# Manual authentication for wireless devices

Christian Gehrmann
Ericsson Mobile Platforms
Lund, Sweden.
christian.gehrmann@ericsson.com

Chris J. Mitchell
Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK.
c.mitchell@rhul.ac.uk

Kaisa Nyberg
Nokia Research Center
Helsinki, Finland.
kaisa.nyberg@nokia.com

January 23, 2004

## Abstract

Manual authentication techniques have been designed to enable wireless devices to authenticate one another via an insecure wireless channel with the aid of a manual transfer of data between the devices. Manual transfer refers to the human operator of the devices performing one of the following procedures: copying data output from one device into the other device, comparing the output of the two devices, or entering the same data into both devices. Techniques currently being standardised are described which achieve this, and which require only small amounts of data to be transferred between the two devices. This makes the mechanisms particularly attractive for non-expert use, as required for ubiquitous mobile wireless devices.

## 1 Introduction

Entity authentication and authenticated key establishment are of fundamental importance in establishing secure communications between a pair of communicating parties. Entity authentication is normally provided when a communications link is established and, if an authenticated key is established simultaneously, this can be used to protect subsequently exchanged data. The purpose of this paper is to examine how these services might best be achieved for personal wireless-enabled devices.

Using the terminology of Stajano [12], the problem is that of securely 'imprinting' a personal device. That is, suppose a user has two wireless-enabled devices, e.g. a mobile phone and a Personal Digital Assistant (PDA); suppose further that he/she wishes the two devices to establish a secure association for their wireless communications. This will, for example, enable the two devices to securely share personal data. The problem is thus for the two devices to mutually authenticate one another and, where necessary, to establish a shared secret key, all using a wireless communications link. A shared secret key can be used as the basis for future secure communications between the two devices, including further mutual authentications.

The main threat to the process is via a so-called 'man-in-the-middle attack' on the wireless link. Because the link uses radio, a third party with a receiver and a powerful transmitter could manipulate the communications between the devices, in a way that will not be evident to the user. Thus, the attacker could masquerade as the first device to the second device, and also as the second device to the first device, and set up separate keys with each. To prevent this, it will be necessary for the device operator to input and output data via the devices' user interfaces (i.e. performing a manual data transfer) to enable the devices to verify each other's identities.

This is the context of use for the manual authentication protocols described here. We make the following assumptions about the two devices.

- The two devices have access to a wireless communications channel, which can be used to exchange as much data as required; however, no assumptions are made about the security of this channel — for example, it may be prone to manipulation by an active attacker.
- The two devices are both under the control of either a single human user, or a pair of users who trust one another and who share a communications channel whose integrity is protected by some means (e.g. using handwritten notes or a voice channel). Both devices have a means to input or output a sequence of digits, i.e. they have at least a numeric keypad or a multi-digit display.
- If a device does not have a keypad, then it must at least have an input, e.g. a button, allowing the successful conclusion of a procedure to be indicated to the device. Similarly, if a device lacks a multi-character display, then it must at least have an output capable of indicating the success or failure of a procedure (e.g. red and green lights or a sound output).

We do not assume that the devices have any prior keying relationship or are equipped with any keys by their manufacturers. Of course, the problem would become dramatically easier if every device had a unique signature key pair and a certificate for their public key signed by a widely trusted Certification Authority (CA). However the overhead of personalising every device in this way is likely to be prohibitive, particularly for low-cost devices.

Similarly, we do not assume that the two devices share a trusted communications link, e.g. as might be provided by a hard-wired connection. Such a link, even if it only guaranteed data integrity and data origin authentication (and not confidentiality), would again make the problem simple, since it could be used to authenticate a Diffie-Hellman exchange (as described in section 2). However, it would be unreasonable to always expect such a link to exist, since many simple wireless devices are likely to possess no wired communications interfaces.

An emerging international standard, ISO/IEC 9798-5 [6], currently at Committee Draft ballot stage, contains a set of 'manual authentication' solutions to the wireless device imprinting problem. Some of the schemes in this standard are described in sections 3 and 4 below. The same schemes may also be included in a future version of the Bluetooth standards. The existing Bluetooth specifications already contain a solution to device imprinting, but this solution has well-known security shortcomings if the initial exchange between devices can be wiretapped [3, 8]. A more detailed discussion of manual authentication can be found in [3].

