The Ephemeral Pairing Problem[*]

Jaap-Henk Hoepman

Department of Computer Science, University of Nijmegen
P.O.Box 9010, 6500 GL Nijmegen, the Netherlands
jhh@cs.kun.nl

Abstract In wireless ad-hoc broadcast networks the *pairing problem* consists of establishing a (long-term) connection between two specific physical nodes in the network that do not yet know each other. We focus on the *ephemeral* version of this problem. Ephemeral pairings occur, for example, when electronic business cards are exchanged between two people that meet, or when one pays at a check-out using a wireless wallet.
This problem can, in more abstract terms, be phrased as an *ephemeral key exchange* 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.
Apart from introducing this new problem, we present several ephemeral key exchange protocols, both for the case of authentic channels as well as for the case of private channels.

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

1 Introduction

In wireless ad-hoc broadcast networks like Bluetooth[1] or IrDA[2] there is no guarantee that two physical nodes that want to communicate with each other are actually talking to each other. The *pairing problem* consists of securely establishing a connection or relationship between two specific nodes in the network that do not yet know each other[3]. For example, to insure that a newly bought television set is only controllable by *your* old remote control, the two need to be paired first. Because this pairing is performed only once (or a few times) during the lifetime of any pair of nodes, the pairing procedure can be quite involved. The importance of pairing, and the security policies governing such long-term paired nodes, is described by Stajano and Anderson [SA99].

---

[*] Id: pairing.tex,v 1.11 2003/11/24 11:34:49 hoepman Exp
[1] See http://www.bluetooth.com.
[2] See http://www.irda.org.
[3] Note the subtle difference with authentication: in the pairing problem we are not interested in the actual identity of any of the nodes. In fact, in a wired network the problem is easily solved by checking that a single wire connects both nodes.

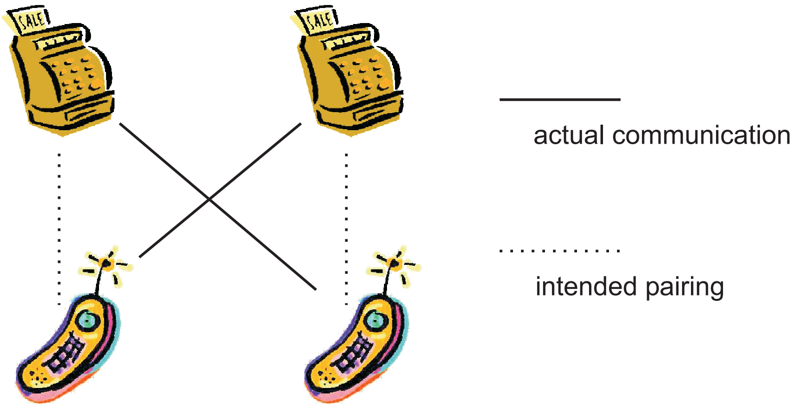

Figure 1. Unwanted exchange of information between unpaired nodes.

Sometimes, pairings may have to be performed much more frequently, and should only establish a relationship for the duration of the connection between the two nodes. Such *ephemeral pairings* occur, for example, when exchanging electronic business cards between two people that happen to meet, or when paying at a check-out using a wireless wallet on your mobile phone. Because such pairings may happen many times a day, the pairing procedure should be fast and the amount of user intervention should be limited. On the other hand, a high level of trust in the pairing may be required. Therefore, the pairing should be established in such a way that a high level of security is achieved even with minimal user interaction. Additionally, privacy may be a concern. Finally, the pairing should be made on the spot, preferably without any preparations.

To achieve such 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. Several realistic methods for doing so are briefly discussed in this paper.

The importance of correctly pairing nodes becomes apparent if we study the two examples just given in slightly more detail (see Fig. 1). If some people in a crowd start exchanging business cards that may also contain quite personal information, the business cards surely should not be mixed up by the wireless network. Similarly, if two people are about to pay using a wireless wallet at two adjacent check-outs in a supermarket, the system should make sure that both are paying the right bills. In fact, similar problems plague smart card purse based systems like the Common Electronic Purse Specifications (CEPS [Cep01]), see [JW01] for details.

