easily disrupt the protocol through the messages they themselves legitimately send over the broadcast network. The protocols to be presented next try to derive a common session identifier as soon as possible, to be used as a header on messages on the broadcast channel. Note that simply transmitting such session identifiers on the point-to-point communication channel is not a good option, as it wastes bits to be used for authenticating the shared session key.

In all protocols we use the following. G is a group of order at least 2^{2s} with generator g for which the Computational Diffie Hellman (DDH) problem is hard. A possible candidate is the subgroup of order q in $\mathbb{Z}_p^*$ for p, q prime and $p = 2q + 1$. Naturally, exponentiations like g^x are computed in the group G.

Passwords are selected uniformly at random from a set P, of size 2^t.

Furthermore, we use several hash functions h_i with varying domains and ranges, which are modelled as random oracles. The domain and range of hash functions h_1, h_2 is specified for each protocol separately. All protocols use $h_3, h_4, h_5 : G \mapsto \{0,1\}^\sigma$, and $h_6 : G \mapsto I$ (where I is a suitably large session identifier set). We use the following property

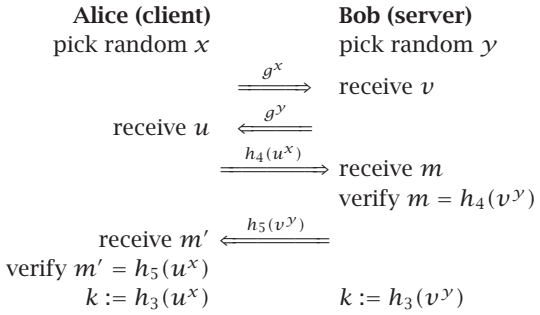

Fig. 1. Diffie-Hellman key exchange with key validation.

Property 4.1. Let G, A and B be groups. Let X be a uniformly distributed random variable over G, let $h : G \mapsto A$ and $h' : G \mapsto B$ be random oracles and let $a \in A$ and $b \in B$ be arbitrary. Then

$$\mathbf{Pr}\left[h(X) = a \,|\, h'(X) = b\right] = \mathbf{Pr}\left[h(X) = a\right] = 2^{|A|}\,.$$

We write bc_s for the broadcast channel restricted to only carry messages with session identifier s: if a messages m is received from bc_s, it was sent with that session identifier[5].

Consider the Diffie-Hellman key exchange with validation[6] in Fig. 1, with G of order at least 2^{2s} and h_3, h_4, and h_5 as defined above. Then under the assumption that the Computational Diffie Hellman problem over G is hard we have [BPR00, Sho99]

Proposition 4.2. *The advantage of any adversary attacking the Diffie-Hellman key exchange with key validation in Fig. 1 — i.e., distinguishing $h_3(g^{ab})$ from a random element of $\{0,1\}^{2s}$, when given $g^a, g^b, h_4(g^{ab}), h_5(g^{qb})$ — is at most $O(2^{-s})$.*

4.1 φKE for a bidirectional private channel

The φKE protocol for a bidirectional private channel (see Prot. 4.1 for the protocol and Fig. 2 for the corresponding message exchange graph) proceeds in five phases: authenticate, commit, synchronise, exchange and validate. In the exchange phase, Alice and Bob use a Diffie-Hellman type key exchange [DH76] to establish a shared session key. This key is then used to derive a session

[5] We stress that this doer *not* guarantee that only an honest party generated this message: s is public, and hence the adversary can generate messages with that identifier as well.

[6] The validation phase only prevents a man-in-the-middle attack if somehow the shares g^x and/or g^y are authenticated. This principle is used in our protocols later on.

