Uppsala Master's Thesis in
Computer Science
Examensarbete DV3
30th January 2006

Secure Drag&Drop Key-exchange

Henrik Andersson

Information Technology
Computer Science Department
Uppsala University
Box 337
S-751 05 Uppsala
Sweden

Abstract

This master thesis describes the technology and an implementation of the Secure Drag&Drop Key-Exchange system, a system that will solve the problem with distribution of key material between devices in an authentic, intuitive and easy to understandable way for a non-expert user. This system involves a camera-equipped mobile phone that will act as a security manager to do this distribution in a "drag & drop" fashion when two devices want secure communication.

Supervisor: Christian Rohner
Examiner: Mats Björkman

Contents

1 Introduction

Secure key-exchange between devices is an active field of research today. The problem in key-exchange is how to do authentic transport of key material from one device to another. The transportation should also be intuitive and easily understandable for a non-expert user and it should involve as little interaction as possible from the user. The user should easily be able to setup a secure communication between two devices. It is often the case that it requires expert users to setup this communication and that it involves much interaction from the user and that it is not easy to understand.

Since more and more mobile phones are being equipped with cameras new fields of research become available. The computational power of the mobile phones are increasing and they are often equipped with wireless techniques like bluetooth and almost everyone has one in their pocket.

A recently investigated field is to use the camera of a camera-equipped mobile phone as a visual code reader to interact with physical objects [5, 6, 13, 14, 15].

The use of this visual code reading technology to transport key material between two devices has also been investigated. The key material can be public keys presented as visual codes that the two devices that want to communicate read from each other using their cameras as visual code readers [3].

In this thesis we investigate a more general approach, also using a camera-equipped mobile phone, to transport key material between two devices. Here we use the camera-equipped mobile phone as a security manager to take care of the key distribution. We only use existing techniques and hardware and the intuitive concept of "drag & drop" to transport key material from one device to another to make it easy to understand for the user.

1.1 Problem description

When two devices want to communicate in a secure manner key material, like public keys, has to be exchanged between the devices. The problem is how to distribute this key material in an authentic, intuitive and easily understandable way so that non-expert users can setup and configure a secure communication between the two devices with as little interaction as possible from the user.

1.2 Goals and methods

The main goal with this thesis is to design and investigate an approach to exchange key material between two devices in an authentic, intuitive and

easily understandable way. The design should use existing hardware and techniques to accomplish the key exchange. It should be easy for the user to do such exchange and involve as little interaction as possible from the user.

Experimental methods have been used where ideas have been discussed, evaluated and tested to come up with a suitable solution. The final solution have been implemented to show "proof-of-concept".

1.3 Thesis structure

Chapter 2 describes some background material that is needed for this thesis. First a background on some security issues is presented that will clarify the security terminology used later. Related solutions and an introduction to the Fiat-Shamir authentication protocol will also be presented and discussed. In Chapter 3 we will get to know the concept of Secure Drag&Drop Key-Exchange. First a description of the main ideas, then a discussion about what kind of devices that are suitable for this concept and then a typical scenario will be presented to clarify the use of it. Chapter 4 will be dedicated to a security evaluation of the algorithm. We will also have a look at possible attacks and how to prevent them. A presentation of the implementation of the algorithm will follow in Chapter 5. Finally Chapter 6 will sum up and conclude the thesis.

2 Background

This chapter begins with a background where we present a calculus for security channel transformations and we define confidentiality and authenticity in the context of channels. In Section 2.2 we will introduce the Fiat-Shamir authentication protocol that is being used in the implementation of the Secure Drag&Drop Key-Exchange algorithm for proving knowledge of secret keys. Last in this chapter we present some related work on visual codes.

2.1 Security background

In this section we will present a calculus and a notation of security channels[1] that we will use in Section 4.1 when we verify the security of the system. This calculus and the notation of security channels are based on the work of Maurer and Schmid [2].

A communication channel, from A to B, can be viewed as a means for transporting a message from the source A, the channels input, to the destination B, the channels output. A channel can provide authenticity and/or confidentiality. A channel that provides confidentiality has its output exclusively accessible to the receiver and a channel that provides authenticity has its input exclusive accessible to the sender. A more formally definition would look like:

- **Confidentiality:** A channel provides confidentiality if its *output* is exclusively accessible to a specified receiver and this fact is known to (or believed by) a sender on the channel.
- **Authenticity:** Similarly, a channel provides authenticity if its *input* is exclusively accessible to a specified sender and this fact is known to (or believed by) a receiver.

Since authenticity and confidentiality are independent we can distinguish four types of channels based on these fundamental security properties. Channels with none of the security properties, channels with just authenticity, channels with just confidentiality and channels with both authenticity and confidentiality. A channel is denoted by the symbol $\longrightarrow$ and the symbol $\bullet$ attached to one side of the channel symbol indicates that the user at the corresponding end of the channel has exclusive access to the channel. The four types of channels are:

[1] A channel is a path between two endpoints. In communication, a channel is the path a message follows from the sender to the receiver. A channel can be a wire, the air, the sight etc.

$A \longrightarrow B$	channel that provides no security
$A \longrightarrow\!\!\bullet\ B$	channel that provides confidentiality but not authenticity
$A \bullet\!\!\longrightarrow B$	channel that provides authenticity but not confidentiality
$A \bullet\!\!\longrightarrow\!\!\bullet\ B$	channel that provides both confidentiality and authenticity

The security symbol $\bullet$ in the channel $A \longrightarrow\!\!\bullet\ B$ refers to A's belief that information sent on this channel will only be available to B. Similarly the security symbol $\bullet$ in the channel $A \bullet\!\!\longrightarrow B$ refers to B's belief that information received on this channel originates from A.

There exist several security transformations where we can transfer security properties from initially available channels to insecure channels. These transformations are available if we use some of the cryptographic primitives like symmetric encryption, message authentication code, public-key cryptosystem or digital signature scheme. The following transformation rules are a selection from all the available ones and they will be used when we verify the security in Section 4.1.

$$\left.\begin{array}{c} A \bullet\!\!\xrightarrow{t_1} B \\ A \xrightarrow{t_2} B \\ t_2 \geq t_1 \end{array}\right\} \implies A \bullet\!\!\xrightarrow{t_2} B \tag{1}$$

$$\left.\begin{array}{c} A \bullet\!\!\xrightarrow{t_1} T \\ T \bullet\!\!\xrightarrow{t_2} B \\ t_2 > t_1 \\ B \text{ trusts } T \end{array}\right\} \implies A \bullet\!\!\xrightarrow{t_1\ t_2} B \tag{2}$$

Transformation rule (1) can be used to transfer the authenticity of the channel t_1 to the insecure channel t_2 if we use digital signatures.