The ephemeral pairing problem can also be phrased in more abstract terms as a key exchange problem. Suppose we are 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? We call this problem the *ephemeral key exchange* (denoted by φKE) problem. Here, the low bandwidth channel models the (implicit) authentication and limited information processing capabilities of the users operating the nodes.

1.1 State of the art

The ephemeral key exchange problem is related to the encrypted key exchange (EKE) problem introduced by Bellovin and Merritt [BM92, BM93]. There, two parties sharing a low entropy password are required to securely exchange a high entropy session key. For φKE, the two parties do not share a password, but instead can use a small capacity authentic and/or private channel. EKE protocols are not suitable for this setting directly, most certainly not when only authentic channels are available. However, using private channels and with some minor additions they can be used to solve the φKE problem. This relationship is explored further in Sect. 3.

Jablon [Jab96] thoroughly discusses other solutions to the EKE problem. This paper also contains a good overview of the requirements on and a comparison among different EKE protocols. 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]. This was followed by several more new proposals for EKE protocols secure in this more formal sense, cf. [Mac01, KOY01].

1.2 Contribution and organisation of this paper

We first introduce and define the ephemeral pairing problem and the ephemeral key exchange problem, and show how both are related. To the best of our knowledge, both problems have never before been studied in the literature. Next, in Sect. 3, we present ephemeral key exchange protocols both for the case where the nodes are connected through authentic channels and when the nodes are connected using private channels. In Sect. 4 we discuss how such authentic and private channels could be implemented in practice. We discuss our results in Sect. 5.

2 The ephemeral pairing problem

Consider n physically identifiable nodes communicating over a broadcast network[4], each attended by a human operator. The operators (and/or the nodes they operate) can exchange *small* amounts of information reliably and/or in private.

[4] In general the wireless network may not be completely connected and may change dynamically during the course of the protocol; we can safely ignore these cases, because they do not change the essence of the problem.

The ephemeral pairing problem requires two of these nodes (to be determined by their operators) to establish a shared secret such that

(R1) both nodes are assured the secret is shared with the correct physical node,
(R2) no other node learns (part of) the shared secret, and
(R3) the operators need to perform only simple, intuitive steps.

The shared secret can subsequently be used to set up a secure channel over the broadcast network between the two nodes. The generalised ephemeral pairing problem among $m < n$ nodes requires m nodes to establish a shared secret. We do not study that problem here. A weaker version of the ephemeral pairing problem requires only one node (the *master*) to be assured that the other node (the *slave*) actually shares the secret with it. This is called the *one-sided* ephemeral pairing problem[5].

2.1 Using channels to define the problem

As explained in the introduction, this problem can be seen in more abstract terms as an ephemeral key exchange (φKE) problem. In this case, Alice and Bob share a low bandwidth communication channel over which they can exchange at most η bits of information per message[6]. 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).

These guarantees may hold in both directions, or only in one direction[7]. We note that the low-bandwidth restriction of both the authentic and the private channel is important in practice. For instance, an authentic channel could be implemented by a terminal showing some small number on its public display, that must entered manually on the other terminal. Sect. 4 discusses more examples of such authentic and private channels.

Alice and Bob are also connected through a high bandwidth broadcast network (see Fig. 2). In this paper, we assume that for the correct delivery of broadcast messages, and to separate message streams from different protocol instances, the nodes on the network have unique identities. Messages on the broadcast channel carry a small header containing the identity of both the sender and the receiver. Clearly, the adversary has full control over the contents of these header fields as well. Given these connections, Alice and Bob are required to es-

[5] This applies to the case where the slave is unattended by an operator. A typical scenario would be paying with a wireless wallet (the master) at a vending machine (the slave). Note that now the slave has no clue (physically) with whom it shares the secret.

[6] We require that the number of messages exchanged over the channel in a single protocol run is constant, and small. This, together with the small size of η formalises requirement (R3) above.

[7] Note that in the case of an unidirectional authentic channel for solving the one-sided φKE problem, the channel runs from the slave to the master. See Sect. 4 for concrete examples.