Authenticate
$H := \emptyset$
pick random x
pick random p
send p **on** pc
receive π **from** pc
Commit
broadcast $h_1(g^x, \pi)$ **on** bc
for c milliseconds **do**
on receiving m **from** bc **do**
if $|H| < z$
then $H := H + \{m\}$
abort if $|H| = \emptyset$
Synchronise
broadcast p **on** bc
receive q **from** bc
abort if $q \neq \pi$

Key exchange
$u := \perp$
broadcast g^x **on** bc
for c milliseconds **do**
on receiving m **from** bc **do**
if $h_1(m, p) \in H$ **then** $u := m$
abort if $u = \perp$
Key validation
$\iota := h_6(u^x)$
$k := \perp$
$j := \begin{cases} 0 & \text{if client} \\ 1 & \text{if server} \end{cases}$
broadcast $h_{4+j}(u^x)$ **on** bc_ι
receive m **from** bc_ι
if $h_{5-j}(u^x) = m$
then $k = h_3(u^x)$
else abort

Protocol 4.1: φKE for bidirectional private channel.

identifier ι (to distinguish relevant messages on the broadcast network)[7]. Both parties engage in a verification phase to ensure that they share the same session key [BPR00].

To identify and authenticate the share sent by the other party, Alice and Bob exchange random passwords through the bidirectional private channel in the authenticate phase. Alice hashes the password received together with her own share, and broadcasts the resulting hash on the network, to commit to the value of the share to be exchanged later on. They use the hash function $h_1 : G \times P \mapsto G$ in this phase.

The commit and exchange phase have to be separate, or else the adversary can still perform a dictionary attack to retrieve the exchanged password and substitute a suitable share of his own choice in both the commitment (that also serves to authenticate the share) and the share itself. Because Alice and Bob cannot reliably setup a session identifier at or before the exchange phase, Bob cannot distinguish Alice's commitment from other commitments broadcast on the network. Therefore, Bob accepts at most z of them.

To separate the commit and exchange phase, and to ensure that no commits will be accepted when the other party starts the key exchange phase, a separate synchronisation phase is introduced. In this phase, both parties reveal their password over the broadcast channel, and the other party only starts the exchange phase if it receives the same password as in the authentication phase.

[7] Note that it is not possible to use $h_6(\pi)$ (with π as exchanged through the private channel) as a session identifier: this session identifier could then be used to verify all tries for the password in a dictionary attack.

