U N I V E R S I T Y O F S P L I T
FACULTY OF ELECTRICAL ENGINEERING, MECHANICAL ENGINEERING
AND NAVAL ARCHITECTURE

S V E U Č I L I Š T E U S P L I T U
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Toni Perković

User Friendly Authentication Mechanisms in User-to-Device and Device-to-Device Interactions

Autentifikacijski mehanizmi usmjereni korisniku u interakcijama korisnika s uređajem i uređaja s uređajem

DOCTORAL THESIS
DOKTORSKA DISERTACIJA

Split, 2013.

The research reported in this thesis was carried out at Department of Electronics, University of Split, Chair of Telecommunication Technology and Signal Processing.

Supervisor: Dr. sc. Mario Čagalj, Associate Professor, FESB, University of Split, Croatia
Disertation number: xxx-xxx

Doktorska disertacija je izrađena na Zavodu za elektroniku Fakulteta elektrotehnike, strojarstva i brodogradnje Sveučilišta u Splitu

Mentor: Dr. sc. Mario Čagalj, izv. prof., FESB, Sveučilište u Splitu, Hrvatska
Rad broj: xxx-xxx

There are 69 figures, 12 tables, 45 equations and 165 references in this doctoral thesis.

Committee for assessment of doctoral dissertation:

1. Dr. sc. Dinko Begušić, Full Professor, FESB, University of Split, Croatia
2. Dr. sc. Mario Čagalj, Associate Professor, FESB, University of Split, Croatia
3. Dr. sc. Nikola Rožić, Professor Emeritus, FESB, University of Split, Croatia
4. Dr. sc. Darko Huljenić, Associate Professor, Ericsson Nikola Tesla d.d., Zagreb
5. Dr. sc. Srđan Čapkun, Associate Professor, ETH Zürich, Switzerland

Povjerenstvo za ocjenu doktorske disertacije:

1. Dr. sc. Dinko Begušić, red. prof., FESB, Sveučilište Splitu, Hrvatska
2. Dr. sc. Mario Čagalj, izv. prof., FESB, Sveučilište Splitu, Hrvatska
3. Dr. sc. Nikola Rožić, prof. emeritus, FESB, Sveučilište Splitu, Hrvatska
4. Dr. sc. Darko Huljenić, izv. prof., Ericsson Nikola Tesla d.d., Zagreb
5. Dr. sc. Srđan Čapkun, izv. prof., ETH Zürich, Švicarska

User Friendly Authentication Mechanisms in User-to-Device and Device-to-Device Interactions

A large number of authentication mechanisms that grant access to an information system or a service are usually too complex or incomprehensible for an end user (e.g., an invalid certificate warning). Interacting with such a large number of authentication methods (e.g., e-mails) can result in an increased user frustration and a higher probability of critical errors, making the proposed mechanisms both unusable and insecure.

In the context of user-to-device authentication (e.g., user authentication on ATM, PC), we consider two types of malicious behavior. The first type deals with the problem of observation attacks in secure user-authentication protocols that exploit nonuniform human behavior. Unfortunately, such type of behavior gives the attacker opportunity to successfully mount a timing attack. This thesis presents the first public report about a successful (timing) attack on two secure authentication methods, Undercover and a US patent.

The second type of malicious behavior concerns the problem of *relay* attack in the context of financial transactions. Inspired by the recent work of Stajano et al. [1] on *unrelayable* channels, in this thesis we show that a piece of paper imprinted with a secret message and appropriately folded (hence *fortune cookie*) can implement a *weakly unrelayable* channel.

Concerning the user-assisted device-to-device authentication problem (e.g., setting up a secure WiFi connection, secure initialization of a relatively large number of wireless sensor networks (eHealth, Smarthome)) we designed new protocols for the initialization of multiple resource constrained wireless devices and proved their security in a formal model. We used a paradigm based on multichannel protocols in which information is transmitted over both a radio and a visible light channel (VLC). By using unidirectional error-detection codes we proved the security of our protocol in a much stronger attacker model where the VLC is semi-authentic.

Keywords: Usable Security, Timing attack, Relay Attack, Multichannel Protocol, Message Authentication Protocol

Autentifikacijski mehanizmi usmjereni korisniku u interakcijama korisnika s uređajem i uređaja s uređajem

Veliki broj autentifikacijskih mehanizama za kontrolu pristupa informacijskim sustavima i uslugama je često kompliciran i nerazumljiv za krajnjeg korisnika (npr. upozorenje o nevaljalim certifikatima u preglednicima). Posljedice ovakog stanja jesu frustracije korisnika i povećana vjerojatnost kritičnih grešaka koje u konačnici mogu ugroziti sigurnost takvih sustava i usluga.

U kontekstu interakcija korisnika sa uređajem (npr. autentifikacija na bankomatu), u ovoj tezi se razmatraju dva oblika malicioznog ponašanja. Prvi slučaj se bavi problemom "napada promatranjem" na sigurne autentifikacijske metode koji se temelji na neuniformnom ponašanju korisnika, a ono otvara napadačima mogućnosti prema tzv. *timing* napadima. Ova disertacija predstavlja ujedno i prvi primjer uspješnog (*timing*) napada na dvije metode autentifikacije.