Over the authentic channel t_1 A can send her public key to B. B will now have A's public key and he can decrypt messages encrypted with A's private key which only A has access to. Hence he knows that messages encrypted with A's private key originates from A.

A uses her private key to generate a digital signature on a message which she sends to B over the insecure channel t_2. With the public key of A, received earlier over the authentic channel, B can verify that the message sent over the insecure channel t_2 originates from A.

Transformation rule (2) can be used if we introduce trust in the model. Trust is often a fundamental ingredient for secure communications in large

distributed systems. If B trusts T and we have an authentic channel from A to T and an authentic channel from T to B we can connect these authentic channels and end up with an authentic channel from A to B. Notice that B have to trust T to send only authenticated information.

2.2 Fiat-Shamir authentication protocol

The Fiat-Shamir authentication protocol is a 3-pass, challenge-response, protocol where the prover A proves, in t rounds, knowledge of the secret s to the verifier B. This must be done without A revealing any information to B , about the secret s, other than information already known or computable to B. B does not learn anything about the secret under the protocol rounds and B can not use information retrieved to successfully convince A that he also knows the secret. This is called a Zero-knowledge protocol since it, after t successful rounds, gives B "zero-knowledge" about the secret.

There are two initialization steps in the protocol. First a trusted-third-party selects n which is the product of two large prime numbers p and q. This n is public and is used in the protocol by both A and B. Second A selects the secret s with the following proterties, $gcd(s, n) = 1$ and $1 \leq s \leq n-1$. Then A computes $v = s^2 \bmod n$, and registers v with the trusted-third-party. In each of the t rounds the following messages are passed between A and B.

$$
\begin{aligned}
A \longrightarrow B: &\quad x = r^2 \bmod n\ , \quad 1 \leq r \leq n-1 \\
A \longleftarrow B: &\quad e \in \{0, 1\} \\
A \longrightarrow B: &\quad y = r \cdot s^e \bmod n
\end{aligned}
$$

Before A sends x to B, A chooses (*commitment*) r, $1 \leq r \leq n-1$, and calculates (*witness*) $x = r^2 \bmod n$. When B has recieved x, B sends (*challenge*) $e = 0$ or $e = 1$. Depending on the value of e, A responds to B with (*response*) $y = r \bmod n$ (if $e = 0$) or $y = r \cdot s \bmod n$ (if $e = 1$). Now B tries to verify that $y^2 \equiv x \cdot v^e$ (mod n) and B rejects if that is not the case or if $y = 0$, otherwise B accepts.

The zero-knowledge protocols use asymmetric techniques but do not rely on digital signatures or public-key encryption, and which avoid use of block ciphers, sequence numbers, and timestamps [1].

Why these protocols, especially the Fiat-Shamir protocol, are interesting to us is because it does not involve any complicated computations and the strength can be reduced, run the protocol in less rounds, if the computational power is a limited factor. Since we are using small wireless devices computational power is a limited factor.

To verify our claim, that small wireless devices have limited computational power, we made some computational tests on a typical small wireless device. How these tests were performed can be viewed in Appendix A.

2.3 Visual codes

There exists several projects where the use of a camera-equipped mobile phone for recognizing and reading visual codes to interact with physical objects are investigated.

Hanna presents several approaches to establish security parameters between devices in a smarthome. One approach is to use visual codes to establish these security parameters [7].

McCune, Perrig and Reiter has developed the concept of "Seeing-Is-Believing" where two camera-equipped mobile phones can authenticate each other using camera and display [3]. User A's mobile phone generates a visual code in the display that user B takes a snapshot of using his camera-equipped mobile phone. Using the cryptographic material encoded in the visual code B can authenticate A. The procedure can of course be done in the other direction so that A authenticate B.

Woodside developed semacodes which uses the data matrix standard for mobile phones [13]. Here a URL is encoded into a visual code that can be read by a camera-equipped mobile phone and associated data can be retrieved.

Rohs and Gfeller developed their own 2-dimensional visual code for use with mobile phones [6]. These codes are suitable to display on both electronic screens or on paper. This code system encodes 78 bits and without the error correction codes 83 bits. The code system also has a guide bar that is used to locate codes in an image.

CyberCode is another 2-dimensional visual code system and it is developed by Rekimoto and Ayatsuka at Sony Computer Science Laboratories in Tokyo. This system is designed to be recognized by low quality cameras found in mobile phones. This code system encodes 24 or 48 bits, excluding error correction bits. With these bits an ID can be encoded that the camera-equipped mobile phone can recognize and identify a physical object. The ID can then be sent to a server, since most mobile phone have network access, and related data can be retrieved [15].

Scott, Sharp, Madhavapeddy and Upton presented the use of SpotCodes to enhance human-computer interaction [5]. With SpotCodes a camera-equipped mobile phone could be used as a remote control or a mouse to interact with a computer when reading codes. These visual codes can be applied on both active displays, like computer displays or plasma displays,

or on passive displays like on a piece of paper.

Researchers at Korea's Yonsei University developed ColorCodes which is a visual code that consists of colors. The colors that a ColorCode can consist of are red, green, blue and black. With a 5×5 cell code, like the one in Figure 1, more than 17 billion patterns can be created. In ColorCodes system each of these patterns are linked to digital content on a server, called content platform. With a camera-equipped mobile phone the user can recognize a color pattern and send it to the content platform which respond with the content linked to the pattern [14].

Figure 1: From the left: ColorCode, Rohs and Gfellers visual code, Semacode and SpotCode.

As you can see there are several ongoing projects that investigate the use of camera-equipped mobile phones as visual code readers for different purposes.

3 Secure Drag&Drop key-exchange algorithm

In this chapter the algorithm will be presented. First a general idea description and then a typical scenario that will demonstrate the use of it. In the last section different types of devices will be discussed and how these can use the algorithm depending on their properties.

3.1 Idea description

The main idea behind Secure Drag&Drop Key-Exchange (SDDKE) is that we want to use a camera-equipped mobile phone as some kind of security manager to initiate a secure communication between two devices. We want to use a "drag & drop" procedure to transport key material between the two devices that want secure communication. The user should get the feeling of "drag & drop" when distributing that material. The security manager will use the camera as a visual code reader to read visual codes attached to the two devices so that the identity of the two device can be established. The following scheme will describe the steps involved in the algorithm.

