# Auxiliary Channel Diffie-Hellman Encrypted Key-Exchange Authentication

Dennis K. Nilsson
dennis.nilsson@chalmers.se

Ulf E. Larson
ulf.larson@chalmers.se

Erland Jonsson
erland.jonsson@chalmers.se

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden

## ABSTRACT

In wireless personal area networks, establishing trust and authentication with previously unknown parties is both necessary and important. We propose an auxiliary channel Diffie-Hellman encrypted key-exchange authentication scheme to establish secure authentication between two previously unknown devices. The key exchange creates a high-entropy shared key from a low-entropy PIN that is transferred through an auxiliary channel. The strong shared key is then used for authentication of exchanged public keys. The scheme protects against both man-in-the-middle and passive eavesdropping attacks, including offline PIN cracking. We focus on Bluetooth version 2.1 and analyze the Simple Pairing protocols. We restrict the supported usage scenarios for the Just Works and Passkey Entry protocols and design a protocol using our proposed solution to replace both protocols. We recognize that our proposed protocol is substantially more secure than the current Just Works protocol, achieves the same security level as the Passkey Entry protocol while maintaining the usability and convenience level for the user. In addition, the proposed protocol considerably reduces the number of messages exchanged compared to the Passkey Entry protocol.

## Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

## General Terms

Security

## Keywords

Wireless personal area networks, auxiliary channel, authentication, Bluetooth, Just Works, Passkey Entry

*QShine'08,* July 28–31, 2008, Hong Kong, China.

## 1. INTRODUCTION

*Wireless personal area networks* (WPANs) are an emerging trend, and establishing connections to devices in close vicinity and accessing services at any time has become increasingly popular. Establishing a WPAN initially involves connecting a device to previously unknown parties. However, there is a fundamental security problem when establishing a communication channel with a previously unknown party. The problem lies in how a user can establish trust that his device is communicating with the correct party the first time communication is established. Several technologies exist in which this problem is addressed by the use of shared secrets. For example, nodes in wireless sensor networks are often preloaded with shared keys [1].

An increasingly popular wireless technology for small mobile devices that currently also relies on shared secrets is Bluetooth. Bluetooth allows two previously unknown devices to establish a relationship in a procedure known as *pairing*. The Bluetooth core 2.0 protocol [2] has been plagued with weaknesses in the pairing and authentication procedures [3, 4, 5, 6, 7], and the new *Simple Pairing* procedure in Bluetooth version 2.1 [8] tries to remedy a number of those weaknesses. The main focus of the Simple Pairing procedure is on the protection against passive eavesdropping and *man-in-the-middle* (MITM) attacks. The Simple Pairing procedure uses an auxiliary channel, consisting of a human user, visual comparison, or an alternative protocol, to improve the security level.

One particularly challenging scenario for establishing secure pairing is when one of the devices lacks sufficient input and output capabilities (e.g., devices with only a button or a LED). The Just Works protocol in Simple Pairing for this scenario is insufficient for creating high-assurance networks since it is vulnerable to simple MITM attacks. To our knowledge there has not been any specific solution proposed to solve the MITM attack vulnerability in Just Works.

Pairing between devices can be achieved by different means depending on the input and output capabilities of the involved devices. There are six basic combinations of device interfaces for the auxiliary channel: limited input and output/limited input and output (-/-), limited input and output/input (-/I), limited input and output/output (-/O), input/input (I/I), output/input (O/I), and output/output (O/O). In this paper, we propose an *auxiliary channel Diffie-Hellman encrypted key-exchange* (ACDHEKE) scheme for authentication in -/I, I/I, and O/I device configurations. A weak secret is transferred over an auxiliary channel and is then used as a key in an encrypted key-exchange. We show how ACDHEKE can be used to establish trust and create a strong shared secret between two previously unknown devices. We focus on Bluetooth as an example and design an ACDHEKE authentication protocol that can be used to replace both the Just Works and Passkey Entry protocols.

The main contributions of the paper are as follows.

- We have analyzed the security for key establishment and authentication in Bluetooth version 2.1, and identified a number of weaknesses in the current protocols.
- We propose an auxiliary channel Diffie-Hellman encrypted key-exchange scheme for authentication of two previously unknown devices that supports -/I, I/I, and O/I device configurations.
- We have designed a protocol for Bluetooth based on the proposed scheme that can replace both the Simple Pairing Just Works and Passkey Entry protocols. It achieves an equal or better level of security while maintaining the usability for the user and significantly reduces the number of messages exchanged over the wireless channel.

The paper is outlined as follows. Section 2 contains related work with proposals to secure pairing and authentication procedures in WPANs. In Section 3, we describe the general concept of the ACDHEKE scheme. Section 4 contains an overview of the Just Works and Passkey Entry protocols and the weaknesses they contain. In Section 5, we analyze the usage scenarios for the Just Works and Passkey Entry protocols to design an improved protocol. Section 6 presents our ACDHEKE protocol and discusses the protection it offers. In Section 7, an analysis of our proposed protocol is provided that describes the achieved protection, practical considerations, and benefits. Section 8 discusses possible future work directions, and Section 9 concludes the paper.