## 2 Using Diffie-Hellman

Perhaps the most straightforward solution to the imprinting problem is to use the Diffie-Hellman key establishment protocol [1, 11]; this approach was first proposed by Maher [10]. As discussed in [3], this is also the solution proposed by Stajano and Anderson, [12, 13]. We thus first describe such a solution.

## 2.1 Procedure

The two devices first agree on (or are pre-programmed with) a secure set of Diffie-Hellman parameters, namely a large prime $p$, a large prime $q$ dividing $p-1$, and a value $g$ of multiplicative order $q$ mod $p$. In fact the Diffie-Hellman parameters could be standardised, or made the subject of an industry agreement. This would make the task of equipping all devices with the parameters very simple, and certainly much simpler and cheaper than giving each device an individual key pair and certificate.

The two devices, $A$ and $B$ say, then both generate a random value between 1 and $q-1$ — call these values $a$ and $b$. $A$ computes $g^a \bmod p$ and sends it to $B$, and $B$ computes $g^b \bmod p$ and sends it to $A$ (in both cases using the wireless link). Finally $A$ computes the shared key $K$ as $K = (g^b)^a \bmod p$ and $B$ computes the same key as $(g^a)^b \bmod p$.

Of course, as is widely understood (see, for example, [11]) this procedure does not provide mutual authentication, since the transmitted Diffie-Hellman values could have been manipulated by an interceptor acting as a man-in-the-middle (as above). Mutual authentication can be achieved by the manual exchange of checksums, i.e. using a 'manual authentication' technique, as follows. Note that, apart from the authentication issue, the security of the Diffie-Hellman protocol has been widely studied. If the parameters are chosen appropriately, then it is believed to be secure.

To provide the desired authentication, Maher [10] proposed the following additional steps. After completing the Diffie-Hellman exchange, both devices input the key $K$ to a one-way hash-function $h$, e.g. SHA-1 [5], to obtain $h(K)$, which can be truncated to the desired length by taking the leftmost bits of the output (the choice for the length is discussed below). Suppose now that one device has a display and the other a keypad. The device with a display outputs the hash-code, e.g. as a sequence of hexadecimal digits. The user now enters this sequence into the second device using its keypad. The second device compares the input hash-code with its computed value and, if they agree, provides a positive indication to the user. If this positive indication is received, the user inputs a success indication into the first device. This completes the mutual authentication process, and both devices also now have an authenticated key.

A similar procedure can be followed if both devices have a display. In this case both devices output their computed hash code, and the user is then simply required to compare the values output by the two devices. If they agree then the user gives a positive indication to both devices.

If one of the devices possesses neither a display nor a keypad, then it is difficult to apply the above method. However, in the case of a device with an audio output, e.g. a wireless headset, it may be possible for the headset to 'speak' the digits to the user, thus providing the functions of a display.

## 2.2 Issues

The main problem with the above procedure is the number of digits that need to be typed or examined by the user. Typing in a large number of digits to a small numeric keypad without making an error is a non-trivial procedure, and one that many users are likely to find too demanding to carry out. This is bad news for manufacturers of consumer devices. If the device cannot operate without completing the procedure then repeated failures to perform it correctly will cause the user to be very frustrated with the supplier. Alternatively, if the device can be used without a secure imprinting process, then this is a situation which could give rise to serious security vulnerabilities, which could also seriously damage the supplier's reputation.

One possible solution is to drastically truncate the hash-code, e.g. to the first 16 or 32 bits — this would mean that the user would only have to type in (or compare) 4 or 8 hexadecimal digits, respectively. Whilst this is attractive, it has serious security weaknesses, as follows.

### 2.3 Attacking short hash-codes

Typically one device (say $A$) will send its Diffie-Hellman value first, and then wait for the response from $B$. Suppose that $A$ first sends $g^a \bmod p$ to $B$. Suppose also that an active interceptor of the wireless link, $C$ say, prevents this from reaching $B$ and replaces it with $g^{a'} \bmod p$, for some value $a'$ chosen by $C$.

$B$ responds with $g^b \bmod p$, and $B$ simultaneously computes the shared key as $K_B = g^{a'b} \bmod p$. Now suppose that $C$ also intercepts $g^b \bmod p$ and prevents it from reaching $A$. It is important to observe that, because $C$ chose $a'$, $C$ is now able to compute the key held by $B$, i.e. $C$ knows $K_B$.