1. The security manager reads the visual code of device A to retrieve the identity of A. (This step is the "drag" step where the user presses down the capture button to capture the visual code of A and holds it down.)
2. The security manager reads the visual code of device B to retrieve the identity of B. (This is the "drop" step where the user lets go of the capture button when he sees the visual code of device B in the display and that visual code is also captured.)
3. The security manager connects to A and asks for the public key of A.
4. A responds and sends her public key to the security manager.
5. The security manager connects to B to ask for the public key of B and to deliver the public key of A and the address of A (received in step 4 and step 1).
6. B responds and sends his public key to the security manager.
7. The security manger sends the public key of B and the address of B to A (received in step 6 and step 2).
8. A connects to B and tries to prove her identity, the knowledge of the secret key that belongs to her public key (that B received in step 5).

9. B tries to prove his identity to A, the knowledge of the secret key that belongs to his public key (that A received in step 7).

A more detailed and technical description of the algorithm will be presented in Chapter 5 where the implementation will be discussed. To clarify the use of SDDKE even more a typical scenario will now follow that will demonstrate a real world example.

3.2 Typical scenario

Figure 2: A typical scenario when the Secure Drag&Drop Key-Exchange could be used.

Figure 2. will demonstrate a typical scenario where the Secure Drag&Drop Key-Exchange algorithm could be used. This scenario could be possible in, for example, a smarthome.

Imagine that you have your mp3 collection on the laptop and you would like to listen to some music. Your laptop has poor speakers so you decide that the stereo should play the music instead. Since both your laptop and your stereo has bluetooth wireless radios built-in, the music will be sent over the wireless medium instead of over a wire. Since it is illegal to share copyrighted music with others you would not like someone to eavesdrop on the communication between the computer and the stereo.

To initialize the communication you take the camera-equipped mobile phone and point the camera to the visual code of the computer. When the visual code is visible in the display you press the capture button on the mobile phone and then you point the camera to the visual code of the stereo. When you see that visual code in the display you release the capture button

and the initialization phase is done. Now both the laptop and the stereo has all the needed information to secure communication.

In the following section different types of devices will be discussed and we will present our own definition of "smart" and "primitive" devices.

3.3 Devices

There exists many types of devices with different properties. The ones that we are interested in, in this thesis, are devices with some kind of wireless communication abilities, like bluetooth. In that category we have everything from advanced devices like computers to "primitive" devices like sensors. These devices are equipped with strong or poor computational power, some have displays and some do not, some have keyboards for input and others lack of it. These properties allow us to make a difference between "smart" and "primitive" devices. This is our definition of a "smart" and a "primitive" device:

- **Smart device:** A device that has *strong* computational power, wireless communication abilities, a display and keyboard for input.
- **Primitive device:** A device that has *poor* computational power, wireless communication abilities, no display and no keyboard for input.

Of course there can be devices with wireless communication abilities that do not fit into either of these two definitions, but the definitions are only presented to define the interval of interesting devices.

A sensor could be a primitive device in our definition if the computational power is enough to use the cryptographic primitives involved. Since we are using Fiat-Shamir authentication protocol in this solution it will not require so much computational power form the device so it would be possible to use even sensors.

4 Algorithm evaluation

In this chapter the algorithm will be evaluated. The evaluation will be done on the basis of security. In the first section a security verification will be performed. Attacks will also be presented in this chapter and ways to prevent them will be discussed. Conclusions about this chapter will be in Section 4.3 and that will conclude this chapter.

4.1 Security verification

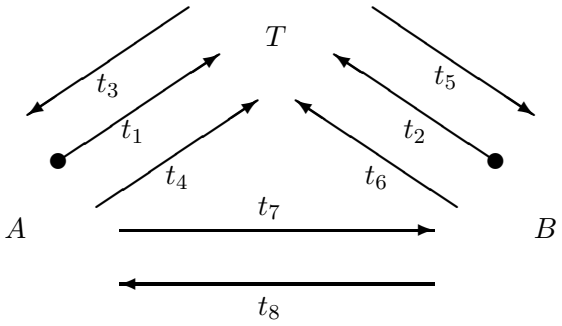

Figure 3: The available channels between the two devices A and B and the security manager T.

In this section we will verify the security of the Secure Drag&Drop Key-Exchange system. To clarify the verification we will use the following notation where T will be the security manager and A and B will be the other two devices. The following scheme describes the available channels in the system:

$A \bullet\xrightarrow{t_1} T$	Visual channel available through visual code reading. (Step 1. in algorithm see Section 3.1)
$T \xleftarrow{t_2}\bullet B$	Visual channel available through visual code reading. (Step 2. in algorithm see Section 3.1)
$A \xleftarrow{t_3} T$	Bluetooth channel available through t_1 channel. (Step 3. and step 7. in algorithm see Section 3.1)
$A \xrightarrow{t_4} T$	Bluetooth channel available through t_3 channel. (Step 4. in algorithm see Section 3.1)
$T \xrightarrow{t_5} B$	Bluetooth channel available through t_2 channel. (Step 5. in algorithm see Section 3.1)
$T \xleftarrow{t_6} B$	Bluetooth channel available through t_5 channel. (Step 6. in algorithm see Section 3.1)
$A \xrightarrow{t_7} B$	Bluetooth channel available through t_3 channel. (Step 8. in algorithm see Section 3.1)
$A \xleftarrow{t_8} B$	Bluetooth channel available through t_5 channel. (Step 9. in algorithm see Section 3.1)

The final goal with this verification is to transform the two insecure channels t_7 and t_8 into authentic channels. The verification will be done in several steps were each channel will be transformed into an authentic channel using the transformation rules presented in Section 2.1. The two visual channels t_1 and t_2 will not be transformed since they already are authentic channels. We consider these channels authentic since the user of T can identify both A and B visually when aiming the camera at the desired device.

We will start the transformations with the insecure channel t_4. This is because we have the authentic channel t_1 which we will use.

4.1.1 Transformation of t_4 channel

The bluetooth channel t_4 is an insecure channel and we need to make it authentic to be able to use it in transformations of other channels. This can be done using the authentic visual channel t_1 and the insecure bluetooth channel t_3.