## 2. RELATED WORK

There has been extensive research in pairing and authentication protocols for WPANs. Balfanz et al. [9] expand the idea of the resurrecting duckling protocol [10] to include an exchange of pre-authentication data over a location-limited channel during the first phase. The pre-authentication data is used for subsequent authentication of the involved parties on the wireless link. A proposal to exchange digests of the involved parties' public keys is presented, which requires a bidirectional O/I auxiliary channel; however, the inconvenience level for the user to input such values and support for devices with limited input and output capabilities are not considered. Cagalj et al. [11] propose methods for key agreement and authentication by visual comparison of commitment values or distance bounds. These methods require O/O device configurations. Asokan and Ginzboorg [12] present a solution based on a shared password, where a user chooses a fresh password and shares it among the involved parties. The password is then used to derive a strong shared key. This approach requires I/I device configurations. Nilsson et al. [13] propose a solution for key agreement in O/I device configurations using an authenticated challenge-response procedure over an auxiliary channel. In contrast to [14], the solution requires only a unidirectional auxiliary channel to achieve mutual authentication. Jakobsson [15] presents two methods to verify a symmetric key calculated using DH public parameters exchanged between two parties. The methods are based on a temporary shared secret, i.e., a shared PIN. The established symmetric key is verified using a commitment value calculated with the key, the shared PIN, and optionally a nonce. The methods require a bidirectional O/I auxiliary channel.

The IEEE 802.15.4 specification [16] is a low data-rate WPAN standard and provides basic security mechanisms but does not include any keying model. In other words, cryptographic techniques for authentication, message integrity, confidentiality, and freshness check are supported but techniques for generating, distributing, and managing cryptographic keys are not suggested.

The ZigBee Alliance [17] is an association of companies cooperating to enable reliable, cost-efficient, low-power, wirelessly networked monitoring and control products based on the IEEE 802.15.4 standard. The ZigBee specification [18] contains security mechanisms for key management, e.g., a shared key can be established on two devices or transported from one device to another. Establishing a shared key on two devices, a so called link key, requires some initial trust data on both devices, such as a shared preinstalled master key or a user-entered PIN. If no initial trust data exists, a device can be loaded with an initial key using an insecure transport-key procedure.

The Bluetooth standard version 2.1 Simple Pairing provides four protocols for pairing [19]. In *Numeric Comparison*, both devices calculate a six digit hash of the exchanged public keys and the user is prompted to do a visual comparison and verify that the same values are shown on both devices (O/O). *Just Works* is similar to numeric comparison except that no number values are shown, and the user simply has to accept the connection. Just Works provides the same protection as Numeric Comparison against passive eavesdropping; however, there is no protection against MITM attacks Just Works supports all device configurations, including -/-, -/I, and -/O. *Out of Band* uses, for example, a near field communication (NFC) solution where the user initially touch the two devices together and the cryptographic information is exchanged over the out-of-band channel. Lastly, for *Passkey Entry* the user chooses and enters a six digit number on both devices (I/I). Alternatively, the user is shown a six digit number on one device and inputs the same number on the other device (O/I). The number is used for verifying that the public keys have been exchanged properly.

In this paper, we focus on -/I device configurations and propose a solution that could also be used for I/I and O/I device configurations. We use Bluetooth as an example to provide a lightweight secure pairing protocol.

## 3. AUXILIARY CHANNEL DHEKE

We extend the *Diffie-Hellman encrypted key exchange* [20] (DHEKE)by sending a PIN, inputted, generated, or stored in one device, over an auxiliary channel to the other device. The PIN is then used to establish an encrypted and authenticated key-exchange. We denote this key exchange as *auxiliary channel Diffie-Hellman encrypted key-exchange* (ACDHEKE). A brief and general description follows below, and a detailed description of an ACDHEKE protocol designed for Bluetooth is found in Section 6.

**Table 1: Overview of Protocols**

| Protocol | Required Device Configuration | Pairing Authentication | PIN Revealed | Protection |  | No. of Msgs | Message Exchange (bytes) |  |  |
|---|---|---|---|---|---|---|---|---|---|
|  |  |  |  | PE | MITM |  | Sent | Received | Total |
| Just Works | None/None | None | N/A | X |  | 5 | 56 | 40 | 96 |
| Passkey Entry | Input/Input | Variable PIN (self-selected) | X | X | X[1] | 82 | 664 | 664 | 1328 |
|  | Output/Input | Variable PIN (generated) | X | X | X | 82 | 664 | 664 | 1328 |
| ACDHEKE | None/Input | Fixed PIN |  | X | X | 6 | 64 | 64 | 128 |
|  | Input/Input | Variable PIN (self-selected) |  | X | X | 6 | 64 | 64 | 128 |
|  | Output/Input | Variable PIN (generated) |  | X | X | 6 | 64 | 64 | 128 |

We propose using an auxiliary channel to output and input the PIN to and from the involved devices. This channel can consist of various elements, and in this example, it consists of a human user, a keypad on one device, and a display on the other device. We assume that a network attacker cannot affect the auxiliary channel but can read, modify, and inject traffic in the wireless channel according to the Dolev-Yao model [21]. To preserve the usability for the user, the auxiliary channel can be used to transfer only low-entropy secrets. For O/I and -/I device configurations the auxiliary channel is unidirectional while for I/I device configurations the auxiliary channel is bidirectional. With support from Laur et al. [22] we state that the auxiliary channel for I/I and O/I device configurations must provide both authentication and confidentiality for mutual authentication of the devices. We expand this requirement to include authentication and confidentiality of the auxiliary channel for -/I device configurations.