$C$ next generates a series of random values $b'$, and for each such value computes $K' = (g^a)^{b'} \bmod p$ and $h(K')$. $C$ then compares the first 16 bits of $h(K')$ with the first 16 bits of $h(K_B)$. If they agree then $C$ simply sends $g^{b'} \bmod p$ to $A$, who generates the key $K_A = (g^{b'})^a \bmod p$. Because of the way in which $b'$ was chosen by $C$, the truncated hashes computed by $A$ and $B$ will match, although they do not share a secret key. Moreover, worst of all, $C$ will know the values of the keys held by $A$ and $B$.

This attack requires the attacker to perform a significant amount of work in a short time, i.e. before $A$ and $B$ 'time out'. Specifically, if the hash-function is truncated to $t$ bits, then on average $C$ will need to generate $2^{t-1}$ values $b'$ before one is found which yields the desired hash-value. Thus for $t = 16$, $t = 32$ and $t = 48$ the attack requires 30,000, 2 billion and 150 trillion trials respectively. If an attacker with significant computing resources wished to attack the imprinting process, then it might be feasible to perform one billion trials in a second, and we might reasonably assume the 'time-out' value to be at most 10 seconds, allowing time for the attacker to perform 10 billion trials.

Thus using a hash-code of at least 48 bits appears to be necessary to rule out the possibility of success in attacking the imprinting process, given a well-equipped 'man-in-the-middle'. 48 bits amounts to 12 hexadecimal digits, which is already quite a significant number for a user to enter in an error-free way, particularly when using a very small keypad with no display to enable the user to check the correctness of each key depression.

Ideally we would like a solution in which such a man-in-the-middle attack can be prevented without requiring the users to type in or compare long strings of digits. Such schemes form the focus of the remainder of this paper.

## 3 Manual authentication using a short check-value

We first describe an example of a scheme which uses keyed check-functions having short check-values (e.g. of around 16–20 bits) and using short keys (again of 16-20 bits). These check-functions are essentially MAC (Message Authentication Code) functions producing short outputs. To maximise the provable performance of the scheme, Gehrmann and Nyberg [3] have proposed using a coding theory construction to compute the check-values; this scheme is included in the draft standard [6]. However, in practice, use of a conventional MAC function (e.g. a CBC-MAC based on use of a block cipher — see, for example, [11]) will almost certainly be sufficiently secure (in such a case the short key could be padded with a fixed string to construct a block cipher key).

## 3.1 The MANA I scheme

The scheme we describe, called MANA I (for MANual Authentication) in [3] and mechanism 1 in [6], is designed for use in the situation where one device ($A$) has a display and the other ($B$) has a keypad, although a simple variant (MANA II) exists for the case where both devices have a display. MANA I and MANA II were originally published in [2].

We also assume that the two devices wish to agree on the value of a public data string $D$. This data string could be the concatenation of $A$'s and $B$'s public keys, for some asymmetric cryptosystem. This could support the registration process for a small-scale PKI, or could simply be used as the basis for subsequent secure communications. In particular the public keys could be used, e.g. as Diffie-Hellman public keys, to provide the basis for an authenticated secret key establishment protocol, requiring no further intervention by the user.

We write $m_K(X)$ for the check-value computed using key $K$ and data string $X$. The scheme operates as follows (see also Figure 1).

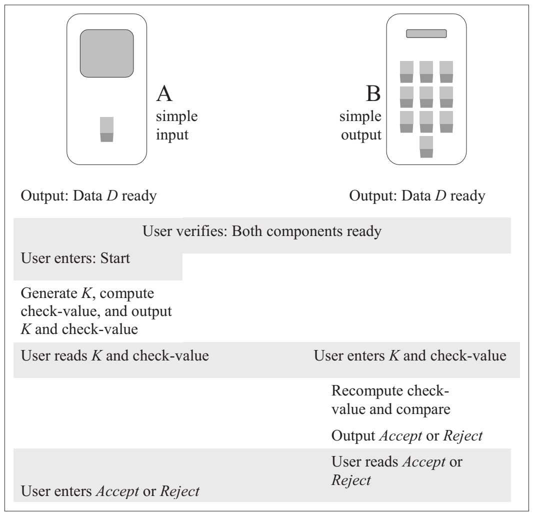
A
simple input
B
simple output
Output: Data $D$ ready
Output: Data $D$ ready
User verifies: Both components ready
User enters: Start
Generate $K$, compute check-value, and output $K$ and check-value
User reads $K$ and check-value
User enters $K$ and check-value
Recompute check-value and compare
Output *Accept* or *Reject*
User reads *Accept* or *Reject*
User enters *Accept* or *Reject*

Figure 1: Manual authentication using a short check-value

1. A data string $D$ is agreed by some means between $A$ and $B$ using the wireless channel. This would typically occur via an exchange of (unprotected) messages.