Drugi slučaj malicioznog ponašanja razmatra problem napada prosljeđivanja komunikacije i to u kontekstu financijskih transakcija (eng. *relay* napad). U ovoj tezi je pokazano kako prekopljeni papir sa tajnom informacijom u kombinaciji sa sveprisutnim pametnim telefonima može implementirati svojstva kanala koji onemogućavaju napadaču prosljeđivanje informacija.

U kontekstu autentifikacije uređaja sa uređajem (uspostava sigurne WiFi veze između više mobilnih jedinica, inicijalizacija velikih bežičnih osjetilnih mreža - eHealth, pametne kuće) kreirani su novi sigurnosni višekanalni protokoli za međusobnu inicijalizaciju višestruko ograničenih bežičih uređaja čija je sigurnost dokazana u formalnom modelu. Korištena je paradigma višekanalnih protokola kod kojih se informacija osim putem radio kanala prenosi i optičkim kanalom. Kombinacijom unidirekcionalnih kodova je dokazana sigurnost predloženog protokola u okviru jačeg napadačkog modela koji dozvoljava napadaču manipulaciju poruka koje se prenose optičkim kanalom.

Ključne riječi: *Timing* napad, *relay* napad, višekanalan protokol, protokol za dokaz autentičnosti poruke

Contents

List of Tables

List of Figures

List of Acronyms

ATM Automated Teller Machine
CASR Cellular Automata Shift Register
D2D Device-to-Device
ENT Pseudorandom Number Sequence Test
LFSR Linear Feedback Shift Register
IBC Intra-Body Communication
IEEE Institute of Electrical and Electronics Engineers
IoT Internet of Things
IR Infrared
IRB Institutional Review Board
ISO/IEC International Organization for Standardization/International Electrotechnical Commission
KDE Kernel Density Estimation
LED Light Emitting Diode
M2M Machine to Machine
NFC Near Field Communication
NIST National Institute of Standards and Technology
OOB Out-Of-Band
PC Personal Computer
RFC Request for Comments
RNG Random Number Generator
RF Radio Frequency
SAS Short Authentication String
SMS Short Message Service
SUS System Usability Scale
U2D User-to-Device
USB Universal Serial Bus
VGA Video Graphics Array
VLC Visible Light Communication
WSN Wireless Sensor Network
WiFi Wireless-Fidelity

1 INTRODUCTION

People in today's society are accessing an increasing number of information systems and services. Many of those systems use various authentication mechanisms that an operator needs to learn and use, e.g., accessing a social network, configuring a secure WiFi network, using email and accessing a bank account online. Increase in the number of such authentication-based interactions burdens the operator with extra demands. Despite the large number of secure authentication mechanisms, the proposed methods are sometimes too complex to use and incomprehensible for the user (e.g. the warning message about an invalid certificate from a browser).

Recent security experience shows an inherent tradeoff between security and usability; interacting with a large number of authentication methods implies more time to authenticate and a higher probability of errors, making the proposed mechanisms unusable. Interacting with such a large number of authentication methods (e.g., e-mails and e-banking) can result in increased user frustration and a higher probability of critical errors (e.g., ignoring instructions), making the proposed mechanisms both unusable and insecure. It is important to develop a set of authentication mechanisms that are both secure and usable for end user. The focus in this thesis will be placed on the development of *user-to-device authentication* mechanisms (H2D) and *user-assisted device-to-device* mechanisms (D2D).

Human-to-Device (H2D) Mechanisms

Users in large corporations typically have a substantial number of unique passwords, each used to protect a system or an application they access. Similarly, with rapid growth in the number of Internet services (e.g., social networks, e-banking, forums, email, blogs and cloud services), users are expected to remember a unique set of credentials for each system. Since users can only remember a limited number of unique passwords, they often reuse the same password for several services. A variety of solutions have been proposed that allow users to remember passwords within a larger period of time, such as graphical passwords, but such solutions still do not protect users from password theft.

One of the simplest ways to steal a password is *shoulder-surfing* which can be automated by installing hidden cameras, fake keypads or even fake terminals (like fake ATMs). Other identity theft schemes include social engineering attacks like phishing, malware-based attacks (keylogging and Trojan horses). A number of solutions have been proposed within

the last couple of years to fight these observation attacks, most of them requiring that users perform cognitive operations (e.g., addition, recognition, visual recall).

Besides observation attacks, users cannot validate the authenticity of the device where they are entering personal credentials. Knowing this, the attacker installs his own fake device to access some legal service (e.g., to withdraw money). The attacker is in the middle of the communication between two legal entities and simply relays user actions from his fake device to another legal device, but keeps the service to himself. Such form of attack is known as a *relay* attack.

In the context of ATMs, the attacker executes the following simple relay attack: the user inserts her smartcard into the fake ATM; the attacker relays the complete transaction over a fast radio channel; the attacker picks up the money from another terminal. Relay attacks are easy to implement since attackers do not need to know anything about the protocols or the underlying cryptographic keys. The attacker simply relays the messages between two (or more) legal entities (enters the PIN or exchanges cryptographic keys).

For money transactions, existing systems like mobile solution M-PESA are insecure against relay attacks but in use by 40 percent Kenya's adults. Using M-PESA mobile solution, users are allowed to withdraw the money from the ATM using an SMS message. Existing solutions to counter relay attacks are based on distance-bounding protocols that estimate the distance between two communicating parties. Other recently proposed solutions are based on multichannel protocols where, in addition to the standard channel, propose an additional "special" communication channel. The exact property of the proposed "special channel" makes such protocols safe against relay attacks. While existing distance-bounding protocols require precise time measurement on the nanoscale level, they still have no practical use because they (i) require specialized hardware changes of existing solutions and (ii) operate on lower layers such as radio channels. Conversely, the proposed multichannel-based solutions against relay attack are highly impractical.