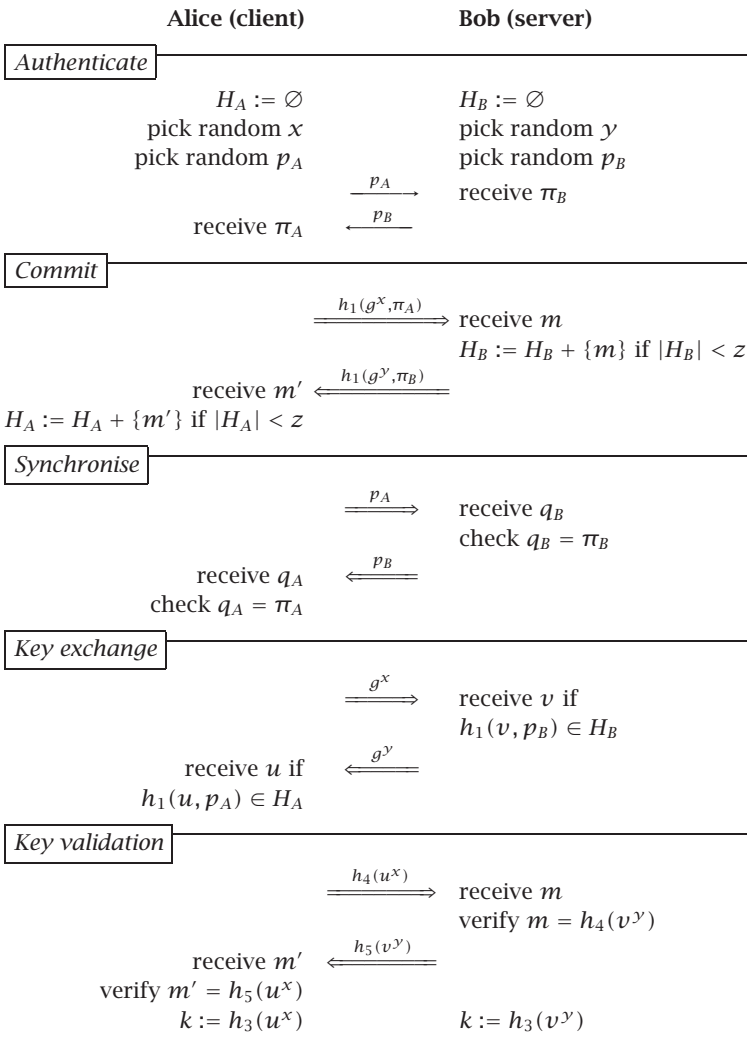

Fig. 2. Message flow of φKE for bidirectional private channel.

Observe that a password has lost all value once the commit phase for which it is used as authenticator is closed.

In the key exchange phase, Bob only accepts a share that together with Bob's own password hashes to a value in the set of commitments received previously. Because Bob only accepts z commitments, an active adversary may plant at most z commitments for its *own* share (using z different guesses for the password sent by Alice), thus limiting its chances to attack the protocol. For all our protocols a good value for z is one that allows some honest concurrent activity on the broadcast channel, while still limiting the advantage of the adversary.

Obviously, by limiting the number of commitments that Bob accepts, we allow the adversary to preempt the protocol by filling all commitment slots with garbage. This prevents Alice from successfully pairing with Bob. However, as this is but one of the many possible (and easier) denial of service attacks, we will not consider this issue further here.

Analysis It is easily shown that using Prot. 4.1, honest Alice and Bob can exchange a shared session key. Next, we prove security of the protocol in the presence of an active and/or passive adversary.

Theorem 4.3. *The advantage of an adversary attacking Prot. 4.1 mounting at most q active attacks is at most*

$$O(1 - e^{-zq/2^t}) + O(2^{-s}) .$$

Proof. We split the proof in two cases. We first consider the case where the session key k generated by an oracle[8] is not based on a share g^a sent by the adversary, but instead derived from a value x of his own choosing, and then consider the case where the adversary manages to convince the oracle to use such a share of his own choosing.

If the session key generated by an oracle is not based on a share g^a sent by the adversary, but instead derived from a value x of his own choosing, then k depends on private random values x, y unobserved by the adversary and publicly exchanged shares g^x and g^y using a Diffie-Hellman (DH) key exchange. Any adversary attacking Prot. 4.1 can be converted to an adversary attacking a DH key exchange with validation (see Prop. 4.2) as follows. Given a run over the basic DH key exchange, generate random passwords p_A and p_B, and insert $h_1(g^x, p_A)$ and $h_1(g^y, p_B)$, p_A and p_B at the appropriate places in the run of the DH key exchange with validation before analysing the run. Hence the advantage of the adversary to distinguish the session key cannot be higher than its advantage in breaking the Diffie-Hellman key exchange with validation, which is at most $O(2^{-s})$ by Prop. 4.2.

In the other case, in order to convince an oracle of A to use the share g^a of the adversary in the second phase of the protocol, the adversary must ensure that $h_1(g^a, p_A) \in H_A$ for values H_A, p_A used in this oracle A. Note that due to the properties of the private channel, p_A is unknown to the adversary. Hence the adversary has probability $2^{-\eta}$ to guess it right and authenticate its own share g^a using $h_1(g^a, p)$ in the commit phase. As $|H_A| \leq z$, the adversary can try at most z different values for p. Hence the total probability that a share of the adversary is accepted is at most $z2^{-\eta}$.

For each active attack then the probability of success is $z2^{-\eta}$. Success with one instance is independent of success in any other instance. Hence, with q attempts, the probability of success becomes (cf. [Fel57])

$$1 - (1 - z2^{-\eta})^q \approx 1 - e^{-z2^{-\eta}q}$$

[8] In the Bellare *et al.* [BPR00] model, the participants in the protocol are modelled as oracles to which the adversary has access. We use the same terminology here.