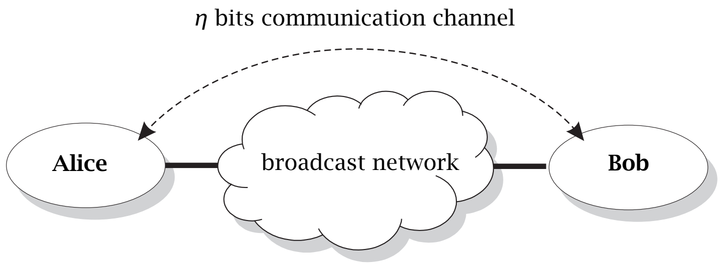

Figure 2. The ephemeral key exchange system model.

tablish 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.

2.2 Model and definitions

We prove security of our protocols in the encrypted key exchange model developed by Bellare *et al.* [BPR00]. For self containment reasons, we briefly summarise this model here.

There is a fixed set of principals, that either behave as clients or as servers. Each principal p may engage in the protocol many times. Each time this creates a new, unique, instance Π_p^i. Instances of a single principal share the global state maintained by that principal. This state is not accessible to the adversary (but see below).

Communication over the network is assumed to be controlled completely by the adversary. Interaction of the adversary with protocol instances of a principal is modelled by giving the adversary access to oracles for those instances. Let P be the protocol under consideration. For each instance Π_p^i the following oracles exist.

Send(p, i, m) Sends or broadcasts message m to instance Π_p^i. Any responses or output according to P are given to the adversary.

Execute(p, i, q, j) Executes a complete protocol run of P between client Π_p^i and server Π_q^j. The adversary learns all the messages exchanged between the instances, and whether they accept or not.

Reveal(p, i) Reveals the session key generated by instance Π_p^i to the adversary.

Test(p, i) Can be called only once at any time in each execution. A bit b is flipped at random, and depending on the outcome the adversary is given either a random session key (when $b = 0$), or the session key generated by instance Π_p^i (when $b = 1$).

An execution of the protocol P is defined as a sequence of oracle calls performed by the adversary. Two instances are called paired[8] if they jointly ran protocol P. For a correct protocol, two paired instances must share the same session key.

The aim of an adversary $\mathcal{A}$ attacking protocol P is to correctly guess whether the call to the $\texttt{Test}(p, i)$ query returned the session key of that instance or just a random session key (or, in other words, to guess the value of the coin flip b used in the query). Let $S_{\mathcal{A}}^P$ denote the event that adversary $\mathcal{A}$ correctly guesses the value of the bit when attacking protocol P. Then the advantage of an adversary $\mathcal{A}$ attacking protocol P is defined as follows:

$$\mathsf{Adv}_{\mathcal{A}}^P = 2\,\mathbf{Pr}\left[S_{\mathcal{A}}^P\right] - 1\ ,$$

(where $\mathbf{Pr}[X]$ denotes the probability of event X). To make this a non trivial task, the adversary is restricted in the the sense that it is not allowed to call the $\texttt{Test}(p, i)$ query if it called the $\texttt{Reveal}$ query on Π_p^i or on the instance paired with it.

Each protocol is actually a collection of protocols that must be instantiated using a particular value for its security parameter. In the case of φKE protocols there are actually two security parameters. There is a large security parameter s (that roughly corresponds to the size of the session key to be established, and that mostly determines the advantage of a passive adversary), and there is a small security parameter t (that roughly corresponds to the capacity of the channel between two principals, and that mostly determines the advantage of an active adversary).

In our analysis we will bound the advantage of the adversary for a particular protocol using s, t and the number of $\texttt{Send}$ queries (denoted by q_s) performed by the adversary. We work in the random oracle model, and assume hardness of the Decisional Diffie Helman 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 :=, and $\stackrel{R}{\leftarrow}$ means selecting an element uniformly at random from the indicated set. 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 $\xRightarrow{m}$ denotes broadcasting m on the broadcast channel. The receiving party puts the message in the indicated variable v at the arrowhead.

[8] Formally, pairing can be defined as follows. Let the trace of an instance be the concatenation of all messages sent and received by that instance. Then two instances are paired when their traces are equal.

if client
  then $p \stackrel{R}{\leftarrow} \{0, \ldots, 2^t - 1\}$
    send p **on** pc
  else receive p **from** pc
$k := \mathrm{EKE}(p)$

Protocol 3.1: φKE for unidirectional private and authentic channel.