Device-to-Device (D2D) Mechanisms

Users are facing problems on a daily basis as they try to establish initial cryptographic credentials between two or more wireless devices such as WiFi networks and Bluetooth. Working with WiFi networks has shown us that users lack sufficient knowledge to install secure communication between two or more wireless computers. New emerging technological trends like Internet of Things, Machine-to-Machine systems (M2M) and sensor networks, predict that users will be surrounded by a large number of devices, most communicating wirelessly. The end user's burden will be to install secret credentials in every single device.

For example, in smart power grid systems, users will be equipped with a number of wireless sensors used to collect and exchange information with particular devices. That form of communication will require the devices to be secured with preinstalled cryptographic

credentials, a procedure that users will generally have to perform by themselves. Many existing systems consider the key predeployment to be a trivial matter (e.g. TinySec), some types of sensor nodes might have a USB (or similar) connector, or assume the keys are exchanged in clear over a radio channel.

Proposed solutions, such as pre-loading keys during the manufacturing phase, are insecure, do not scale well, rely on the presence of specialized equipment such as a Faraday cage or are simply too complicated for the user. Therefore, it is necessary to find a mechanism that is easy to administrate by a non-specialist or end user. The mechanism should be able to initialize a reasonably large number of multiple constrained wireless devices. Such devices do not usually have wired or other traditional interfaces, such as displays and keypads, but are equipped with few on-board LEDs and buttons (e.g. wireless sensor devices).

Hypothesis

Existing and most widely adopted solutions for user-authentication are the ones based on a static PIN/password. However, these solutions are insecure against observation attacks. One of the simplest ways to steal a user's PIN is shoulder-surfing which can be automated with hidden cameras or fake keypads (like fake ATMs). Many solutions have been proposed to fight observation attacks and most of them are challenge-response authentication protocols that require users to perform cognitive operations (e.g. addition, recognition, visual recall).

Unfortunately, such cognitive operations increase mental focus during the authentication process which decreases usability and gives the attacker opportunity to successfully mount a timing attack. With the timing attack, the adversary tries to recover the PIN/password simply by analyzing the time a user invests to execute a particular cryptographic function (a mental process as a function of that PIN/password digit).

In this thesis, we hypothesize that cognitive burden during the protocol execution (authentication method) affects the non-uniform user behavior - an inconsistent reaction time that the adversary exploits to reduce entry of the PIN/password. From the detailed analysis of the existing protocols, a paradigm will be created to further enhance the design and performance evaluation of existing and future security authentication methods.

Most existing solutions to fight relay attacks are based on distance bounding protocols. Other proposed solutions fight relay attacks with the aid of multichannel protocols. Despite the existence of practical relay attacks, in which the attacker as a consequence steals the card or picks up the money from another ATM terminal, the existing solutions are economically impractical as they require modification of the existing hardware. On the other hand, existing multichannel protocols are highly impractical and serve as an illustration of their multichannel-based relay protection.

Within this thesis, we consider creating a practical and a user-friendly implementation of the protocol which is safe against relay attack including financial transactions and requires

minimal or no hardware changes to the existing equipment. Also, the proposed solution will use the concept of multichannel protocols, will operate at the application layer and therefore will be oblivious to the underlying communication technologies (being wireless such as WiFi, NFC, Bluetooth, or Intrabody Communication - IBC).

Experience learned from working with WiFi networks showed that users do not possess enough knowledge to create a secure communication between two or more wireless computers. New emerging technological trends like Internet of Things, Machine-to-Machine systems (M2M) and sensor networks, predict that users will be surrounded by a large number of devices, most of which will communicate wirelessly. This places a large burden on the end user because secure communication will require that users install secret credentials in every device (device-to-device authentication).

Proposed solutions are insecure (e.g., factory initialization), do not scale well (e.g., initialization over a USB port), require the existence of a specialized equipment (e.g., a Faraday cage) or are too complicated for the user. Within this thesis, we consider a hypothesis that it is possible to create both a secure and user-friendly solution for the initialization of a large number of wireless devices based on multichannel protocols.

In the context of device-to-device interaction, the focus will be placed on wireless sensor devices that have constrained interfaces (e.g. one LED and a pushbutton). Although the proposed protocol generally applies to sensor devices, it works with wireless devices that have minimal hardware requirements (one LED, pushbutton and radio transceiver) such as WiFi routers, smartphone devices, etc.

Research Methodology

Within the context of authentication methods, the main focus of this thesis was placed on the security analysis of some existing and important user-authentication methods/papers that play an important role in security, usability and human-computer interaction (according to their citations numbers). Specifically, the main focus was placed on more usable methods that achieve faster authentication time.

These methods fall in the group of partially observable authentication schemes where the adversary has partial access to the complete PIN/password entry process. We investigated the design flaws of methods that are generally secure within a particular attacker model, but whose authors did not consider a scenario where the attacker tries to find information about the secret PIN/password by observing some public information. As public information, the attacker can observe a number of unique and user-independent patterns for a PIN/password digit, like error rates, response time as a function of a unique (and secret) challenge, etc. Such type of attacks where an adversary records the time required to execute some cryptographic protocol as a function of a particular key (PIN/password) is called a timing attack.