What is happening in step 1 of the algorithm (see Section 3.1) is that T gets the address of A over the authentic visual channel t_1. Hence T will be sure that the address is indeed the address of A. In step 3 T connects to A and asks for the public key of A. T receives the public key in step 4. The following scheme illustrates what is happening:

$$
\begin{array}{lll}
A \overset{t_1}{\bullet\!\!\longrightarrow} T & \text{Visual:} & Addr_A \\
A \overset{t_3}{\longleftarrow} T & \text{Bluetooth:} & \textit{"Hello"} \\
A \overset{t_4}{\longrightarrow} T & \text{Bluetooth:} & PK_A
\end{array}
$$

Here we can argue that, since T got the address of A over the authentic channel, T knows that the device he is trying to connect to using $Addr_A$ is indeed A and hence the public key, PK_A, received from that address is indeed A's public key. Hence a transformation would be possible.

This is a pretty weak assumption since there is the risk of bluetooth address spoofing and then T can not be sure that he is connecting to A, although he has A's address, hence PK_A would not be authentic.

A way to make the receiving of PK_A more secure could be to receive it over the visual channel as well. If we use the visual channel for both the bluetooth address and the public key then T would not have to receive anything over the insecure bluetooth channel.

The problem with this approach is that we must have two visual codes, one with the bluetooth address and one with the public key, or one more advanced visual code that could present both.

If we would like to receive more authentic information from A, other than the bluetooth address and the public key, we could transform the insecure bluetooth channel into an authentic channel using digital signatures. Information sent over channel t_4 could be signed with a digital signature of A that T could verify using the public key PK_A. The following scheme illustrates the scenario:

$$
\begin{array}{lll}
A \overset{t_1}{\bullet\!\!\longrightarrow} T & \text{Visual:} & PK_A,\ Addr_A \\
A \overset{t_3}{\longleftarrow} T & \text{Bluetooth:} & \textit{"Hello"} \\
A \overset{t_4}{\longrightarrow} T & \text{Bluetooth:} & m,\ SIG_A(m)
\end{array}
$$

If we use the argument that the bluetooth address, received through visual code reading, is enough to make the insecure bluetooth channel authentic or if we use digital signatures, a formal transformation using rule (1) presented in Section 2.1 would look like:

$$
\left.
\begin{array}{c}
A \overset{t_1}{\bullet\!\!\longrightarrow} T \\
A \overset{t_4}{\longrightarrow} T \\
t_4 > t_1
\end{array}
\right\}
\quad \Longrightarrow \quad
A \overset{t_4}{\bullet\!\!\longrightarrow} T
$$

The insecure bluetooth channel t_4 has now been transformed into an authentic bluetooth channel. We will continue this verification by looking at channel t_6 which has the same conditions as channel t_4.

4.1.2 Transformation of t_6 channel

As with the channel t_4 this channel also has to be transformed into an authentic channel. The procedure will be much like the one with t_4. Here we will be using the authentic visual channel t_2 and the insecure bluetooth channel t_5.

What is happening in step 2 of the algorithm (see Section 3.1) is that T gets the address of B over the authentic visual channel t_2. Hence T will be sure that the address is indeed the address of B. In step 5, T connects to B and asks for the public key of B. T receives the public key in step 6. The following scheme illustrates what is happening:

$$\begin{array}{lll} T \overset{t_2}{\longleftarrow\!\bullet} B & \text{Visual:} & Addr_B \\ T \overset{t_5}{\longrightarrow} B & \text{Bluetooth:} & \textit{"Hello"} \\ T \overset{t_6}{\longleftarrow} B & \text{Bluetooth:} & PK_B \end{array}$$

As we pointed out earlier this is a weak assumption and it could be compromised by bluetooth address spoofing. A way to prevent this would be to use digital signatures, as we saw in previous section. Then B can sign information sent over the insecure bluetooth channel with his signature that T could verify using the public key PK_B. The following scheme illustrates the use of digital signatures:

$$\begin{array}{lll} T \overset{t_2}{\longleftarrow\!\bullet} B & \text{Visual:} & PK_B,\ Addr_B \\ T \overset{t_5}{\longrightarrow} B & \text{Bluetooth:} & \textit{"Hello"} \\ T \overset{t_6}{\longleftarrow} B & \text{Bluetooth:} & m,\ SIG_B(m) \end{array}$$

If we use the same arguments as before, with the channel t_4, that the bluetooth address is enough to make the insecure bluetooth channel authentic or if we use digital signatures, a formal transformation using rule (1) presented in Section 2.1 would look like:

$$\left.\begin{array}{c} T \overset{t_2}{\longleftarrow\!\bullet} B \\ T \overset{t_6}{\longleftarrow} B \\ t_6 > t_2 \end{array}\right\} \implies T \overset{t_6}{\longleftarrow\!\bullet} B$$

The insecure bluetooth channel t_6 has now been transformed into an authentic bluetooth channel. We will continue this verification by looking at the two insecure channels we have left between the security manager and the two devices.

4.1.3 Transformation of t_3 and t_5 channels

These two insecure channels can not be transformed into authentic channels using the formal transformation rules in Section 2.1. This is because we do not have any other channels from T to A or from T to B to use that are authentic. Since we can not transform these insecure channels we have to see if we could make them authentic from the beginning using some technique.

One simple way to solve this is to make use of the fact that the authenticity refers to the receiver's state of belief. If the receiver believes, trusts, that information is sent by a specified sender, and accepts that information, then it can be seen as authentic. If the receiver does not believe the information to be authentic he can just reject it.

In practice we could have a display or some other type of output on the device where information can be presented that the user of the device can accept or reject. The information presented could be, in this case, the bluetooth address of the security manager or some other information that could be used to prove the identity of the security manager. If the user of the device believes that it is indeed the security managers bluetooth address then he could accept the communication and be able to retrieve the public key of the security manager, hence be able to transform the insecure bluetooth channel into an authentic channel.

This solution is very simple but it will not be so convenient to use. Every time when some secure communication will be established, the user have to make a decision whenever to accept or reject the communication by for example pressing a button. The devices must have displays or some other type of output for the user to be able to make the decision to accept or reject. This solution is, in other words, not suitable for primitive devices (see Section 3.3).

Another way could be to let all devices to have a secret key stored in them during manufacturing. This secret could be printed as a visual code on a paper shipped with the device. The one knowing that secret would be treated as the owner of the device, be trusted and be an authenticated user.