The principle for ACDHEKE is shown in Figure 1. A PIN is transferred over the auxiliary channel from device $B$ to device $A$. The rest of the protocol is based on DHEKE [20, 23] with the modification that ECDH [24] is used. A temporary public key is encrypted with the PIN and sent over the wireless channel from $A$ to $B$. $B$ replies with another temporary public key and a hash of the public key and the established DH key. The PIN is used for authentication of the two devices during the key establishment.

The major benefits of the ACDHEKE scheme are:

- Unidirectional auxiliary channel for O/I and -/I device configurations to preserve the usability for the user.
- Provides mutual authentication of exchanged public values.
- Supports -/I, I/I, and O/I device configurations without revealing the PIN used.
- Low number of exchanged messages over the wireless channel (encrypted temporary public key, temporary public key, and two commitments).

In the next section, we briefly describe the Just Works and Passkey Entry protocols and the shortcomings that exist. We then analyze the supported usage scenarios for Just Works and Passkey Entry and apply the ACDHEKE scheme to design a new authentication protocol in Section 6.

# 4. OVERVIEW OF PROTOCOLS

We present an overview of the Just Works, Passkey Entry, and our proposed ACDHEKE protocols that shows the requirements, shortcomings, and security levels achieved. The overview is summarized in Table 1. The *Required Device Configuration* column state which input and output capabilities are required by the two involved devices: *Input* requires a keypad to enter six digits, *Output* requires a display to show six digits, and *None* does not require any input or output capabilities beyond one bit, i.e., pressing a button or blinking a LED. *Pairing Authentication* specifies the value used for authentication during the initial relationship establishment. *PIN Revealed* shows if the PIN used in the pairing procedure can be calculated by a passive attacker after the procedure (marked with an $X$). *Protection* indicates whether protection against passive eavesdropping ($PE$) and $MITM$ attacks are fulfilled (marked with an $X$). *No. of Msgs* reveals the number of messages exchanged over the wireless channel, and *Message Exchange* contains the *Sent*, *Received*, and *Total* number of bytes.

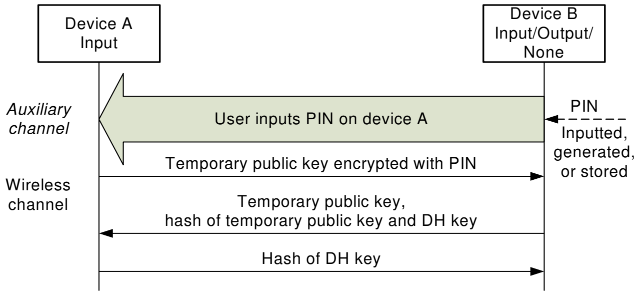

**Figure 1: Auxiliary channel Diffie-Hellman encrypted key-exchange principle.**

## 4.1 Just Works Protocol

The Just Works protocol is similar to the Numeric Comparison protocol [19] with the difference that the six digit values used for visual comparison are not shown. Thus, the user cannot verify that the correct public key values have been exchanged.

### *4.1.1 Man-in-the-Middle Attack*

The Just Works protocol is vulnerable to MITM attacks as identified in the Simple Pairing white paper [19]. An active attacker can inject his own public key value to the attacked devices and establish DH keys with both of the devices.

## 4.2 Passkey Entry Protocol

In the Passkey Entry protocol, a six digit number is inputted by a user in both devices, alternatively, displayed on one device and inputted in the other. The public keys are exchanged and authenticated using the six digit number.

[1] If a PIN is reused, an attacker can successfully perform a MITM attack as described in Section 4.2.1.

### 4.2.1 *Shortcomings*

If a PIN is reused, e.g., a user chooses to use the same PIN for pairing between several devices, which is not unlikely, an attacker can perform a MITM attack [25]. To successfully conduct the attack, the attacker needs to know the PIN that is going to be used in the pairing. The attacker can learn it by passively eavesdropping on a previous pairing procedure or guess it if a trivial PIN, such as "000000", is selected.

In addition to this attack possibility, the Passkey Entry protocol is not energy efficient since a large portion is repeated for 20 rounds. In each round, four messages (two for the commitment values and two for the nonces) are sent over the wireless channel. In addition, two messages are transmitted for the initial public key exchange. Thus, a total of 82 messages are exchanged over the wireless channel during the pairing procedure. Furthermore, two $f1$ functions (one for calculating the commitment value and one for verifying the received commitment value) are calculated in each round. Thus, a total of 40 $f1$ calculations are performed in one device during the pairing procedure. Since these devices often run on batteries, an energy-efficient protocol is desirable. Moreover, since communication over the wireless channel is the most energy-consuming function [26, 27, 28], it is necessary to minimize communications overhead.

# 5. DESIGNING ACDHEKE FOR BLUETOOTH

In this section, we present the design behind the ACDHEKE protocol for Bluetooth. The scenarios supported by Just Works and Passkey Entry are analyzed and adjusted to fit the design of the ACDHEKE protocol.

## 5.1 Analyzing Usage Scenarios for Just Works

The Just Works protocol is primarily designed for -/I device configurations. Such common usage scenarios are pairing a cell phone with a headset or a laptop with a mouse. Just Works also supports -/- and -/O device configurations (such as connecting two headsets or a mouse and a display). However, in practice such scenarios are not realistic. To our knowledge, there exists no practical example usage of such a scenario.

With the support of the usage scenarios found on bluetooth.com [29], we restrict the design of ACDHEKE to support -/I but not -/- and -/O device configurations. The supported scenarios for ACDHEKE are similar to the fixed PIN scenarios in Bluetooth 2.0 [2]. To the best of our knowledge, the restriction does not have any practical implications.