2. Device $A$ generates a random key $K$ of length appropriate for use with the check-function (i.e. of 16–20 bits); $A$ also generates the check-value $m_K(D)$. The key and check-value are then output to the display by device $A$.

3. The user enters the check-value and the key $K$, read from the display of device $A$, into device $B$ (using the keypad).

4. Device $B$ uses the key $K$ provided by the user to recompute $m_K(D)$, and compares this with the value entered by the user. The device outputs an indication of success or failure, depending on whether or not the check-values agree.

5. The user copies this success/failure indication back into device $A$.

## 3.2 Analysis of the scheme

First note that the key and check-value are not available to any would-be attacker, who only sees the data $D$. The only possible strategy for the attacker is to try to persuade $A$ and $B$ to agree on different data strings $D_A$ and $D_B$ respectively, with the property that

$$m_K(D_A) = m_K(D_B)$$

for the largest possible number of secret keys $K$. In the coding theory construction for the check-function $m$ (as in [3]) this largest number of keys is known. From it an upper bound on the success probability of the attacker can be determined. However to carry out such a substitution attack, and to reach the largest success probability, would require a lot of computation. In practice an attacker cannot do significantly better than choose arbitrary strings $D_A$ and $D_B$ and hope that the check-values computed with the key that is unknown to the attacker will be the same. Even if a check-value construction based on a conventional CBC-MAC is used, it is very unlikely that the attacker will be able to do much better than with the coding theory construction, given that $A$ and $B$ choose parts of $D_A$ and $D_B$, respectively. If a guess fails, no off-line computation can help the attacker any further. Once the parties have agreed on a key, it is no longer possible to attack the manual authentication scheme or to persuade either party to accept an incorrect value $D$. The only possible remaining vulnerability would be in any subsequent key exchange process based on use of the authenticated value $D$, e.g. a Diffie-Hellman key agreement (which, as we have discussed previously, is believed to be secure).

In summary, and assuming that the coding theory construction is used for the check-function, the probability of a successful attack (where $A$ and $B$ agree on different data values) is less than $2^{-13}$, i.e. 1 in 8,000, for 16-bit keys and check-values and less than $2^{-17}$, i.e. 1 in 130,000, for 20-bit keys and hash-codes. More details are given in [3, 6].

### 3.3 Variants of the scheme

Since the data string $D$ is generated by $A$, it does not need to be sent to $B$ until after step 2. In fact, the data transfer can be delayed indefinitely, and the key and check-value transferred manually to $B$ can act as a 'certificate' for the subsequently exchanged data $D$. This might be useful, for example, where a newly purchased device is 'manually authenticated' as soon as it is switched on, but where public keys are only generated and exchanged at some later time.

Finally note that a variant of the above mechanism, known as MANA II in [3] and mechanism 2 in [6], can be devised to cover the situation where both devices $A$ and $B$ have a display, but neither of them has a keypad (although they must both possess a means of indicating successful completion of the protocol).

Briefly, in this case, the first two steps are as in MANA I. However, in addition to displaying the key and check-value, device $A$ also sends the key to device $B$ via the wireless channel (and hence in this case the key is available to an attacker). Device $B$ uses the received key to recompute the check-value on its version of the data string, and finally displays the key received from $A$ together with the check value it has computed. The user completes the process by comparing the values displayed by the two devices. Only if the key and check-value agree completely does the user give a 'success' indication to both devices.

## 4 Manual authentication using a MAC function

A different class of manual authentication protocols can be constructed using a conventional MAC function, such as HMAC [4] or a block cipher based CBC-MAC.