When the user buys a new product he can use the security manager to read the visual code on the paper, shipped with the device, to get the secret key. Then the security manager will be authenticated with the new device since he can encrypt information with that key which the device then can verify. The one knowing the key will be treated as owner, hence be authenticated.

There are some problems with this solution. First the secret key has to be exclusive to the owner. If someone else get their hands on the secret key they will also be authenticated as the owner. When the user of the device

have read the visual code with his security manager the document with the secret key have to be stored in a safe place. How this could be done will not be discussed in this thesis.

There are other problems associated with this approach. If the secret key can not be changed there could be some serious problems if the paper with the secret key is lost. Also if someone unauthorized person get to know the secret key there would be security problems. If the device is sold to a new owner the new owner would not like the previous owner to know the key since that would be the same as an unauthorized person knowing it. If the key should be changeable there are issues like who can change the secret key and how to read the new key.

We could also think of having this secret key hidden on the device like the approach with the paper. This approach would be more convenient to use but it would not be so good in a security perspective [7].

Another interesting approach would be to let all devices to be associated with a master device, like the security manager. A device will only obey and trust the security manager and will only accept security information from the security manager, hence the security manager can be authenticated.

The association could be done by letting the first one to present a secret key to the device to be its master device. The secret key could be presented as a visual code on the device so that the security manager could read it through the camera. The device will obey and trust the security manager until the device is reset. That could be done by just pressing a reset button on the device [8].

With this approach we do not have to care about what have happened to the device before we buy it. If someone has taken control over the device, by presenting the secret key for the device, then we could just reset the association by pressing the button and we could associate it with our security manager instead. To take control over the device someone must have physical contact with the device to be able to read the visual code. It is not enough to be in range of the bluetooth radio. This will make an security attack from an evil neighbor hard.

The important thing here is that it is possible to make the insecure channels t_3 and t_5 authentic using one of the presented techniques. The verification will now continue with the transformation of the insecure bluetooth channel t_7.

4.1.4 Transformation of t_7 channel

After the previous transformations and discussions we have the following channels available:

$A \overset{t_1}{\bullet\!\!\longrightarrow} T$ $\quad$ $T \overset{t_2}{\longleftarrow\!\!\bullet} B$ $\quad$ $A \overset{t_7}{\longrightarrow} B$

$A \overset{t_3}{\longleftarrow\!\!\bullet} T$ $\quad$ $T \overset{t_5}{\bullet\!\!\longrightarrow} B$ $\quad$ $A \overset{t_8}{\longleftarrow} B$

$A \overset{t_4}{\bullet\!\!\longrightarrow} T$ $\quad$ $T \overset{t_6}{\longleftarrow\!\!\bullet} B$

Now we have an authentic bluetooth channel from A to T, using channel t_4, and an authentic bluetooth channel from T to B, using channel t_5. Authentic information can now be sent from A to B through T. If we apply the formal rule (2) in Section 2.1 on these two channels we would get:

$$\left.\begin{array}{c} A \overset{t_4}{\bullet\!\!\longrightarrow} T \\ T \overset{t_5}{\bullet\!\!\longrightarrow} B \\ t_5 > t_4 \\ B \text{ trusts } T \end{array}\right\} \implies A \overset{t_4\ t_5}{\bullet\!\!\longrightarrow} B$$

Now we have this authentic bluetooth channel from A to B through T and an insecure bluetooth channel from A to B, using channel t_7. Using the same arguments as before, when we transformed the insecure bluetooth channels t_4 and t_6 into authentic channels, we could use the formal transformation rule (1) in Section 2.1 and end up with:

$$\left.\begin{array}{c} A \overset{t_4\ t_5}{\bullet\!\!\longrightarrow} B \\ A \overset{t_7}{\longrightarrow} B \\ t_7 \geq t_5 \end{array}\right\} \implies A \overset{t_7}{\bullet\!\!\longrightarrow} B$$

This transformation could be done since we have transferred PK_A to B over the authentic channel $A \overset{t_4\ t_5}{\bullet\!\!\longrightarrow} B$ and B can use that public key to verify information signed and sent by A over the bluetooth channel t_7. Now we have transformed the insecure bluetooth channel from A to B into an authentic channel using the trusted security manager T.

The only thing left is to argue that we can do the same transformations with the t_8 channel and that will be discussed in the next section.

4.1.5 Transformation of t_8 channel

From the previous transformations we have the following channels available:

$A \overset{t_1}{\bullet\!\!\longrightarrow} T$ $\quad$ $T \overset{t_2}{\longleftarrow\!\!\bullet} B$ $\quad$ $A \overset{t_7}{\bullet\!\!\longrightarrow} B$

$A \overset{t_3}{\longleftarrow\!\!\bullet} T$ $\quad$ $T \overset{t_5}{\bullet\!\!\longrightarrow} B$ $\quad$ $A \overset{t_8}{\longleftarrow} B$

$A \overset{t_4}{\bullet\!\!\longrightarrow} T$ $\quad$ $T \overset{t_6}{\longleftarrow\!\!\bullet} B$

The arguments are the same as before when we made the transformation of channel t_7. We have an authentic bluetooth channel from B to T, using channel t_6, and an authentic bluetooth channel from T to A, using channel t_3. Authentic information can now be sent from B to A through T. If we apply the formal rule (2) in Section 2.1 on these two channels we would get:

$$\left.\begin{array}{c} T \overset{t_6}{\longleftarrow\!\bullet} B \\ A \overset{t_3}{\longleftarrow\!\bullet} T \\ t_3 > t_6 \\ A \text{ trusts } T \end{array}\right\} \implies A \overset{t_6\ t_3}{\longleftarrow\!\bullet} B$$

Channel t_3 is available after the time when channel t_6 is available and hence the rule can be applied.

Now we have this authentic bluetooth channel from B to A through T and an insecure bluetooth channel from B to A, using channel t_8. Using the same arguments as before, when we transformed channel t_7 into an authentic channel, we could use the formal transformation rule (1) in Section 2.1 and end up with:

$$\left.\begin{array}{c} A \overset{t_3\ t_6}{\longleftarrow\!\bullet} B \\ A \overset{t_8}{\longleftarrow} B \\ t_8 \geq t_3 \end{array}\right\} \implies A \overset{t_8}{\longleftarrow\!\bullet} B$$

This security verification has shown that we can transform the insecure bluetooth channels t_7 and t_8 into authentic channels, hence our goal with this verification is achieved.