The -/I scenario is particularly challenging since visual comparison or entering a secret value in both devices is not possible. To maintain the convenience level for the user and supported by [18], we propose to preinstall a secret value in the device lacking sufficient input and output capabilities.

We assume that the secret value is a unique and nontrivial six decimal digit PIN. We admit that preinstalling a secret value involves a weakness, which we discuss further in Section 7.3.2; however, we believe that achieving a certain level of security is better than no security at all. One important aspect is that the PIN is not revealed to an attacker after an ACDHEKE pairing procedure as is the case for pairing in Bluetooth 2.0 [6].

**Table 2: Notation used in the following protocol**

| | |
|---|---|
| $Ca$ | Commitment value from device $A$ |
| $DHKEYab$ | DH key between devices $A$ and $B$ |
| $f1'()$ | Function that generates the 128-bit commitment values Ca and Cb for the ACDHEKE protocol |
| $PKa$ | Public key of device $A$ |
| $SKa$ | Secret (private) key of device $A$ |
| $TPKa$ | Temporary public key of device $A$ |
| $TSKa$ | Temporary secret (private) key of device $A$ |
| $Kab$ | Temporary DH key between devices $A$ and $B$ |
| $PIN$ | Six decimal digit PIN stored, generated, or inputted manually into a device |
| $E_K(M)$ | Encryption of message $M$ using the key $K$ |
| $D_K(M)$ | Decryption of message $M$ using the key $K$ |
| $P192(u, V)$ | Function used to calculate a DH key (x-coordinate of the u-th multiple uV of the point V on the elliptic curve P-192 [30]) |

## 5.2 Analyzing Usage Scenarios for Passkey Entry

The Passkey Entry protocol is suitable for I/I and O/I device configurations. Common usage scenarios include pairing a cell phone and a computer, or two PDAs. We design ACDHEKE to support the same configurations as Passkey Entry, i.e., both I/I and O/I are supported.

# 6. ACDHEKE AUTHENTICATION PROTOCOL

The ACDHEKE authentication protocol is designed to replace both Just Works and Passkey Entry, and thus three separate cases exist: -/I, I/I, and O/I device configurations. For -/I, we assume that a PIN is preinstalled in the device. For I/I, a PIN is inputted in both devices, and for O/I, a random PIN is generated and displayed on one device. The notation used in the following protocol is described in Table 2.

The desired security properties for the ACDHEKE protocol are to prevent MITM attacks, and provide protection against passive eavesdropping.

## 6.1 Functions

We have adopted the improved minimal DHEKE II [23] to create a high-entropy shared secret (DH key) from the initial low-entropy PIN. We extend the protocol to include the public key values in the verification. Using the DH key, we can verify that the exchanged public key values are the same on both devices.

We attempt to use the same functions that exist in Bluetooth Simple Pairing. ECDH is used to calculate a temporary DH key used for verifying the public keys exchanged. We also extend the Simple Pairing commitment function $f1$ to include a 192-bit number instead of the 8-bit number as the fourth input value. The original $f1$ function uses this input value only to verify one bit of the PIN. We extend the function to verify a 192-bit temporary public key instead. We call the new function $f1'$. The inputs to $f1'$ are: U (192 bits), V (192 bits), X (128 bits) and Z (192 bits). The output of the $f1'$ function is:

$$f1'(U, V, X, Z) = HMAC - SHA - 256_X(U||V||Z)/2^{128}$$

where || denotes concatenation.

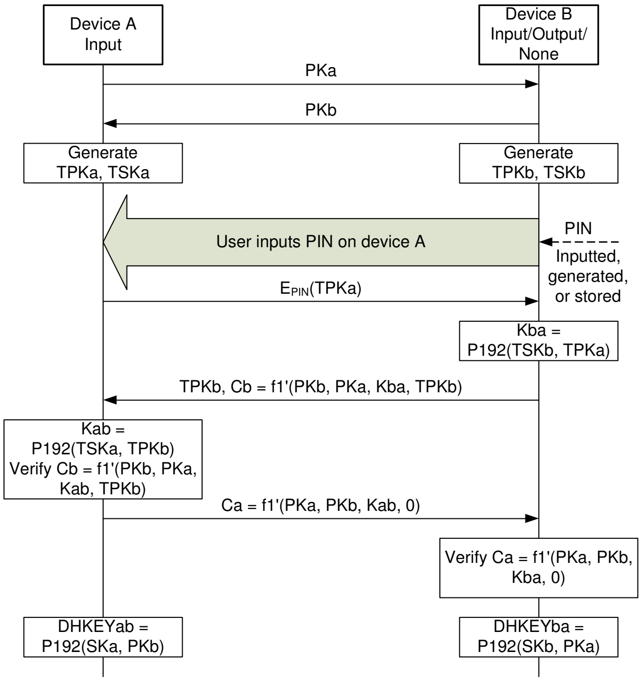

**Figure 2: ACDHEKE protocol.**

## 6.2 Protocol

The proposed protocol is shown in Figure 2. First, the public keys $PKa$ and $PKb$ are exchanged between the two devices. Devices $A$ and $B$ generate temporary ECDH public-private key pairs ($TPKa, TSKa, TPKb, TSKb$). The $PIN$ is sent over the auxiliary channel. $A$ encrypts $TPKa$ with the inputted $PIN$ and sends to $B$ which decrypts the message and calculates the DH key $Kba = P192(TSKb, TPKa)$. The following message, containing $TPKb$ and the commitment value $Cb$ calculated using the public key values $PKb$, $PKa$, the key $Kba$, and $TPKb$, is sent from $B$ to $A$.