As seen, the timing attack presents nothing more than a classification problem. The

attacker observes some public information and wants to classify it to the PIN/password digit. In order to create such a classification mechanism, we first created a training set that holds various patterns (e.g., login times, error rates) extracted from the login sessions of multiple users (other than the tested one). The pattern recognition system used for our PIN/password classification was used as a result of a classifier combination technique that combines results from multiple classifiers.

As a classifier combination technique, we used Bayes Sum rule. As a first step to building a classification mechanism, we created an application and a web page (depending on a tested login method) that collects information about the authentication method (login times, error rates) into a central database for further processing. Next, we conducted a study with a large number of users. Depending on the time and available resources, the type of the study was between-group and within-group. In a between-group study each participant was randomly assigned to one experiment, while in a within-group study all users randomly conducted all the tested login methods.

The results of a between-group study are cleaner, since users avoid the learning effect of multiple methods and usually experience less fatigue. However, this study usually requires a large number of participants (one participant is tested for one method). In contrast, a within-group study requires less testers because the data from the same participant is collected from multiple tested login methods. However, such a test is vulnerable to learning effect and fatigue.

To collect clean and unbiased results, all participants were volunteers asked to help our research; none of them were economically compensated/motivated. For the purpose of our tests, a total number of 96 users took part in the study. Besides testing the security of each method against the proposed timing attacks, we conducted an extensive usability evaluation on each method. In that direction, each user was given a pretest questionnaire that collects user demographics (age, sex, computer experience) and a posttest questionnaire that collects user satisfaction with the tested method. As a posttest questionnaire, we used the SUS test, a widely used and highly reliable 10-item Likert scale test that polls subjects' satisfaction with computer systems. Testers were also rated to judge the security and acceptability of the tested methods using Likert scale questions.

The login results from tested methods were compared with standard tests like student t-test and F-test. Depending on the types of study, we used different types of t-test. For example, in between-group tests, we used paired samples t-test, while in within-group study an independent samples t-test was used. To test correlation between two variables (e.g., login times and error rates), we used Pearson's product moment coefficient called Pearson's r, whose value ranges from -1.00 to 1.00. For example, when the Pearson's r value is -1.00, it suggests a perfect negative linear relationship between the two variables. On the other hand, when the Pearson's r value between two variables is 1.00, it suggests that any increase in the scores of one variable will perfectly predict a specific amount of increase in the scores of

the other variable. At the end, when the Pearson's r value is 0, there is no linear relationship between the two variables.

A first step to finding a solution against a mafia fraud attack was to analyze existing solutions in the context of financial transactions. Focus was placed on finding a solution based on multichannel protocol (that use fast radio channel and a slow channel). Some existing multichannel protocols cannot prevent relay attacks as information is easy to modulate from a slow channel to a fast (radio) channel. On the other hand, existing multichannel-based relay protection solutions are highly impractical.

The goal in this thesis was to create such a "special" channel that the attacker will not be able to relay. One solution would be to use a simple folded paper that holds a secret nonce. To test the performance and the security of the proposed protocol on a large population of users, we created an ATM cash machine terminal. For the purpose of our tests, a total number of 54 users took part in the study. The login results from tested methods were compared with standard tests like Student t-test or F-test. At the end we proved the security of the protocol in a formal model.

A first step to finding a secure method for the mutual authentication of wireless devices was to analyze existing solutions for the initialization of a large number of devices (assisted from an unaided user). As an interesting direction to finding a solution is a generalization of a twoparty SAS protocol (Short Authentication String) to a multiparty scenario. SAS protocol uses authentic out-of-band (OOB) channel to convey a small number of bits (e.g., 15) that authenticates much longer messages exchanged over a fast radio channel.

As a solution, we used optical channel (blinking LEDs on a wireless devices) as an out-of-band channel. Existing solutions that use LEDs as an out-of-band channel are insecure and prone to bit-manipulation attacks that render an out-of-band channel semi-authenticated. As a possible solution to counter this problem, we proposed a new coding scheme of the optical channel based on the combination of Manchester and Berger unidirectional error-detection code that is secure and easy to interpret by non-specialists and unaided end users. We proved the security of the proposed method in a formal model. We also demonstrated the feasibility of the proposed method via a preliminary usability study with 28 users. To estimate the usability of the tested method we used the SUS test (System Usability Scale).

Dissertation Outline

In Chapter 2, we report successful side-channel timing attacks on two authentication schemes: the original design of Undercover in [3] and US patent [7] modulo 10. The timing attack on Undercover is based on the design flaws of the GUI interface implementation, and exploits users' nonuniform behavior on how they respond to different hidden challenges. On the other hand, the key component of the timing attack in modulo 10 method is a mechanism trained on the database of response times per ith PIN digit from a relatively large number

of users. Our decision to perform the timing attacks on Undercover and US patent was motivated by the fact that they fall in two different categories of authentication methods that use a secret channel: Undercover is an interface-based authentication method where the user visually looks for patterns on the interface to compute the response, whereas modulo 10 is a non-interface based authentication method where the user mentally calculates the response (performs modulo 10 additions). In this way we wanted to show that non-uniformities can be found in many aspects of human behavior and that side-channel attacks are not particular to a specific cognitive authentication method.