### 4.1 The MANA III scheme

The scheme we describe is MANA III from [3] (it is also specified as mechanism 3a in ISO/IEC CD 9798-6 [6]). It is designed for use in the situation where both devices have a keypad, although a simple variant exists for the case where one device has a display (see below). As previously, we assume that the two devices wish to agree on the value of a public data string $D$, where $D$ could be used for agreeing public keys, as described in section 3.1.

We write $m_K(X)$ for the MAC computed using key $K$ and data string $X$. The scheme operates as follows (see also Figure 2).

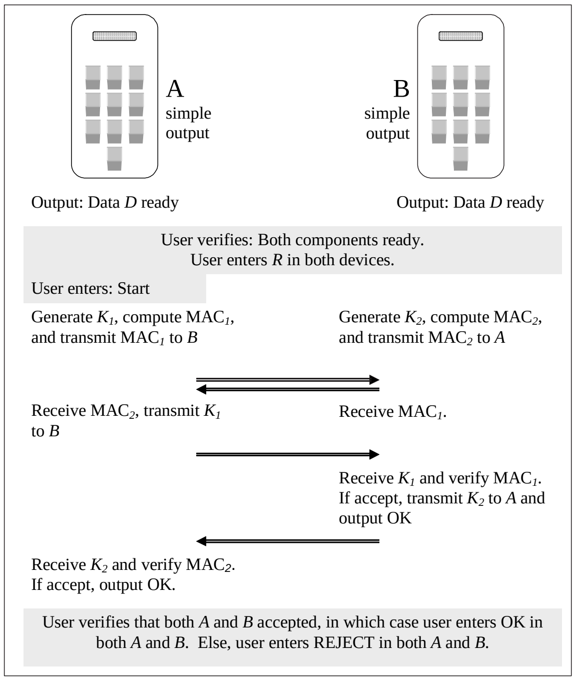
A
simple output
B
simple output
Output: Data $D$ ready
Output: Data $D$ ready
User verifies: Both components ready.
User enters $R$ in both devices.
User enters: Start
Generate $K_1$, compute MAC$_1$, and transmit MAC$_1$ to $B$
Generate $K_2$, compute MAC$_2$, and transmit MAC$_2$ to $A$
Receive MAC$_2$, transmit $K_1$ to $B$
Receive MAC$_1$.
Receive $K_1$ and verify MAC$_1$. If accept, transmit $K_2$ to $A$ and output OK
Receive $K_2$ and verify MAC$_2$. If accept, output OK.
User verifies that both $A$ and $B$ accepted, in which case user enters OK in both $A$ and $B$. Else, user enters REJECT in both $A$ and $B$.

Figure 2: Manual authentication using a MAC

1. A data string $D$ is agreed between $A$ and $B$ using the wireless channel.
2. The user generates a short random bit-string $R$, e.g. of 16–20 bits, and enters it into both devices.
3. Device $A$ generates a random MAC key $K_1$ and computes the MAC value $M_1 = m_{K_1}(I_A||D||R)$, where $I_A$ is an identifier for $A$ and $||$ denotes concatenation of data items. Device $A$ sends $M_1$ to $B$ via the wireless link.
4. Device $B$ generates a random MAC key $K_2$ and computes the MAC value $M_2 = m_{K_2}(I_B||D||R)$, where $I_B$ is an identifier for $B$. Device $B$ sends $M_2$ to $A$ via the wireless link.
5. When device $A$ receives $M_2$ from $B$ (and not before), $A$ sends $B$ the key $K_1$.
6. When device $B$ receives $M_1$ from $A$ (and not before), $B$ sends $A$ the key $K_2$.
7. On receipt of $K_2$, $A$ uses it to recompute $M_2$, where the data employed in the computation consists of its stored value of $D$, the expected identifier $I_B$, and the random value $R$ input by the user. If the recomputed $M_2$ agrees with the value received from $B$ then $A$ indicates success.
8. On receipt of $K_1$, $B$ uses it to recompute $M_1$, where the data employed in the computation consists of its stored value of $D$, the expected identifier $I_A$, and the random value $R$ input by the user. If the recomputed $M_1$ agrees with the value received from $A$ then $B$ indicates success.
9. If (and only if) both devices indicate success, the user indicates success to both devices.

Finally note that steps 2/3 and also 4/5 may be conducted in parallel.

## 4.2 Analysis of the scheme

The MANA III scheme is a slightly modified version of a protocol called SHAKE [9]. Informally, the security of the scheme relies on the fact that $R$ remains secret to the attacker (it is never sent over the air) and both $A$ and $B$ release a commitment (i.e. the MAC value) to the data $D$ before releasing the key used to compute this commitment.