When we have the two channels t_7 and t_8 authentic we could easily make the communication confidential if we want to. Since A has B's public key and B has A's public key, ordinary public key encryption could be used. Since we only where interested in making the channels authentic no further discussion will be made about the confidential issues.

This final figure, Figure 4, shows the security properties of the available channels after the transformations made in this section:

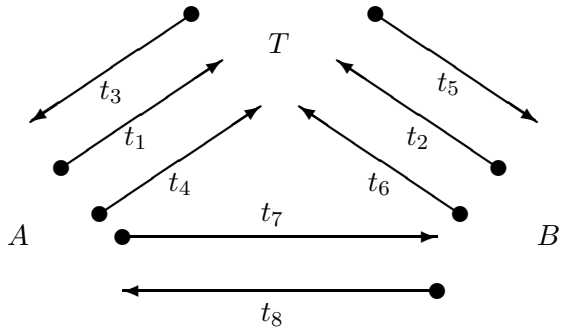

Figure 4: All channels are now authentic channels.

4.2 Attacks

In this section we will discuss some various types of attacks that could be used to compromise the security in Secure Drag&Drop Key-Exchange.

There could be different goals associated with an attack. An attacker may want to eavesdrop information sent by a device, take control over a device or just detect devices. Information that could be interesting could be audio signals, sent by for example the stereo, or it could be information sent by a sensor of some kind. If we have doors and windows that have the ability to be controlled over a wireless connection they could be a likely target. The detection of devices could reveal a technical equipped home, a smarthome, that could be a good target for burglary.

There is always the risk of bugs and vulnerabilities in a device software or hardware that an attacker can use and compromise the security. Many manufacturers include backdoors so that support staff can help customers. If these backdoors are available over a network they can be used by an attacker.

If the security mechanism is eligible then most people would not even bother to enable it. An attacker can enable the security and prevent the real user to access the device. To prevent this the enabling and disabling of the security should only be possible when you have physical contact with the device [7].

Since we are using visual codes to get authentic information, like the bluetooth address, there is a risk of someone trying to manipulate the visual code to be able to interfere with the communication. A visual code could be viewed in a display or it could be placed as a label on a device. If we have devices with displays it would be safer to view the visual code in the display. It would be easier to manipulate a visual code label. This is an

attack that requires physical contact with the device. With physical contact the attacker can replace the visual code with his own visual code. When the owner/user of the device is trying to connect to it, he or she will instead try to connect to the attackers device. Valuable information could be lost to the attacker.

4.3 Conclusions

The security evaluation of the algorithm is very important. We argued that the algorithm is secure and that had to be proven. This was done in Section 4.1 and some issues where discovered. These issues were discussed and some proposals for solutions were presented.

The algorithm provides authentication and that was what we wanted. When we have authentication, confidentiality can easily be introduced using public key systems. The attacks presented are common in many systems and there are ways to prevent them.

5 Prototype implementation

In this chapter the prototype implementation will be presented. First the implementation of the security manager, the application for the camera-equipped mobile phone, and then the implementation for the other two devices, the mobile phone and the laptop computer.

Since we have three devices involved in a key exchange, the security manager, the mobile and the computer, three different applications has been implemented. If we use the same type of platform to run the application on, for the mobile phone and the computer, one implementation would be enough instead of two. The same type of platform could be to use two mobile phones instead of one mobile phone and one laptop computer. The platform could be for example two Nokia Series 40 mobile phones like the one we used for computational evaluation in Appendix A. To make it more realistic and flexible two different types of platforms were used, one Nokia Series 40 phone and one laptop computer. These implementations will be discussed in the following sections.

Figure 5: From the left: Nokia 6230i mobile phone, Nokia N70 mobile phone and a Dell D410 laptop computer.

5.1 Camera-equipped mobile phone

The implementation of the security manager presented in Chapter 3 has been done using J2ME MIDP 2.0 [10]. The most important APIs used are the

bluetooth API and the mobile media API [9, 11]. To make this implementation robust and responsive concepts like threads and process synchronization has been used.

The camera-equipped mobile phone used to test this implementation is a Nokia Series 60 model called N70. Important to notice is that the mobile media API is not fully implemented on all Nokia camera phones. The simpler Series 40 models has not a full implementation of the API and the camera can not be accessed.

A simple visual code reader has been implemented. This simple version can only recognize the eight RGB colors, red, green, blue, magenta, cyan, yellow, white and black. This allow us to have eight unique devices where each color represent one bluetooth address. The security manager has a address list stored which maps a color to a bluetooth address.

When connecting to a device via bluetooth a bluetooth address and a channel number is required. The bluetooth address is unique and fixed. The channel number is not fixed and it is similar in concept to a TCP/IP port where each service on the device is associated with a port. When establishing a bluetooth connection between two devices in the standard way a device discovery process and a service discovery is required. The device discovery process detects nearby devices and the service discovery searches these devices for the service you wish to connect to. Since we already have the bluetooth address of the device, through the visual code reading, and know that it has the service we are looking for the device discovery process is not needed. Since we do not know the channel number a service discovery is needed and through this service discovery we retrieve the channel number. Now we have both the bluetooth address and the channel number and we can now connect to the other device and start commmunicate [4].

The following scheme describes the communication involved in the implementation. In the scheme we see which devices that communicate, what is being sent and over what channel.

Initialization:

$Camera\ mobile \longleftarrow mobile:$	$URL_{mobile}:$	visual channel
$Camera\ mobile \longleftarrow computer:$	$URL_{computer}:$	visual channel
$Camera\ mobile \longrightarrow mobile:$	Service request:	bluetooth channel
$Camera\ mobile \longleftarrow mobile:$	Channel:	bluetooth channel
$Camera\ mobile \longrightarrow computer:$	Service request:	bluetooth channel
$Camera\ mobile \longleftarrow computer:$	Channel:	bluetooth channel

Communication:

$Camera\ mobile \longrightarrow mobile:$	state:	bluetooth channel

$Camera\ mobile \longrightarrow computer:$	state:	bluetooth channel
$Camera\ mobile \longleftarrow mobile:$	$PK_{mobile}:$	bluetooth channel
$Camera\ mobile \longleftarrow computer:$	$PK_{computer}:$	bluetooth channel
$Camera\ mobile \longrightarrow mobile:$	$URL_{computer}:$	bluetooth channel
$Camera\ mobile \longrightarrow computer:$	$URL_{mobile}:$	bluetooth channel
$Camera\ mobile \longrightarrow mobile:$	$PK_{computer}:$	bluetooth channel
$Camera\ mobile \longrightarrow computer:$	$PK_{mobile}:$	bluetooth channel

First the security manager reads the visual code of the mobile to get URL_{mobile}. How this is done has been discussed earlier. The same procedure is done with the computer. When the right URLs has been retrieved from the address list a service discovery has to be performed to get the channel number of the SDDKE service. This is done on both the mobile and the computer. When the bluetooth address and the channel number is retrieved the initialization steps are done and the communication can start.