Commit
$H := \emptyset$
pick random x
broadcast $h_1(g^x)$ **on** bc
for c milliseconds **do**
on receiving m **from** bc **do**
if $|H| < z$
then $H := H + \{m\}$
send close **on** ac
receive close **from** ac
abort if $|H| = \emptyset$
Authenticate
send $h_2(g^x)$ **on** ac
receive β **from** ac
$\iota := \beta$
$\omega := h_2(g^x)$

Key exchange
$u := \perp$
broadcast g^x **on** bc_ω
receive m **from** bc_ι
if $h_1(m) \in H$ and $h_2(m) = \beta$
then $u := m$
else abort
Key validation
$k := \perp$
$j := \begin{cases} 0 & \text{if client} \\ 1 & \text{if server} \end{cases}$
broadcast $h_{4+j}(u^x)$ **on** bc_ω
receive m **from** bc_ι
if $h_{5-j}(u^x) = m$
then $k = h_3(u^x)$
else abort

Protocol 4.2: Anonymous φKE for bidirectional authentic channel.

With $t = \eta$ this proves the theorem. □

4.2 φKE for a bidirectional authentic channel

This protocol (see Prot. 4.2 and Fig. 3) again proceeds in four phases: commit, authenticate, exchange and validate (but with the first and second phase changing order). The exchange and verification phase are essentially equal to that of the previous protocol, except that Alice and Bob use session identifiers ι and ω derived from the authentication messages exchanged earlier.

To avoid man-in-the-middle attacks, the shares used in the key exchange phase must be authenticated. However, the capacity of the authentic channel is too small to do so directly. Instead, it is used to authenticate a small hash of the share to be used later on. This is not enough to ensure security: the adversary can trivially (in an expected $2^{\eta-1}$ number of tries) find a share of his own that matches the authenticator that will be sent by Alice. Therefore, Alice and Bob must first commit to a share using a much larger hash value (against which it is infeasible to find collisions) but that has to be sent over the broadcast channel. The hash functions used for this are $h_1 : G \mapsto G$ and $h_2 : G \mapsto \{0,1\}^\eta$.

But even then a problem remains. For the adversary may try to commit to 2^η shares, corresponding to a specific authenticator value[9]. After the commit phase, when Alice reveals the authenticator a, the adversary simply reveals the share

[9] The adversary can easily do this by spending an expected $2^{\eta-1}$ amount of work to find a candidate for each possible authenticator value, thus using a total expected $2^{2\eta-1}$ amount of work.