Next, $A$ calculates the DH key $Kab = P192(TSKa, TPKb)$ and verifies the received commitment value $Cb$ with the calculated value from the $f1'$ function of the public key values $PKb$ and $PKa$, the calculated key $Kab$, and $TPKb$. Then, a commitment value is calculated on $A$ using $PKa$, $PKb$ and $Kab$ and sent to $B$ which verifies the received message. Thus, mutual authentication that both devices possess the key $Kab == Kba$ is performed. In addition, mutual authentication that both devices share the view of the exchanged public key values is performed. Next, the ECDH calculation is repeated using the initially exchanged public keys ($PKa$ and $PKb$). Identical to Bluetooth version 2.1 [8], a DH key is calculated on both devices. After the last step, both devices are assured that they have exchanged public key values and that a DH key is established between them ($DHKEYab == DHKEYba$).

The temporary public and private keys are discarded after the final DH key is established. It is imperative that $TPKa$ is not revealed as it would allow an attacker to bruteforce the PIN.

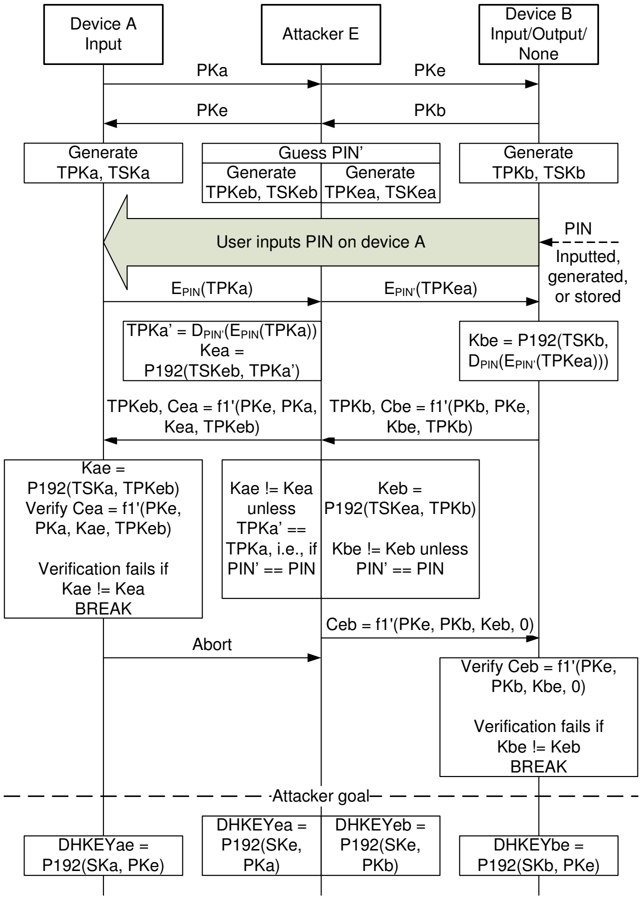

**Figure 3: Man-in-the-middle attack against the ACDHEKE protocol.**

# 7. ANALYZING THE ACDHEKE AUTHENTICATION PROTOCOL

The desired security properties of the ACDHEKE authentication protocol are prevention of the MITM attack that the original Just Works protocol is vulnerable to, and assurance that passive eavesdropping (in particular to calculate the PIN) is still prevented. A security level equal or better to that of the Passkey Entry protocol should be achieved.

## 7.1 Man-in-the-Middle Attack

An active attacker can still inject his own public key value to the attacked devices; however, the attack is not successful since the two attacked devices will detect the MITM attack. The MITM attack flow and the protection mechanism is described as follows.

The objective of the attacker is to create the keys $Kea$ equal to $Kae$ and $Keb$ equal to $Kbe$. If the attacker manages to create these keys, the attacker can successfully create DH keys with both devices and perform the attack. The attack scenario is illustrated in Figure 3. The action on the left-hand side of the attacker is performed for attacking device $A$, and the right-hand side is performed for attacking device $B$. The attacker goal is shown at the bottom in the figure.

First, the attacker $E$ sends its public key value ($PKe$) to the two devices. The attacker then receives the encrypted $TPKa$ value. Since the $PIN$ is unknown to the attacker, he cannot learn the $TPKa$ value. If the attacker simply forwards the message to $B$, a key between $A$ and $B$ is established, and the attacker cannot learn anything. Therefore, the attacker generates two ECDH public-private key pairs ($TPKeb$, $TSKeb$, $TPKea$, $TSKea$), and encrypts $TPKea$ with a guessed $PIN'$. This value is sent to $B$ which calculates the DH key $Kbe$ according to $P192(TSKb$, $D_{PIN}(E_{PIN'}(TPKea))$. $B$ then sends $TPKb$, together with the commitment value of $PKb$, $PKe$, $Kbe$, and $TPKb$.

To continue the attack on $A$, the attacker decrypts the received encrypted $TPKa$ with the guessed $PIN'$ to get $TPKa'$, and calculates the DH key $Kea$ using $TSKeb$ and $TPKa'$. Then, the attacker sends $TPKeb$ together with the commitment value of $PKe$, $PKa$, $Kea$, and $TPKeb$ to $A$.