We also discuss what is generally required to make the proposed mechanisms safe against the side-channel timing attacks. Our investigation of non-uniformity highlights some unique principles that we need to follow in the design of future secure human-computer authentication mechanisms. By following these principles, we will render the existing protocols safe against the timing attacks. Moreover, we will show that in some cases the only way to enhance the security of some cognitive authentication methods is to perturb the usability (e.g. by increasing the entropy of the secret **S** and/or by increasing the authentication time), which only shows the existence of a strong tradeoff between security and usability in cognitive authentication methods.

In addition, we will also demonstrate a successful intersection attack on the original design of Undercover based on the flaws in the Undercover design [3]. As we will show, this attack can be generalized to the alternative designs in [4].

Inspired by the recent work of Stajano et al. [1] on *unrelayable* channels, in Chapter 3 we show that a piece of paper imprinted with a secret message and appropriately folded (hence *fortune cookie*) can implement a protection against relay attacks (a *weakly unrelayable channel*). As we show, weakly unrelayable fortune cookies in combination with omnipresent smartphones provide strong protection against mafia fraud attacks.

Our mechanism is based on a novel (multichannel) protocol, called Forces (**For**tune **C**ooki**es**) that shares some similarities with multichannel protocols proposed by Stajano et al. [1] but still differs in one important way. Essentially, in our case we reduce the time frame within which the attacker can relay a message transmitted over the special (paper) channel. It is exactly this property that makes our solution more practical compared to highly impractical solutions outlined in [1].

The proposed solution is suitable for scenarios that involve transaction (paper) receipts such as ATM money withdrawals and payment terminals. Compared to existing solutions for contactless systems, mainly based on RF distance bounding, our solution is oblivious of the used contactless/wireless technology (e.g., NFC, WiFi, Bluetooth, infrared) or contact-based technology (e.g., intra-body communication). Moreover, we show that it requires minimal or no hardware changes to the existing equipment (in particular on the user's side).

We implemented two instantiations of our solution and performed an extensive user performance study with 54 participants. The study indicates that the proposed solution is both

easy to understand and perform by end users. Moreover, our study shows that it is possible to eliminate critical errors that rely on the alertness of the honest user. Finally, we prove formally the security of the proposed Forces protocol.

In Chapter 4, we propose two novel *multichannel* protocols for initialization of large scale wireless sensor networks. Similar to [8], our protocols involve communication over a radio channel and the out-of-band visible light channel (VLC). The first protocol uses only secret key cryptography and is suitable for CPU-constrained sensor nodes. In this protocol, each sensor node establishes a unique secret key with a *base station (BS)*. The base station comprises of a simple web camera and one sensor node attached to an ordinary PC. In the first phase of the protocol, the sensor nodes transmit secret keys to the base station over a *protected* visible light channel. In the second phase, each sensor node runs a key verification protocol with the base station over a bidirectional radio channel. Once the keys are verified, the base station can serve as a *trusted third party* and mediate establishment of security associations between any pair or any group of sensor nodes.

Our second protocol uses public key cryptography. As with the previous protocol, the ultimate goal is to establish security association between each sensor node and the base station. This protocol is based on the multichannel pairing protocol from [6, 5]. Thus, each sensor node first exchanges its public key (through specially formed commitment/openning pairs) with the base station over the radio channel. In turn, each sensor node transmits a *short authentication string* (SAS) using the visible light channel. The proposed "public key"-based protocol is similar to [8], with the difference that our protocol is designed to be secure in a very strong attacker model, where an attacker can eavesdrop, jam and modify transmitted messages by adding his own message to both a radio and a visible light channel; the attacker however cannot disable the visible light communication channel.

In Chapter 5, we present the first secure and usable initialization mechanism that works with multiple (sensor) devices having constrained resources (a LED, a button and limited power supply) and does not require any auxiliary devices, thus satisfying all the requirements outlined previously.

Our initialization mechanism is based on a novel (multichannel) protocol, called the *Group message Authentication Protocol* - GAP. GAP involves communication over a radio channel and an out-of-band visible light channel (VLC). GAP is inspired by the two-party SAS protocol [6, 5]; we show that straightforward generalizations of SAS to a multiparty protocol may easily fall short of being secure. A notable feature of GAP is that the information to be authenticated is independent of the *short authentication string* (an *indirect* binding protocol [9]) to be verified by the user over a visible light channel (i.e. the GAS and authenticated information are completely independent in the sense of probability). This, as we show, results in a lower communication cost compared to existing *direct* binding protocols. The advantage in the communication cost of our GAP protocol is especially important for battery-powered devices, such as wireless sensor nodes. We also show how to secure GAP against

malicious insider attacks (compromised sensor nodes); in [10], the devices are assumed to be benign during the initialization phase.

As we show later, the visible light channel (VLC) is prone to certain bit-manipulation attacks, that (contrary to the common belief) renders VLC *semi-authenticated*; however many existing protocols [11, 12, 8, 10] that use VLC consider it to be authenticated. In order to prevent these attacks, we use a simple combination of well known unidirectional codes (Berger and Manchester), that are easy to interpret by an end user.