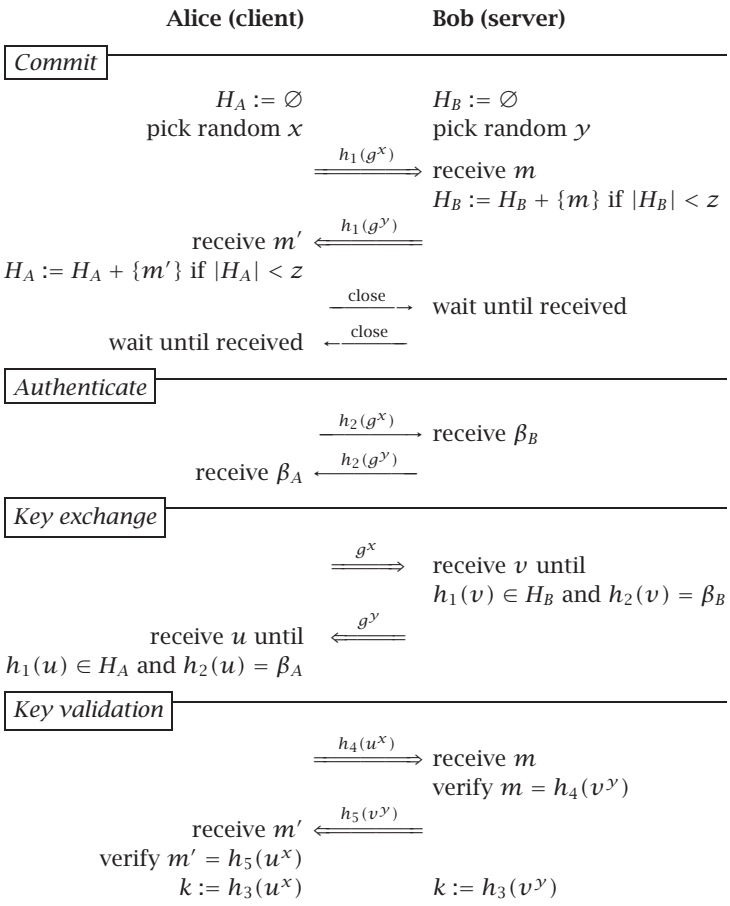

Fig. 3. Message flow of φKE for a bidirectional authentic channel.

w with $h_2(w) = a$. This scenario can be prevented[10] by limiting the number of commitments accepted to a constant $z \ll 2^\eta$.

In Prot. 4.2, the security parameters are determined by the size of the session key established and the capacity of the authentic channel. We set $s = \sigma$ and $t = \eta$.

Analysis It is straightforward to show that in an honest execution of Prot. 4.2, if Alice and Bob want to exchange a key, at the end of the protocol they do actually share the same key.

Security of Prot. 4.2 is proven as follows.

[10] This allows for an obvious denial-of-service attack. However, as the adversary is also capable of blocking any value sent by Alice, he already has the power to prevent Alice from connecting to Bob.

Client (Alice):

Authenticate (1)
$H_A := \emptyset$
pick random x
pick random p
send p **on** pc
Commit
broadcast $h_1(g^x)$ **on** bc
for c milliseconds **do**
on receiving m **from** bc **do**
if $|H_A| < z$
then $H_A := H_A + \{m\}$
abort if $|H_A| = \emptyset$
Authenticate (2)
send $h_2(g^x)$ **on** ac
Synchronise
send close **on** ac
receive q **from** bc
abort if $q \neq p$
Key exchange
$u := \perp$
broadcast g^x **on** bc
receive m **from** bc
if $h'_1(m, p) \in H_A$
then $u := m$
else abort
Key validation
broadcast $h_4(u^x)$ **on** bc
receive m **from** bc
if $h_5(u^x) = m$
then $k = h_3(u^x)$
else abort

Server (Bob):

Authenticate (1)
$H_B := \emptyset$
pick random y
receive π_B **from** pc

Commit
broadcast $h'_1(g^y, \pi_B)$ **on** bc
for c milliseconds **do**
on receiving m **from** bc **do**
if $|H_B| < z$
then $H_B := H_B + \{m\}$
abort if $|H_B| = \emptyset$
Authenticate (2)
receive β_B **from** ac
Synchronise
receive closed **from** ac
broadcast π_B **on** bc

Key exchange
$v := \perp$
broadcast g^y **on** bc
receive m **from** bc
if $h_1(m) \in H_B$ and $h_2(m) = \beta_B$
then $v := m$
else abort
Key validation
broadcast $h_5(v^x)$ **on** bc
receive m **from** bc
if $h_4(v^x) = m$
then $k = h_3(v^x)$
else abort

Protocol 4.3: φKE for a private plus an authentic channel.

Theorem 4.4. *The advantage of an adversary attacking Prot. 4.2 mounting at most q active attacks is at most*

$$O(1 - e^{-zq/2^t}) + O(2^{-s}) .$$

Proof. We split the proof in two cases. We first consider the case where the session key k generated by an oracle is not based on a share g^a sent by the adversary and derived from a value a of his own choosing, and then consider the case where the adversary manages to convince the oracle to use such a share of his own choosing.