3 Ephemeral key exchange protocols

In this section we present φKE protocols, for varying assumptions on the properties of the low bandwidth channel between Alice and Bob. We start with the case where the channel between Alice and Bob is unidirectional and private as well as authentic. Then we discuss the case where the channel is bidirectional. We present a protocol for just private channels, and finish with a protocol where the channel is only authentic.

In some of the protocols, an EKE protocol [BM92, KOY01] is used as the basic building block. This EKE protocol is assumed to broadcast its messages over the broadcast channel instead of sending them point to point.

3.1 φKE for an unidirectional private and authentic channel

In the unidirectional private and authentic channel case, existing EKE protocols can easily be used as a building block. The channel is simply used to reliably send a random password from the client to the server, after which the EKE protocol is run to exchange the key. This is laid down in Prot. 3.1. The security parameters are set by $t = \eta$ and $s = \sigma$.

Analysis We assume the underlying EKE protocol is correct and secure. If Alice and Bob want to exchange a key, it is straightforward to show that in an honest execution of Prot. 3.1, at the end of the protocol they do actually share the same key.

Next we show this protocol is secure.

Theorem 3.1. *The advantage of an adversary attacking Prot. 3.1 is at most the advantage of any adversary attacking the basic EKE protocol.*

Proof. Suppose an adversary attacks a run of protocol Prot. 3.1 with advantage a. Because by assumption, the adversary cannot control or gain information from the messages sent over the private and authentic channel, the advantage of the adversary would still be a when given this run where all messages sent over the channel are random, independent, values. But this is a run over the basic EKE protocol, with additional random values added to it. Hence the adversary can attack the basic EKE protocol with advantage a by adding random values to it and treating it as a run over Prot. 3.1. □

$p \xleftarrow{R} \{0, \ldots, 2^t - 1\}$
send p **on** pc
receive q **from** pc
$r := p \oplus q$
$k := \mathrm{EKE}(r)$

Protocol 3.2: φKE for bidirectional private channel.

Note that each execution of the EKE protocol is given a fresh password. This is unlike the typical case for EKE protocols, where each pair of nodes use the same password each time they wish to connect. This negatively impacts the upper bound for φKE protocols on the advantage of the adversary, in that the advantage of the adversary increases too quickly with the number of times he tries to guess the password. Because Prot. 3.1 uses a fresh password for each execution of the EKE protocol, the upper bound could be improved slightly if we consider one particular instance of an EKE protocol in our analysis.

3.2 φKE for a bidirectional private channel

If the channel is bidirectional and private (without being authentic), existing EKE protocols can also be used as a building block. If the channel is bidirectional, Alice and Bob simply generate two short t bit passwords, exchange them over the private channel, and subsequently run an EKE protocol using the exclusive OR[9] of both passwords as the EKE password to establish the shared session key. Security of this protocol is based on the observation that although anybody can try to set up a session with Bob by sending him a password, Bob will only divulge his own password to the person he wants to connect to, i.e., Alice. Therefore, only Alice is capable of generating the EKE password that will be accepted by Bob. In other words, Alice's authenticity is verified by the fact that she knows Bob's password. The protocol is detailed in Prot. 3.2. Again, the security parameters are set by $t = \eta$ and $s = \sigma$.

Analysis It is again straightforward to show that if Alice and Bob want to exchange a key using Prot. 3.2, they will actually share the same key in an honest execution thereof, if we assume the underlying EKE protocol is correct.

Next we prove security of the protocol.

Theorem 3.2. *The advantage of an adversary attacking Prot. 3.2 is at most the advantage of any adversary attacking the basic EKE protocol.*

Proof. Suppose in a run of Prot. 3.2, an adversary attacks this run with advantage a. The password used by an instance depends on a value received on the private

[9] Using the exclusive OR instead of concatenation makes the resulting EKE password as long as the φKE short security parameter. Moreover, it makes the protocol for Alice and Bob symmetric.