Finally, we demonstrate the feasibility of the proposed mechanism via a preliminary usability study with 28 users. The study indicates that the method has reasonably low execution time, minimal error rate and is user-friendly. We further discuss how the usability and scalability of our mechanism can be improved by utilizing a zero-configuration auxiliary device (e.g., a standard camera phone with no additional computational logic), when available.

We note that although we target our scheme as a means of secure initialization of a WSN, our proposal is also equally applicable to other wireless devices scenarios. This includes, for example, the initialization of a number of commodity wireless access points that need to be installed as part of an enterprise's wireless network.

2 COGNITIVE AUTHENTICATION METHODS AND SIDE-CHANNEL ATTACKS

2.1 Introduction

Any reasonably-sensitive computer system typically starts from a user authentication process where a human user has to prove her identity. Contemporary systems use one, or a combination of the following authentication methods: "what you know" (e.g., passwords), "what you have" (e.g., hardware tokens) or "who you are" (e.g., biometrics like our fingerprints) [13]. User authentication process plays a key role in the security of the whole system since it is the first (and often the only) way to prevent unauthorized access by illegitimate users.

Despite the existence of advanced user authentication methods, the simplest one, based on static passwords/PINs, has been the most widely adopted method since its birth in the 1960s. This is because other advanced methods either require additional costs or decrease the usability. A salient drawback of static passwords/PINs is that they are extremely sensitive to replay attacks: they can be stolen and then simply replayed by attackers to impersonate legitimate users. In other words, when a user's identity is protected by a static password/PIN, stealing this password/PIN results in stealing the user's identity.

There are many different ways to steal a user's static password/PIN. One of the simplest ways is shoulder surfing [14], which can be automated by installing hidden cameras or fake keypads or even fake terminals (such as fake ATMs) [15]. Other ways to identity theft include social engineering attacks like phishing [16] and malware-based attacks like keylogging and Trojan horses [17, 18]. These attacks are often described as "observation attacks" in literature, to highlight the fact that the attacker can monitor the communication between the user and the verifier computer.

Since the early 1990s many solutions have been proposed to fight observation attacks. With the exception of a few specialized hardware based solutions, most solutions are challenge-response user authentication protocols based on shared secrets. In each authentication session, the user is asked to give responses to a number of random challenges based on her knowledge of the shared secret. Let the shared secret, the challenges, and the responses be denoted by **S**, **C** and **R**, respectively. The user will be accepted only when $\mathbf{R}=f(\mathbf{C},\mathbf{S})$. In the observation attack, we assume that the attacker can observe both **C** and **R** but does not have access to **S**. The main task of the attacker is to solve **S** from **C** and **R**. Accordingly,

the task of the user authentication system is to design a mapping f in such a way that the attacker cannot (partially or completely) recover **S** from **C** and **R**. Different solutions use different mappings f and generate random challenges in a different manner. Unfortunately, some solutions have been found insecure against multiple observations and others are not usable in terms of average login time. A solution that is both secure and usable remains an open problem.

While most previous efforts were based on the assumption that random challenges **C** and responses **R** are fully observable to the attacker, some researchers proposed to make **C** and **R** completely or partially unobservable in order to increase the complexity of solving **S**. The idea of unobservable challenges was proposed by several different groups of researchers independently in 2006 [19, 20, 21]. The unobservable challenge is transmitted via a tactile device that can be sensed by the user but not visible to an observer. Later at CHI'2008, Sasamoto et. al. proposed another design named Undercover [3], in which a part of the challenges is sent to the user via a moving trackball covered by the user's hand. Hayashi et. al. claimed that Undercover is secure against multiple observations as long as the hidden challenges are truly unobservable to the attackers. Hasegawa et. al. from the same research group proposed two alternative designs in [4], one of which uses an audio channel as the carrier of the hidden challenges. Some other researchers have also proposed similar solutions [22, 23, 24].

All proposed solutions claim to achieve perfect secrecy as long as shared secret **S** and hidden challenge **C** are unobservable to the attacker. By observing the response f, the attacker will recover nothing about the secret **S**, which is perfectly encrypted by the challenge **C**. However, all these responses f require from users some form of a cognitive effort (e.g., addition, multiplication, visual recognition, visual search). Unfortunately, as we will show in this chapter, mappings f of different challenge-secret pairs affect non-uniform human behavior. This decreases usability and gives the attacker opportunity to successfully mount a side-channel timing attack. With the side-channel timing attack, the adversary will try to recover the secret **S** simply by analyzing the time a user invests to execute a particular cryptographic function (a mental process as a function of that secret).

In this chapter, we show that the assumptions of a perfectly secure human-authentication protocols based on a secret channel may be unfounded. More specifically, we demonstrate successful side-channel timing attacks on two authentication schemes: the original design of Undercover in [3] and US patent [7] modulo 10. The timing attack on Undercover is based on the design flaws of the GUI interface implementation, and exploits users' nonuniform behavior on how they respond to different hidden challenges. On the other hand, the key component of the timing attack in modulo 10 method is a mechanism trained on the database of response times per ith PIN digit from a relatively large number of users. Our decision to perform the timing attacks on Undercover and US patent was motivated by the fact that they fall in two different categories of authentication methods using a secret chan-

nel: Undercover is an interface-based authentication method where the user visually looks for patterns on the interface to compute the response, whereas modulo 10 is an *interfaceless* (lacking a physical/visual user interface) based authentication method where the user mentally calculates the response (performs modulo 10 additions). In this way we wanted to show that non-uniformities can be found in many aspects of human behavior and that side-channel attacks are not particular to a specific cognitive authentication method.