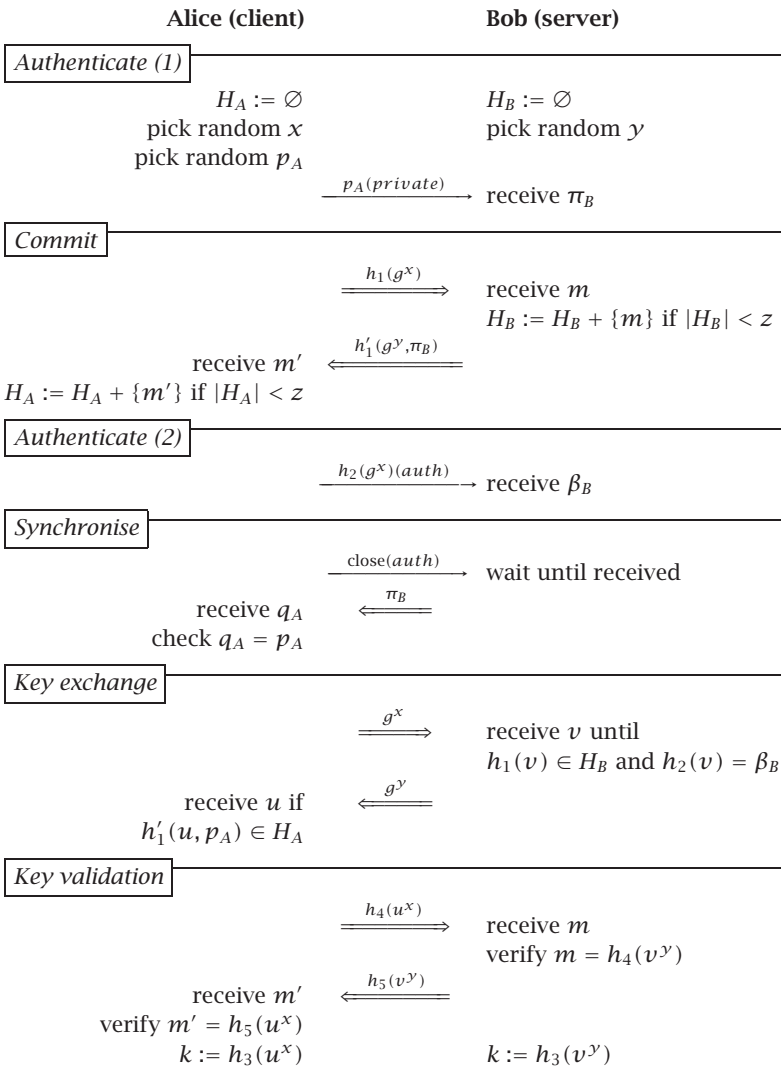

Fig. 4. Message flow of φKE for a private plus an authentic channel.

If the session key generated by an oracle is not based on a share g^a sent by the adversary and derived from a value a of his own choosing, then the proof is almost equal to that of theorem 4.3.

In the other case, in order to convince an oracle of A to use the share g^a of the adversary in the third phase of the protocol, the adversary must ensure that both $h_1(g^a) \in H_A$, and $h_2(g^a) = \beta_A$ holds for values H_A, β_A used in this oracle. Note that β_A is unknown in the commit phase. Moreover, property 4.1 guarantees it is independent of values exchanged during the commit phase. Therefore, for each

value g^a committed by the adversary in the commit phase, the probability that $h_2(g^a) = \beta_A$ is $2^{-\eta}$. As $|H_A| \leq z$, the adversary can commit at most z shares with different hash values for h_2. Hence the total probability that a share of the adversary is accepted is at most $z2^{-\eta}$. With the same estimate as used in the proof of Theorem 4.3, and with $t = \eta$, this proves the theorem. □

4.3 φKE for a private channel plus an authentic channel

In [Hoe04], it was shown that a *single* channel that is both authentic and private can be used to solve the φKE problem straightforwardly using an Encrypted Key Exchange (EKE) protocol [BM92, KOY01, Jab96] as a building block. A combination of techniques from the previous two protocols can be used to show that with two channels both from Alice to Bob, one of which is authentic while the other is private, one can implement φKE. The complete protocol is shown in Prot. 4.3 and Fig. 4. The analysis is very similar to the previous two protocols, and is therefore omitted here.