In order for the scheme to work, the last step must be performed, since it is indeed easy for a forgery to make a full (not successful) exchange with $A$, calculate $R$ by exhaustive search, and then make a successful exchange with $B$. However, such an attack will be detected by the double check in the last step. Hence, the interceptor's only hope of attacking the scheme is to determine $R$ from the MAC values, but in the absence of the keys this is infeasible.

Thus the best approach for the attacker is to guess $R$. The likelihood of a successful attack is thus $2^{-r}$, for an $r$-bit value $R$, i.e. the odds against a successful attack are 1 in 70,000 for a 16-bit random value $R$, and 1 in a million for a 20-bit $R$.

Of course, this calculation assumes that $R$ is chosen at random from all possible $r$-bit values. In practice, if $R$ is chosen by a human, then some values will be more likely than others. This can be exploited by an attacker whose best strategy is simply to guess the most likely $r$-bit value. However, this is unlikely to drastically reduce the security of the system unless the attacker understands well the behaviour of the user being attacked. Also, if $R$ is instead chosen by one of the devices, as in the variant scheme described immediately below, then the risks associated with a poor choice of $R$ are avoided.

### 4.3 Variants of the scheme

Two variants of the scheme exist, both of which are included in the draft standard [6].

The first variant (originally proposed by Jakobsson [7] and listed as mechanism 3b in [6]) involves a total of $r$ 'rounds', where $r$ is the number of bits in $R$. In each round, one bit of $R$ is used, and the devices exchange MACs and keys as in the scheme described above. While this increases significantly the amount of data exchanged between the devices, it removes the need for the user to give success indications to the devices at the end of the protocol.

The second variant (mechanism 4 in [6]) applies to the case where one device has a display and the other has a keypad. In this case, step 2 is modified so that the device with the display generates the random value $R$ and displays it to the user, who then enters it into the other device. All other steps of the scheme remain unchanged.

## 5 Concluding remarks

The schemes described in this paper meet the objective of enabling two wireless devices to securely authenticate one another and agree on a shared data string. This is achieved without the need for the user to enter or compare long strings of digits. The user is typically only required to type in (or compare) around 32 binary digits (e.g. in the form of eight hexadecimal digits).

Mechanism MANA III further improves on the situation and only requires the user entry of 16 bits, although these digits must be entered into both devices or read from one device and typed into the other.

## References

[1] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, **IT-22**:644–654, 1976.

[2] C. Gehrmann and K. Nyberg. Enhancements to Bluetooth baseband security. In *Proceedings of Nordsec 2001, Copenhagen, Denmark*, November 2001.

[3] C. Gehrmann and K. Nyberg. Security in personal area networks. In C. J. Mitchell, editor, *Security for Mobility*, pages 191–230. IEE, London, 2004.

[4] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 10118–2, Information technology — Security techniques — Hash-functions — Part 2: Hash-functions using an $n$-bit block cipher*, 2nd edition, 2000.

[5] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 10118–3: 2003, Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions*, 2nd edition, 2003.

[6] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 1st CD 9798-6, Information technology — Security techniques — Entity authentication — Part 6: Mechanisms using manual data transfer*, December 2003.

[7] M. Jakobsson. Method and apparatus for immunizing against offline dictionary attacks. U.S. Patent Application 60/283,996. Filed on 16th April 2001.

[8] M. Jakobsson and S. Wetzel. Security weaknesses in Bluetooth. In David Naccache, editor, *Topics in Cryptology — CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings*, volume 2020 of *Lecture Notes in Computer Science*, pages 176–191. Springer-Verlag, Berlin, 2001.

[9] J.-O. Larsson. Higher layer key exchange techniques for Bluetooth security. Open Group Conference, Amsterdam, October 2001.

[10] D. P. Maher. Secure communication method and apparatus. U.S. Patent Number 5,450,493, September 1995. Filed on 29th December 1993.

[11] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.

[12] F. Stajano. *Security for Ubiquitous Computing*. John Wiley and Sons Ltd., 2002.

[13] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, editors, *Security Protocols, 7th International Workshop, Cambridge, UK, April 19-21, 1999, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 172–194. Springer-Verlag, Berlin, 2000.