We also discuss what is generally required to make the proposed mechanisms safe against the side-channel timing attacks. Our investigation of non-uniformity highlights some unique principles that we need to follow in the design of future secure human-computer authentication mechanisms. By following these principles, we will render the existing protocols safe against the timing attacks. Moreover, we will show that in some cases the only way to enhance the security of some cognitive authentication methods is to perturb the usability (e.g. by increasing the entropy/size of the secret **S** and/or by increasing the overall authentication time), which only shows the existence of a strong tradeoff between security and usability in cognitive authentication methods.

In addition, we will also demonstrate a successful intersection attack on the original design of Undercover based on the flaws in the Undercover design [3]. As we will show, this attack can be generalized to the alternative designs [4].

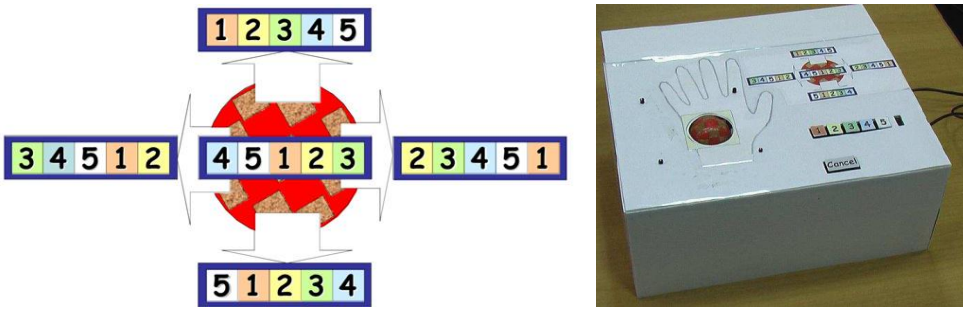
The rest of the chapter is organized as follows. First we give a brief survey of the related work, and then we describe designs of Undercover and modulo 10. Afterwards we describe our attacks and demonstrate their real performance with experimental results. We then propose some enhancements and show how the Undercover and the modulo 10 systems can be made secure against our attacks. Finally, we describe the performance of the intersection attack on Undercover schemes.

2.2 Undercover and Similar Solutions

Since observation attacks are mainly performed in a visual form, most solutions based on unobservable challenges and/or responses aim towards preventing the attacker's visible access to challenges and/or responses. Mainly two kinds of devices are used to achieve this goal: haptic/tactile devices, and audio devices. In the following, we first introduce Undercover [3, 4] in detail. Next, we describe Mod10 method [7]. We conclude this section with a brief overview of some similar solutions [25, 23, 26, 27, 28, 29, 20, 30, 21].

2.2.1 Undercover: Original Design

Undercover is based on the idea of "partially observable challenges": the challenges **C** are split into public challenges $\mathbf{C}_p$ and hidden challenges $\mathbf{C}_h$. For Undercover, the relationship between the challenge, the response and the shared secret becomes $\mathbf{R}=f(\mathbf{C}_p,\mathbf{C}_h,\mathbf{S})$, where



(a) (b)

Figure 2.1. a) Five button layouts of the Undercover prototype, corresponding to the five hidden challenges (Fig. 7 in [3], ©2008 ACM, Inc.), b) the input device box of the Undercover prototype (Fig. 5a in [3], ©2008 ACM, Inc.).

Figure 2.2. A public challenge composed of four pictures and a "no pass-picture" icon (Fig. 9b in [3], ©2008 ACM, Inc.).

only $\mathbf{C}_p$ and $\mathbf{R}$ are observable to the attacker. Clearly, if $\mathbf{C}_h$ and $\mathbf{S}$ have the same number of possible values and the same entropy, it is possible to conceal $\mathbf{S}$ perfectly with $\mathbf{C}_h$.

The device used in Undercover is a haptic device covered by the user's palm, which is supposed to be unobservable to passive attackers. In the prototype system reported in [3], a trackball driven by two servo motors is used as the haptic device. The trackball has five different "vibrate" modes (i.e., hidden challenges): upward rotation, downward rotation, leftward rotation, rightward rotation, and vibration. The five hidden challenges are referred to as "Left", "Right", "Up", "Down" and "Center" in this paper, respectively. Each hidden challenge corresponds to a different layout of five buttons as shown in Figure 2.1(a), which is used to input the response by pressing one of the five buttons "1", "2", "3", "4" and "5" located near the trackball. The input device of the Undercover prototype is a box as shown in Figure 2.1(b).

The Undercover prototype is built on top of a graphical password scheme. The user selects five pass-pictures from an image pool to form his/her password. The system selects 23 more distractor pictures to create the user's portfolio. Each login session is composed of seven challenges, and each challenge contains: 1) a hidden challenge transmitted via the trackball, and 2) a public challenge - four pictures and a "no pass-picture" icon shown on the monitor of the terminal computer (see Figure 2.2).

To avoid potential security problems, the Undercover prototype system is designed so that five public challenges contain one pass-picture and the other two contain no pass-picture.

Each pass-picture and distractor picture in a user's portfolio is shown *once and only once* in a login session. However, [3] does not make it clear how the seven public challenges should be generated in each login session. One may understand that the public challenges are fixed during all login sessions or randomized from session to session. In Section 2.8 we will show insecurity against an intersection attack when randomized public challenges are used.