The security manager establishes one connection to the mobile and one to the computer. The first thing the security manager sends to the two devices are a state flag. This state flag will indicate which device that will start to be the prover and which one that will start to be the verifier in the Fiat-Shamir authentication protocol used later. The device that the first visual code, read by the security manager, belongs to will get a state flag that is 0 which indicate that the device will start to be prover and the device that gets a state flag that is 1 will start to be the verifier. This will be helpful later in the communication when the mobile and the computer will communicate with each other and it will be discussed in Section 5.2. The two devices responds to the security manager with the their public keys, PK_{mobile} and $PK_{computer}$. Now the security manager has all information needed and can now start distribute connection information and public keys.

The security manager sends URL_{mobile} and PK_{mobile} to the computer and $URL_{computer}$ and $PK_{computer}$ to the mobile. When this is done the two connections are being closed and the security manager is done. What is happening after the security manager is done will be discussed in the next two sections.

5.2 Mobile phone

This implementation is very simple since it only is sending and receiving data. Receiving the state flag, the URL and the public key of the other device from the security manager and sending its channel number and public key to the security manager. Like the implementation of the security manager the J2ME MIDP 2.0 has been used and the bluetooth API [9, 11].

What is happening in the communication between this device and the security manager has be discussed in the previous section. What is left to discuss is what happens when the security manager is done.

When the security manager is done, the mobile phone and the computer have all needed information to be able to connect and verify the identity of each other. To verify the identity the Fiat-Shamir authentication protocol is used which was presented in Section 2.2. Here is were the state flag is being used. The device that got the state flag that was 0 will start as prover to prove knowledge of the secret key S that belongs to the public key PK to the other device that got the state flag that was 1. That device will start as verifier.

When the protocol is done they switch so that the device that started to be the verifier now will prove knowledge of his secret key S and the other device will be the verifier. If both succeed in proving their knowledge of their secret keys they are authenticated. Notice that the this only assures the identity of an entity only at a given instant in time. Now the mobile and the computer are authenticated to each other and the SDDKE algorithm is done.

5.3 Computer

As discussed earlier the version of the application for the two devices, the mobile and the computer, are almost the same with some small differences depending on the platform running on. Since this implementation is for a computer the J2ME MIDP 2.0 could not been used. This implementation was done in Java (J2SE) with an additional open source implementation of the JSR-82 bluetooth API [12]. For more details on the implementation see Section 5.2.

6 Summary & Conclusions

The following chapter will summarize and conclude this thesis and a future work section will present some ideas of possible extensions of this work.

6.1 Summary

This thesis has presented the concept of Secure Drag&Drop Key-Exchange, the general idea and the practical use. An implementation has shown a "proof-of-concept" and the security verification, that has been made, shows that the algorithm provides authentication using a public key system. Possible attacks on the system have also been presented and discussed.

The thesis started with an introduction to the subject and a problem description was presented. Background information that was useful to understand the security terminology and concepts was also presented. Last this summary and the following conclusions section concludes the thesis.

6.2 Conclusions

The goal of this thesis was to design and investigate the approach to use a camera-equipped mobile phone as some kind of security manager to distribute public keys between devices that want secure communication. The distribution of the key material had to be intuitive and easy to understand and it would involve as little configuration and interaction as possible with the user.

The solution to this we call Secure Drag&Drop Key-Exchange. To distribute public keys for secure communication the distribution itself has to be secure, the keys has to be authentic. The receiver of a public key must be sure that the key received belongs to the one it says to be. The security verification proved that the system is providing authentication if the two devices that want to communicate has some kind of relationship with the security manager. Maybe the most interesting approach to solve this problem is to let all devices to associated with a master device. The association could be done by letting the first one to present a secret key to the device to be its master device. The secret key could be presented as a visual code on the device so that the security manager could read it with the camera. The device will obey and trust the security manager until the device is reset. That could be done by just pressing a reset button on the device. This approach was presented by Stajano and Anderson in their work *The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks*.

The "proof-of-concept" prototype of Secure Drag&Drop Key-Exchange was implemented successfully. The implementation provide a simple visual

code reader that can easily be replaced with a more powerful one if needed. This simple visual code reader was powerful enough to prove the concept. The prototype for the security manager was tested on a Nokia N70 mobile phone and the two devices that wanted secure communication was a Nokia 6230i mobile phone and a Dell Latitude D410 laptop.

Since we have more and more devices at home and they will be equipped with wireless radios, like bluetooth, the security risks will increase. The communication between devices will increase and small ad-hoc networks will pop up. The configuration of these networks will be hard for a regular user and the security configuration even harder. This will create a need for an easy solution that will be intuitive and easy to understand and that does not involve buying new special devices for the purpose. Digital cameras in mobile phones are becoming standard and they are well suited for this purpose and almost everyone owns a mobile phone. Secure Drag&Drop Key-Exchange could be the solution.

6.3 Future work

A natural extension of this work would be to implement a "real" visual code reader instead of this simple version. The goal of this thesis was to prove the concept of Secure Drag&Drop Key-Exchange and for that purpose the simple version was enough. If more time had been available this could have been done.

Implementations for other platforms could also have been done. This would allow us to see if it would be possible to use this algorithm in its current state on more "primitive" devices than a simple mobile phone, like a sensor of some kind.

This prototype is only implemented for bluetooth so a natural extension would be to make is possible to use other technologies as well.

6.4 Acknowledgments

I would like to thank my supervisor, Christian Rohner for his valuable insights and comments during my master thesis project. Mats Björkman for being the examiner and my girlfriend Therese Jansson for love and support.

A APPENDIX: Test results

Here test results from the evaluation of the Fiat-Shamir authentication protocol will be presented. We claimed, in Section 2.2, that small wireless devices have limited computational power. To verify or reject our claim several tests were made.