Commit
  pick random x
  broadcast $h_1(g^x)$ **on** bc
  receive α **from** bc
Authenticate
  send $h_2(g^x)$ **on** ac
  receive β **from** ac
Key exchange
  broadcast g^x **on** bc
  receive m **from** bc
  if $h_1(m) = \alpha$ and $h_2(m) = \beta$
    then $u := m$
    else abort
Key validation
  $j := \begin{cases} 0 & \text{if client} \\ 1 & \text{if server} \end{cases}$
  broadcast $h_{4+j}(u^x)$ **on** bc
  receive m **from** bc
  if $h_{5-j}(u^x) = m$
    then $k = h_3(u^x)$
    else abort

Protocol 3.3: φKE for bidirectional authentic channel.

channel, xor-ed with a private random value that is also sent privately to the other party. Because by assumption the adversary cannot gain information from the messages sent over the private channel, the password used in an instance of the basic EKE protocol is independent of the values exchanged over the private channel. Hence by similar reasoning as in theorem 3.1, the advantage of the adversary attacking the basic EKE protocol is at least a. □

3.3 φKE for a bidirectional authentic channel

For the φKE protocol for a bidirectional authentic channel we use a different approach, not using an EKE protocol as the basic building block. The idea behind the protocol (presented as protocol 3.3) is the following.

To establish a shared session key, Alice and Bob will use a Diffie-Helman type key exchange [DH76]. To avoid man-in-the-middle attacks, the shares must be authenticated. However, the capacity of the authentic channel is too small to do so directly. Instead, Alice and Bob proceed in four phases. In the first phase (the *commit phase*) Alice and Bob commit to their shares without revealing them. Then in the *authentication phase* they will send a small authenticator of their share to each other over the authentic channel. In the *key exchange phase*, both will reveal their share. Only shares committed to will be accepted, and the share matching the authenticator will be used to compute the shared session key. The key is verified in the final *key validation phase* to ensure that Alice and Bob

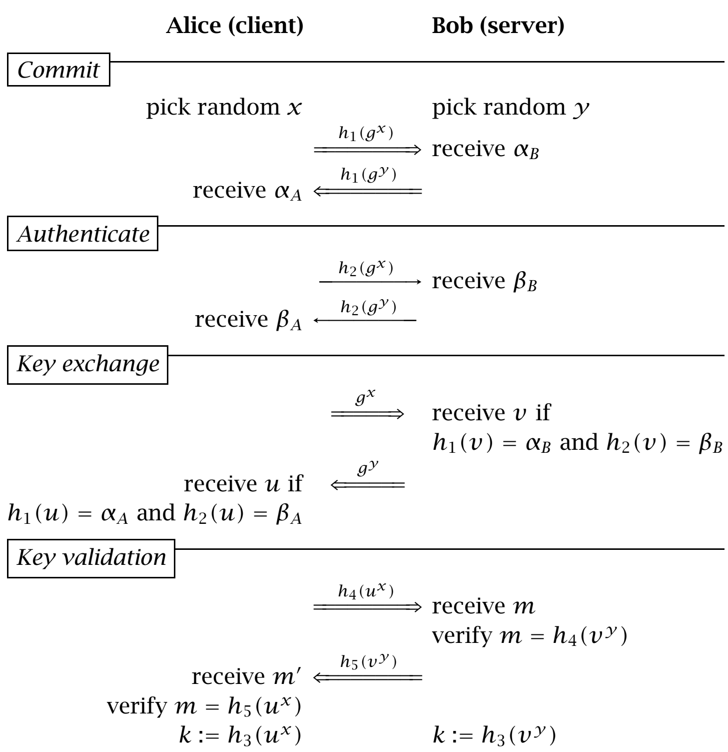

Figure 3. Message flow of φKE for a bidirectional authentic channel.

indeed share the same session key, using the mechanism described in [BPR00]. Only if the validation phase is successful the protocol will accept.

Note that we must first commit to a value before revealing either the value or the authenticator, or else 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.

In Prot. 3.3, 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 $t = \eta$. G is a group of order at least 2^{2s} with generator g for which the Decisional Diffie Helman (DDH) problem is hard. A possible candidate is the subgroup of order q in $\mathbb{Z}_p^*$ for p, q prime and $p = 2q + 1$ [Bon98]. Naturally, exponentiations like g^x are computed in the group G.

Furthermore, we use two hash functions $h_1 : G \mapsto G$ and $h_2 : G \mapsto \{0,1\}^\eta$, that satisfy the following property.