5 Conclusions and further research

We have shown that the ephemeral pairing problem, and the corresponding ephemeral key exchange problem can also be solved in completely anonymous broadcast networks. Generalisations of the ephemeral pairing problem to larger groups of nodes need to be investigated, as well as the possibilities to weaken the cryptographic assumptions and to simplify the protocols.

References

[BSSW02] Balfanz, D., Smetters, D. K., Stewart, P., and Wong, H. C. Talking to strangers: Authentication in ad-hoc wireless networks. In *NDSS* (San Diego, CA, USA, 2002).

[BPR00] Bellare, M., Pointcheval, D., and Rogaway, P. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT 2000* (Bruges, Belgium, 2000), B. Preneel (Ed.), LNCS 1807, Springer, pp. 139–155.

[BM92] Bellovin, S. M., and Merritt, M. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *IEEE Security & Privacy* (Oakland, CA, USA, 1992), IEEE, pp. 72–84.

[BMP00] Boyko, V., MacKenzie, P., and Patel, S. Provably secure password-authenticated key exchange using Diffie-Hellman. In *EUROCRYPT 2000* (Bruges, Belgium, 2000), B. Preneel (Ed.), LNCS 1807, Springer, pp. 156–171.

[CK01] Canetti, R., and Krawczyk, H. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT 2001* (Innsbruck, Austria, 2001), B. Pfitzmann (Ed.), LNCS 2045, Springer, pp. 453–474.

[DH76] Diffie, W., and Hellman, M. E. New directions in cryptography. *IEEE Trans. Inf. Theory* **IT-11** (1976), 644–654.

[Fel57] Feller, W. *An Introduction to Probability Theory and Its Applications*, 2nd ed. Wiley & Sons, New York, 1957.

[GMN04] GEHRMANN, C., MITCHELL, C. J., AND NYBERG, K. Manual authentication for wireless devices. *RSA Cryptobytes* **7**, 1 (2004), 29–37.

[GN04] GEHRMANN, C., AND NYBERG, K. Security in personal area networks. In *Security for Mobility*, C. J. Mitchell (Ed.). IEEE, 2004.

[GL03] GENNARO, R., AND LINDELL, Y. A framework for password-based authenticated key exchange. Tech. rep., IBM T.J. Watson, 2003. Abstract appeared in EUROCRYPT 2003.

[Hoe04] HOEPMAN, J.-H. The ephemeral pairing problem. In *8th Int. Conf. Fin. Crypt.* (Key West, FL, USA, 2004), LNCS 3110, Springer, pp. 212–226.

[Jab96] JABLON, D. P. Strong password-only authenticated key exchange. *Comput. Comm. Rev.* (1996). `http://www.std.com/~dpj` and `www.integritysciences.com`.

[KOY01] KATZ, J., OSTROVSKY, R., AND YUNG, M. Efficient password-authenticated key exchange using human-memorable passwords. In *EUROCRYPT 2001* (Innsbruck, Austria, 2001), B. Pfitzmann (Ed.), LNCS 2045, Springer, pp. 475–494.

[Luc97] LUCKS, S. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *The Security Protocol Workshop '97* (1997), pp. 79–90.

[Sho99] SHOUP, V. On formal models for secure key exchange. Tech. Rep. RZ 3120 (#93166), IBM, 1999. Invited talk at ACM Computer and Communications Security conference, 1999.

[SA99] STAJANO, F., AND ANDERSON, R. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Procotols, 7th Int. Workshop* (1999), B. Christianson, B. Crispo, and M. Roe (Eds.), LNCS, pp. 172–194.