Next, $A$ calculates the DH key $Kae$ using the received $TPKeb$, and verifies the received commitment value $Cea$ with the calculated commitment value of $PKe$, $PKa$, $Kae$, and $TPKeb$. This matches only if $Kae$ equals $Kea$, which is true only if $TPKa'$ equals $TPKa$, which in turn is true only if the attacker guessed the correct PIN ($PIN'$ equals $PIN$). $A$ aborts if the verification fails and sends an abort message to $B$.

Before the abort message reaches $B$, the attacker sends the commitment value $Ceb$ of $PKe$, $PKb$ and $Keb$ to $B$ which verifies if the received commitment value matches the calculated commitment value of $PKe$, $PKb$ and $Kbe$. This matches only if $Kbe$ equals $Keb$, which is true only if the attacker guessed the correct PIN ($PIN'$ equals $PIN$). $B$ then aborts if the verification fails.

The goal of the attacker is to create DH keys between $A$ and $E$ ($DHKEYae == DHKEYea$) and between $B$ and $E$ ($DHKEYbe == DHKEYeb$). $A$ and $B$ will both detect the attack if the attacker does not know the PIN. Since the PIN is regenerated for devices with output capabilities, the attacker cannot learn this value. Since an attacker cannot learn the PIN after a pairing procedure, the protocol is secure for fixed PIN devices as long as the PIN is not revealed in any other way.

## 7.2 Passive Eavesdropping

The following messages can be learned from passive eavesdropping: $PKa$, $PKb$, $E_{PIN}(TPKa)$, $TPKb$, $f1'(PKb, PKa, Kba, TPKb)$, and $f1'(PKa, PKb, Kab, 0)$. Even if an attacker captures those messages, calculating the key $Kab/Kba$ is hard without knowing either of the private keys because of the elliptic curve discrete logarithm problem. Moreover, using only $E_{PIN}(TPKa)$, $TPKb$ and $f1'(PKb, PKa, Kba, TPKb)$ to bruteforce the PIN is hard because of the fact that $TPKa$ is randomly generated and calculating the DH key is difficult. Even if an educated guess on the PIN is made and $TPKa$ is decrypted, verification is hardly possible because of the randomness involved. Thus, the protocol includes protection against offline PIN cracking. Even if the same PIN is reused a number of times, the messages sent between two devices will be different because of the random temporary public key values, and therefore an attacker cannot learn the PIN. In comparison, using the same PIN more than once in Passkey Entry compromises the security of the protocol. Thus, the ACDHEKE authentication protocol includes protection against passive eavesdropping, and the PIN cannot be calculated offline.

## 7.3 Practical Considerations

There are some practical issues that need further consideration before this protocol can be adopted.

### 7.3.1 Energy Efficiency

The ACDHEKE authentication protocol involves six exchanged messages (two public keys, one encrypted temporary public key, one temporary public key, and two commitment values) during the pairing procedure. In contrast, the Just Works protocol involves five exchanged messages and the Passkey Entry protocol 82 exchanged messages. The ACDEKE protocol considerably reduces the communication overhead in comparison with the Passkey Entry protocol and is comparatively the same for Just Works. Since messages on the radio is most energy-consuming, minimizing the communication overhead is of great importance.

Furthermore, the ACDHEKE protocol comprises one symmetric encryption/decryption operation using the PIN, two ECDH calculations (to calculate the DH key $Kab/Kba$ and $DHKEYab/DHKEYba$), and two $f1'$ functions (for $Ca/Cb$ and verification of $Ca/Cb$) in one device during the pairing procedure. The $f1'$ function is comparable to the $f1$ function in terms of computational efficiency. The Just Works protocol entails only one $f1$ function, one $g$ function, and one ECDH calculation. The Passkey Entry protocol involves 40 $f1$ calculations and one ECDH calculation. ACDHEKE uses the same functions as Simple Pairing, and thus no additional computational requirement is necessary.

### 7.3.2 Fixed and Variable PINs

The ACDHEKE protocol supports both fixed and variable PINs. Variable PIN devices regenerates a PIN for each pairing procedure and thus prevents an attacker from learning the value. In contrast, fixed PIN devices use the same PIN for all pairing procedures. Since fixed PINs are supported, ACDHEKE provides protection against offline PIN cracking to prevent attackers from learning the PIN.

A fixed PIN is appropriate for -/I device configurations, for which the Just Works protocol is suitable. The fixed PIN in the device must be preinstalled and kept secret. The security of the ACDHEKE protocol for such scenarios relies on the secrecy of this value. Moreover, the PIN must be unique and nontrivial. It should be noted that the PIN is used only once during the pairing procedure and is not needed for subsequent authentication between those devices. Therefore, there exists no need for a user to remember the secret PIN (as opposed to, e.g., ATM cards where the PIN is required for every usage). For subsequent pairing of other devices with a fixed PIN device, the same PIN is used, which is similar to fixed PIN device pairing in Bluetooth 2.0. However, in contrast to Bluetooth 2.0 [6], the PIN is not revealed to an attacker after one pairing, and therefore, ACDHEKE is still protected. To our knowledge no other solutions for devices with limited output capabilities to achieve the same level of security exist.

A variable PIN, on the other hand, is appropriate for I/I and O/I device configurations, for which the Passkey Entry protocol is suitable.

We argue that the use of fixed PINs should be restricted to devices that lack output capabilities, and that variable PINs should be used when possible since they achieve a higher level of security.