A.1 Time measurement

The following tests were performed on a Nokia 6230i mobile phone. When running the Fiat-Shamir protocol several calculations are made. In each round of the protocol $r^2 \bmod n$, random e and $r \cdot s^e \bmod n$ is calculated. If we look at the last equation we see that, since e is either 0 or 1, we could get either $r \bmod n$ or $r \cdot s \bmod n$. Since both s and r must be $\geq$ to 1 and $\leq$ to $n-1$ we could argue that $r \cdot s \bmod n \approx r^2 \bmod n$. From this approximation we could argue that it would be enough to test how long time it takes to perform the $r^2 \bmod n$ equation and then use that as an approximation for the time to calculate $r \cdot s \bmod n$. Hence the interesting calculations to do time measurements on are the $r^2 \bmod n$ equation, the random generation of e and the random generation of r. For the test we used $n = 2127342101$ which is the product of the two random prime numbers $p = 6781$ and $q = 313721$. To generate the random numbers, e and r, the standard java.util.Random class was used. Time measurements were made using the system call System.getMillis().

The random generation of e was made 10000 times to get enough values to make any conclusions about the time (see Table 2). Of these 10000 generations 4952 generated $e = 0$ and 5048 generated $e = 1$. The generations took from $0-2$ milliseconds and the distribution was 8725 generations that took 0 milliseconds, 1273 that took 1 millisecond and 2 that took 2 milliseconds to generate. The average generation time was 0.1277 milliseconds and that is not much.

With the random generation of r we got similar results (see Table 4). Here we also performed 10000 generations and it took from $0-2$ milliseconds and the distribution was 8501 generations that took 0 milliseconds, 1497 that took 1 millisecond and 2 that took 2 milliseconds to generate. The average generation time was therefore 0.1501 and that is not much either.

The time measurements of the $r^2 \bmod n$ equation was made in two different tests. To see if it took more time with large values than small values on r we made one test with r going from $1-10000$ and one with $(n-10000)-n$ which was $2127332101 - 2127342101$. In each round the equation was calculated 100 times. This means that there was $100 \cdot 10000$ calculations made

in each test. We had to make 100 calculations of each equation to get any measurable times.

With the first test (see Table 1) the 10000 calculation rounds of $r^2 \bmod n$ took from $0 - 9$ milliseconds and the distribution was 6413 that took 0 milliseconds, 3561 that took 1 millisecond, 25 that took 2 milliseconds and 1 that took 9 milliseconds. That gave us a average of 0.362 milliseconds.

With the second test it took a little longer (see Table 3). The calculation rounds of $r^2 \bmod n$ took from $0 - 11$ milliseconds and the distribution was 4642 that took 0 milliseconds, 5198 that took 1 millisecond, 155 that took 2 milliseconds, 3 that took 3 milliseconds, 1 that took 5 milliseconds and 1 that took 11 milliseconds. The average time spent was 0.553 milliseconds. Remember that each round was performed 100 times so the average time spent on one calculation would be 0.00553 milliseconds in the second test and 0.00362 in the first test.

Table 1: $r^2 \bmod n$ (r from 1 - 10000)

Seconds	Nr of rounds
0	6413
1	3561
2	25
3	0
4	0
5	0
6	0
7	0
8	0
9	1

Table 2: Generation of e

Seconds	Nr of generations
0	8725
1	1273
2	2

Table 3: $r^2 \bmod n$ (r from 2127332101 - 2127342101)

Seconds	Nr of rounds
0	4642
1	5198
2	155
3	3
4	0
5	1
6	0
7	0
8	0
9	0
10	0
11	1

Table 4: Generation of r

Seconds	Nr of generations
0	8501
1	1497
2	2

A.2 Analysis

As we can see it take almost no time at all to perform a calculation of this type. The random number generations takes less than 0.2 milliseconds and the $r^2 \bmod n$ takes less than 0.01 milliseconds. If we assume that we run the Fiat-Shamir protocol for 40 rounds the calculations involved would take less than 17 milliseconds to perform, $(40 \cdot 0.2 + 40 \cdot 0.2 + 40 \cdot 0.01 + 40 \cdot 0.01)$, with $n \leq 2127342101$. 17 milliseconds is nothing compared to the communication time involved in the protocol. By these time measurements and this little discussion we reject our first claim, made in Section 2.2, that small wireless devices have limited computational power. It is not a limited factor in this context.

References

[1] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1996, Chapter 10. ISBN 0-8493-8523-7, http://www.cacr.math.uwaterloo.ca/hac/.

[2] Ueli M. Maurer, Pierre E. Schmid. *A Calculus for Security Bootstrapping in Distributed Systems*, 1996.

[3] Jonathan M. McCune, Adrian Perrig, Michael K. Reiter. *Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication*, November 2004, CMU-CS-04-174.

[4] David Scott, Richard Sharp, Anil Madhavapeddy, Eben Upton. *Using Visual Tags to Bypass Bluetooth Device Discovery*, January 2005.

[5] David Scott, Richard Sharp, Anil Madhavapeddy, Eben Upton. *Using Camera-Phones to Enhance Human-Computer Interaction*, 2004.

[6] Michael Rohs, Beat Gfeller. *Using Camera-equipped Mobile Phones for Interacting with Real-World Objects*, April 2004.

[7] Stephen R. Hanna. *Configuring Security Parameters in Small Devices*. Internet Draft, Sun Microsystems Inc, July 2002, draft-hanna-zeroconf-seccfg-00.txt.

[8] Frank Stajano and Ross Anderson. *The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks*, In Security Protocols, 7th International Workshop, 1999.

[9] Java Community Process, Community Development of Java Technology Specifications. *JSR-82: Java APIs for Bluetooth*, http://www.jcp.org (search for JSR-82).

[10] Java Community Process, Community Development of Java Technology Specifications. *JSR-118: Mobile Information Device Profile 2.0*, http://www.jcp.org (search for JSR-118).

[11] Java Community Process, Community Development of Java Technology Specifications. *JSR-135: Mobile Media API*, http://www.jcp.org (search for JSR-135).

[12] SourceForge.net. *BlueCove*, http://sourceforge.net/projects/bluecove.

[13] Simon Woodside, Semacode. *All technical details you could possibly want, and more*, http://semacode.org, Home>About>Technical Chapter 5. Technical challenges.

[14] Colorzip Media Inc. *ColorCode Technology*, http://www.colorzip.com, Home>Technology.

[15] Jun Rekimoto and Yuji Ayatsuka. *CyberCode: Designing Augmented Reality Environments with Visual Tags*, Proceedings of DARE, Designing Augmented Reality Environments, 2000.