Property 3.3. Let X be a uniformly distributed random variable over G, and let $a \in \{0,1\}^\eta$ and $b \in G$ be arbitrary. We assume that the two hash functions h_1, h_2 satisfy

$$\mathbf{Pr}\,[\, h_2(X) = a \,|\, h_1(X) = b\,] = \mathbf{Pr}\,[h_2(X) = a] = 2^{-\eta} \ .$$

Finally, pairwise independent hash functions $h_3, h_4, h_5 : G \mapsto \{0,1\}^\sigma$ are used as well. In practice, these hash functions can be derived from a single hash function h using the equation $h_i(x) = h(x \parallel i)$ (where $\parallel$ denotes concatenation of bit strings).

Analysis It is straightforward to show that in an honest execution of Prot. 3.3, 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. 3.3 is proven as follows. We use the following result presented by Boneh [Bon98], which holds under the assumption that the Decisional Diffie Helman problem over G is hard.

Proposition 3.4. *Let the order of G be at least 2^{2s}, and let $h_3 : G \mapsto \{0,1\}^s$ be a pairwise independent hash function. Then the advantage of any adversary distinguishing $h_3(g^{ab})$ from a random element of $\{0,1\}^s$, when given g^a, g^b is a most $O(2^{-s})$.*

Using this proposition we are able to prove the following theorem.

Theorem 3.5. *The advantage of an adversary attacking Prot. 3.3 using at most q_{send} send queries is at most*

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

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.

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 k depends on private random values x, y unobserved by the adversary and publicly exchanged shares g^x and g^y using a Diffie-Helman (DH) key exchange. Any adversary attacking Prot. 3.3 can be converted to an adversary attacking a basic DH key exchange, by inserting the necessary hashes $h_i(g^x)$ and $h_j(g^y)$ (for $i, j \in \{1,2,3\}$) and random values for $h_4()$ and $h_5()$ (this is possible due to the random oracle model and Prop. 3.4) in the run of the basic DH key exchange 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-Helman key exchange, which is at most $O(2^{-s})$ by Prop. 3.4.

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

- $h_1(g^a) = \alpha$, and
- $h_2(g^a) = \beta$

for values α, β used in this oracle. Note that β is unknown in the commit phase. Moreover, property 3.3 guarantees it is independent of values exchanged during the commit phase. Therefore, for any value g^a committed by the adversary in the commit phase, the probability that $h_2(g^a) = \beta$ is $2^{-\eta}$.

For each send query then the probability of success is $2^{-\eta}$. Success with one instance is independent of success in any other instance. Hence, with q_{send} send queries, the probability of success becomes (cf. [Fel57])

$$1 - (1 - 2^{-\eta})^{q_{\text{send}}} \approx 1 - e^{-2^{-\eta} q_{\text{send}}}$$

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

Note that in fact the advantage of the adversary attacking the φKE protocol is strictly less than the advantage of the adversary attacking password based EKE protocols, like the protocol of Katz *et al.* [KOY01] whose advantage is bounded by

$$O(q_{\text{send}}/2^t) + O(2^{-s}) ,$$

where, loosely speaking, q_{send} is the number of times the adversary tries to guess the password. The difference is caused by the fact that in the EKE setting, multiple instances of the protocol use the same password[10].

4 Applications

Beyond those mentioned in the introduction, there are many other situations that involve ephemeral pairing.

- Connecting two laptops over an infrared connection, while in a business meeting.
- Buying tickets wirelessly at a box office, or verifying them at the entrance.
- Unlocking doors using a wireless token, making sure the right door is unlocked.

For all these applications it is very important that the burden of correctly establishing the right pairing should not solely rest on the user. The user may make mistakes, and frequent wrong pairings will decrease the trust in the system. This is especially important for applications that involve financial transactions. On the other hand, some user intervention will obviously always be required. The trick is to make the user actions easy and intuitive given the context of the pairing.

In the next section, we describe how a low bandwidth authentic or private channel can be implemented in quite practical settings. These are of course merely suggestions. There are probably many more and much better ways to achieve the same effect. The point here is, however, to merely show that such channels can be built in principle.