## 7.4 Benefits

The benefits with the ACDHEKE protocol compared to Just Works are improved security and protection against MITM attacks. Compared to Passkey Entry the benefits are improved security and protection against MITM when the user reuses the PIN. Table 1 contains the benefits compared to Just Works and Passkey Entry. Most importantly, the number of messages exchanged over the wireless channel is reduced from 82 to 6 in comparison with Passkey Entry. Moreover, the PIN is not revealed after a pairing procedure.

The ACDHEKE protocol can replace the Passkey Entry protocol by using variable PINs and replace the Just Works protocol by using fixed PINs. The supported scenarios are restricted to suit the ACDHEKE protocol, and therefore -/- and -/O device configurations are not supported. To our knowledge, these restrictions have no practical implications, and all the usage scenarios on bluetooth.com [29] are still possible with the ACDHEKE protocol.

## 7.5 Analysis of the Protocol in the Formal Model

We have used ProVerif [31, 32], a cryptographic protocol verifier developed by Bruno Blanchet, for formal analysis of the protocol. It can be proved to be secure against an attacker who cannot access the auxiliary channel. Chang and Shmatikov used ProVerif [33] to perform a formal analysis of Bluetooth device pairing, and ProVerif has also been used to perform formal verification of other protocols [34, 35, 36].

A detailed description of ProVerif is found in [32], and a brief description is as follows. In ProVerif, the protocol to be verified is specified in a process calculus, which is an extension of the $\pi$-calculus. Each role in the protocol is represented by a separate process, and the complete protocol is constructed as a model with unbounded sessions of the individual protocol role processes. These processes are automatically translated by ProVerif into a set of Horn clauses, which represent the protocol using abstract algebra. The solving algorithm, which is sound but not complete, uses the Horn clauses as input and determines what an attacker can learn from protocol executions.

ProVerif uses two fundamental principles: the Dolev-Yao attacker model, where the attacker has complete control of the network and the cryptographic algorithms used, and perfect cryptography, where it is impossible to decrypt an encrypted message without knowing the decryption key.

We assume that the attacker cannot access the auxiliary channel. To perform a MITM attack the attacker needs to construct the same DH key or temporary DH key as $A$ and $B$, which would require knowledge of the PIN used. We assume that the PIN is of low-entropy, and using the captured messages over the wireless channel an attacker could attempt to bruteforce the PIN. Under this assumption, the following properties are satisfied.

- An attacker *cannot* learn the value of $PIN$ from the messages exchanged.
- An attacker *cannot* learn the value of $DHKEY$ or the temporary $Kab/Kba$ from the messaged exchanged.

This proof shows that the protocol is secure against an attacker who can read, intercept, inject, and modify messages, as long as the attacker cannot access the auxiliary channel. An attacker cannot learn the value of the PIN used in the ACDHEKE protocol, and can therefore not perform a successful MITM attack. Moreover, the attacker cannot calculate the DH key or the temporary DH key used by $A$ and $B$ both during and after the procedure.

# 8. FUTURE WORK

Using our proposed ACDHEKE authentication protocol, secure authenticated pairing with devices with limited input and output capabilities can be achieved. However, the fixed PIN scenarios involves a weakness, and a way to establish a strong secret in a device with insufficient input and output capabilities is an area that requires further research. If an attacker can learn the value of the fixed PIN, he can compromise that device. Therefore, it is interesting to investigate the possibilities of using variable PINs for -/I device configurations. For example, if the PIN is generated the first time the device is used, the PIN must be revealed to the device owner somehow. This is not trivial considering the limited output interface. For example, for headsets, since a six digit number cannot be displayed, presenting the six digit number through the audio speaker could serve as an alternative. However, this method would potentially involve new hardware and extra costs. Other alternatives are to reset the PIN to a certain factory-default value by pressing a reset button or to set the PIN to a value of the owner's choice. Once again, given the limited input and output capabilities, these alternatives must be properly investigated.

# 9. CONCLUSION

We have proposed an auxiliary channel Diffie-Hellman encrypted key-exchange authentication scheme to establish a PIN-based secure authentication between two previously unknown devices. The key exchange creates a high-entropy shared key from a low-entropy PIN that is transferred over an auxiliary channel. The high-entropy key is then used to verify that the two involved devices possess the exchanged public key values. The scheme protects against MITM attacks since only a device that knows the PIN can perform the authenticated and encrypted key exchange. In addition, it provides protection against passive eavesdropping, including offline PIN cracking, due to the random temporary public keys involved in the key exchange.

In WPANs, establishing trust between previously unknown devices is both necessary and important. We have adopted our authentication scheme to Bluetooth and designed an ACDHEKE authentication protocol that can replace the Simple Pairing Just Works and Passkey Entry protocols. Moreover, our proposed protocol is substantially more secure than the current Just Works protocol and reduces the number of messages exchanged compared to the Passkey Entry protocol while maintaining the usability and convenience level for the user.

# 10. REFERENCES

[1] Seyit A. Camtepe and Bent Yener. Key Distribution Mechanisms for Wireless Sensor Networks: A survey. Technical report, Rensselaer Polytechnic Institute, 2005.

[2] Bluetooth SIG. Bluetooth Specification Version 2.0 + EDR, 2004.

[3] Dennis K. Nilsson, Phillip A. Porras, and Erland Jonsson. How to Secure Bluetooth-Based Pico

Networks. In *The 26th International Conference on Computer Safety, Reliability and Security (SAFECOMP)*, Nuremberg, Germany, 2007.

[4] Albert Levi, Erhan Cetintas, Murat Aydos, Cetin K. Koc, and M. U. Caglayan. Relay Attacks on Bluetooth Authentication and Solutions. In *ISCIS, LNCS 3280*, Kemer-Antalya, Turkey, 2004.