[10] This could be overcome by allowing only the first z connections to use the password alone, and using parts of the previously established shared secrets to generate new, longer, passwords. Then the bound on the advantage of the adversary essentially becomes equal to ours.

4.1 Implementing the low bandwidth authentic or private channel

To implement an authentic or private channel in practice, several solution strategies are applicable.

- Establishing physical contact, either by a wire, through a connector, or using proximity techniques.
- Using physical properties of the wireless communication link, that may allow 'aiming' your device to the one you wish to connect to.
- Using fixed visible identities, either using explicitly shown unique names on devices, or using the unique appearance of each device.
- Using small displays that can either be read by the operator of the other device or read directly by the other device.

Which strategy to select depends very much on the specific application requiring ephemeral pairing. We will discuss each of these strategies briefly.

Physical contact The easiest way to solve ephemeral pairing is to connect both nodes (temporarily) physically, either by a wire, or by making them touch each others conductive pad. The resulting physical connection can be used as the private or authentic channel in the previous protocol. Or it can be used to exchange the shared secret directly, of course

Fixed visible identities Here one could use for example numbers, or the physical appearance. Each node holds a unique private key, and the physical identity is bound to the corresponding public key using a certificate generated by the certification authority (CA) managing the application.

The main drawback is that these solutions require a central Certification Authority. Moreover, the a priori distribution of secrets is contrary to the spirit of the φKE problem.

A variant (described in [Mob01]) uses the fixed identity of nodes in the following way. Any node wishing to connect can do so. Each connection is assigned a unique and small connection number, which is shown on a display. The user mentions the number to the merchant, who then initiates a payment over the indicated connection. The problem with this setup is that it is vulnerable to man-in-the-middle attacks.

Physical link properties Depending on the properties of the physical link, one could reliable aim a device at another, or safely rule out connections to/from other far away devices.

Operator read displays In this scheme, each node has a small display and a way to select several images or strings from the display (through function keys or using a touch pad). An authentic channel can be implemented as follows. To send a η bit string, it is converted to a simple pattern that is shown on the display of the sender. The receiver enters the pattern on its device, which converts it back to the η bits.

5 Conclusions

We have formulated the ephemeral pairing problem, and have presented several ephemeral key exchange protocols showing that this problem can be solved using small capacity, and mostly bidirectional, point to point channels and a broadcast network with identities to separate communication streams.

More work needs to be done to develop φKE protocols using only unidirectional channels, or on truly anonymous broadcast networks.

It would be interesting to develop protocols that are correct under less strong assumptions, i.e., ones that do not require to assume either the random oracle model or hardness of the Decisional Diffie Helman problem (or both). The same holds for the assumption on the authentic channel that adversary cannot modify or inject messages of his choice at all. More research is needed to investigate the effects on the advantage of the adversary if he can modify or inject messages on the authentic channel with low success probability.

6 Acknowledgements

This work was inspired by the work of Yan Yijun on a payments architecture for mobile systems, under the supervision of Leonard Franken of the ABN AMRO bank and myself. I would like to thank Yan and Leonard for fruitful initial discussions on this topic.

References

[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.

[BM93] Bellovin, S. M., and Merritt, M. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *1st CCS* (Fairfax, VA, USA, 1993), ACM, pp. 244–250.

[Bon98] Boneh, D. The decision Diffie-Hellman problem. In *Proc. of the 3rd Algorithmic Number Theory Symp.* (1998), LNCS 1423, pp. 48–63.

[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.

[Cep01] CepsCo. Common Electronic Purse Specifications: Technical specification, version 2.3, 2001. `http://www.cepsco.com`.

[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.

[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.

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

[JW01] JÜRJENS, J., AND WIMMEL, G. Security modelling for electronic commerce: The common electronic purse specifications. In *First IFIP conference on e-commerce, e-business, and e-government (I3E)* (Zürich, Switzerland, 2001), Kluwer.

[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.

[Mac01] MACKENZIE, P. More efficient password-authenticated key exchange. In *Topics in Cryptography (CT-RSA)* (2001), LNCS 2020, Springer, pp. 361–377.

[Mob01] MOBILE ELECTRONIC TRANSACTIONS. Solution to Bluetooth multiuser problem using method of tokens. Tech. rep., 2001. `http://www.mobiletransaction.org/`.

[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.