[5] BSI. Bluetooth, Threats and Security Measures. Technical report, BSI, 2003.

[6] Yaniv Shaked and Avishai Wool. Cracking the Bluetooth PIN. In *3rd USENIX/ACM Conf. Mobile Systems, Applications, and Services (MobiSys)*, Seattle, Washington, USA, 2005.

[7] Markus Jakobsson and Susanne Wetzel. Security Weaknesses in Bluetooth. In *CT-RSA, LNCS 2020*, 2001.

[8] Bluetooth SIG. Bluetooth Specification Version 2.1 + EDR, 2007.

[9] Dirk Balfanz, D. K. Smetters, Paul Stewart, and H. Chi Wong. Talking To Strangers: Authentication in ad-hoc wireless networks. In *Symposium on Network and Distributed Systems Security (NDSS)*, 2002.

[10] Frank Stajano and Ross Anderson. The Resurrecting Duckling: Security issues for ad-hoc wireless networks. In *Security Protocols, 7th International Workshop Proceedings*, 1999.

[11] Mario Cagalj, Srdjan Capkun, and Jean-Pierre Hubaux. Key Agreement in Peer-to-Peer Wireless Networks. *Proceedings of the IEEE Special Issue on Cryptography and Security*, 94(2):467–478, 2006.

[12] N. Asokan and Philip Ginzboorg. Key Agreement in Ad Hoc Networks. *Computer Communications*, 23(17):1627–1637, 2000.

[13] Dennis K. Nilsson, Ulf E. Larson, and Erland Jonsson. Unidirectional Auxiliary Channel Challenge-Response Authentication. In *Proceedings of the Seventh Annual IEEE Wireless Telecommunications Symposium (WTS)*, Pomona, CA, USA, 2008.

[14] Jaap-Henk Hoepman. The Ephemeral Pairing Problem. In *Financial Cryptography*, Key West, Florida, USA, 2004.

[15] Markus Jakobsson. Lecture Notes in I400/I590: Issues in Security and Privacy. `http://www.informatics.indiana.edu/markus/i400/`. Visited October, 2007.

[16] IEEE Standards. 802.15.4 Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), 2003.

[17] ZigBee. ZigBee Alliance. `http://www.zigbee.org`.

[18] ZigBee Standards Organization. ZigBee Specification, 2006.

[19] Bluetooth SIG. Simple Pairing Whitepaper, 2006.

[20] Steven M. Bellovin and Michael Merritt. Encrypted Key Exchange: Password-based protocols secure against dictionary attacks. In *IEEE Symposium on Security and Privacy*, pages 72–84, May 1992.

[21] Danny Dolev and Andrew C. Yao. On the Security of Public Key Protocols. In *IEEE 22nd Annual Symposium on Foundations of Computer Science*, Stanford, CA, USA, 1981.

[22] Sven Laur, N. Asokan, and Kaisa Nyberg. Efficient Mutual Data Authentication Using Manually Authenticated Strings. Cryptology ePrint Archive, Report 2005/424.

[23] Chun-Li Lin, Hung-Min Sun, and Tzonelih Hwang. Efficient and Practical DHEKE Protocols. *SIGOPS Oper. Syst. Rev.*, 35(1):41–47, 2001.

[24] National Institute of Standards and Technology. Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography. NIST Special Publication 800-56A, March 2007.

[25] Jani Suomalainen, Jukka Valkonen, and N. Asokan. Security Associations in Personal Networks: A Comparative Analysis. In *Security and Privacy in Ad-hoc and Sensor Networks 4th European Workshop (ESAS)*, Lecture Notes in Computer Science, Cambridge, UK, 2007.

[26] Adrian Perrig, Robert Szewczyk, Victor Wen, David E. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Mobile Computing and Networking*, pages 189–199, 2001.

[27] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, Baltimore, November 2004.

[28] Mark Luk, Ghita Mezzour, Adrian Perrig, and Virgil Gligor. MiniSec: A secure sensor network communication architecture. In *IPSN '07: Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pages 479–488, New York, NY, USA, 2007. ACM Press.

[29] bluetooth.com. Bluetooth Usage Scenarios for Play. `http://www.bluetooth.com/Bluetooth/Connect/Play/Scenarios/`, 2007.

[30] U.S. Department of Commerce/National Institute of Standards and Technology. Digital Signature Standard (DSS), 2000.

[31] Bruno Blanchet. Analysis of Cryptographic Protocols in the Formal Model. http://www.di.ens.fr/ blanchet/crypto-eng.html.

[32] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96, Cape Breton, Nova Scotia, Canada, June 2001. IEEE Computer Society.

[33] Richard Chang and Vitaly Shmatikov. Formal Analysis of Authentication in Bluetooth Device Pairing. In *LICS/ICALP Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA)*, Wroclaw, Poland, July 2007.

[34] Riccardo Bresciani. The ZRTP Protocol: Security Considerations. Technical report, LSV, ENS Cachan, France & Scuola Superiore Sant'Anna, Italy, 2007.

[35] Steve Kremer. Formal Verification of Cryptographic Protocols. Invited tutorial, 7th School on Modelling and Verifying Parallel Processes (MOVEP'06), Bordeaux, France, June 2006.

[36] Erik Angelin. Verifying IKEv2 using ProVerif. http://www.imit.kth.se/courses/2G1517/03-04/03-04/Erik/IKEv2.pdf,
2004.