To make a correct response to a challenge, the user needs to derive a "hidden response" first: 1) if there is a pass-picture in the public challenge, derive the hidden response (1, 2, 3, or 4) according to the position of the pass-picture among the four pictures; 2) if there is no pass-picture, the hidden response is 5 (i.e., the position of the "no pass-picture" icon). Then, the user looks for the hidden response in the button layout corresponding to the hidden challenge and presses the button matching the location of the hidden response in the correct button layout. For instance, if the hidden response is 3 (i.e., the third picture in the public challenge is a pass-picture) and the hidden challenge is "Right", the user needs to press button "2" because the hidden response appears on the 2^{nd} button of the "Right" button layout.

Given one observed login session, the password space of the Undercover prototype is $C_7^5 \times 4^5 = 20480$, which is larger than a 4-digit PIN. Under the assumption that the hidden challenges are unobservable, the Undercover system is considered secure even if an infinite number of login sessions are observed. From an information-theoretic point of view, this is equal to the claim that the password-related information leaked in each login session is 0.

The median login time of the Undercover prototype system is 32 seconds, which is much better than previous solutions. The overall failure rate is 26%, which is rather high but could be significantly reduced as users become more familiar with the system.

2.2.2 Undercover: Alternative Designs

In addition to the original Undercover design, Hasegawa et al. from the same research group proposed two alternative designs [4] . The main goal is to reduce the size of the system. To further simplify the design, a 4-digit PIN is used as the underlying password and the public challenge $\mathbf{C}_p$ is removed.

One design is based on six vibrating tactile devices covered by the user's five fingers and his/her palm. To generate a hidden challenge, one of the tactile devices covered by the user's five fingers will vibrate to select a column of the 2×5 matrix shown in Figure 2.3. The tactile device covered by the user's palm vibrates to determine the row of the 2×5 matrix. The vibrating statuses of all the six tactile devices then determine a hidden challenge – a specific element of the 2×5 matrix. Note that the ten elements of the 2×5 matrix are labeled with numbers from 0 to 9. The hidden challenge is actually a number between 0 and 9, which will henceforth be referred to as the "hidden digit" in this dissertation.

To make a correct response, the user needs to find out his/her current PIN digit in the

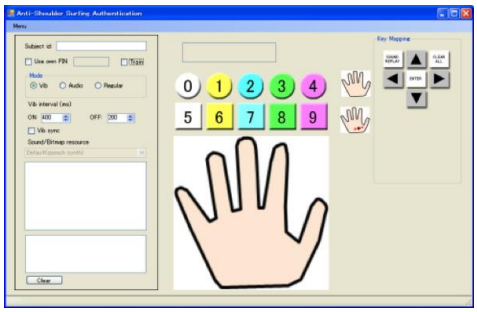


Figure 2.3. An alternative Undercover design (Fig. 1b in [4]).

2×5 matrix and then press the four arrow buttons in Figure 2.3 to show a route from the PIN digit to the hidden digit. This procedure repeats four times so that the user can input all four PIN digits.

Another design proposed in [4] is similar to the tactile one, but the hidden challenges are sent to the user via an audio channel, i.e., via a headphone set.

2.2.3 Mod10 method

In the solution described in US patent [7], the user receives a challenge in form of a random number from interval $(0,1,\ldots,9)$, adds modulo 10 each digit of his genuine PIN to the digits of the random number (performs mental computation to calculate the response), and enters back the outcome via the public keypad. Similarly to Undercover method, this solution also belongs to the partially observable adversarial model; an adversary who can not observe the challenges, can not learn the PIN even if it observes the responses. Although not suggested in the patent [7], the challenge values can be transmitted over an audio channel (e.g., via a headphone) in order to protect them from observation. Surprisingly and to the best of our knowledge, this "modulo-10" method (henceforth referred to as *Mod10*) has never been studied or analyzed before. In this paper, we present results from the usability study of the Mod10 method and we also provide a detailed security analysis.

Mod10 scheme implements the *one-time pad* paradigm. In order to enter the ith digit d_i of his/her PIN, the user receives a challenge c_i (one digit long) selected uniformly at random from the set $(0,\ldots,9)$ via a protected channel (e.g., earphones). In turn, the user adds the two digits modulo 10 (i.e., $r_i = (c_i + d_i) \bmod 10$) and enters back the outcome r_i of this addition via the public channel (e.g., a numerical pad). Compared to the original Undercover scheme, the Mod10 does not require users to perform any visual recognition of challenges and/or responses on the interface to calculate the response. In fact, Mod10 is a non-interface based method where the response is a product of a mental operation (a function of a secret digit d_i and a received challenge c_i).

2.2.4 Similar Solutions

There are some other early designs for human authentication based on haptic/tactile devices with/without using the concept of hidden challenges. The main goal of these systems is to resist shoulder-surfers, the simplest form of observation attacks. The solution in [21] involves pressure of a haptic pen as part of the password, thus making the password input partly unobservable to shoulder surfers. The solution called TAS (Tactile Authentication System) in [20, 30] uses the VT Player tactile mouse to transmit hidden challenges to the user for entering the password without the danger of being observed. The design reported in [29] is very similar to TAS but the VT player tactile mouse is replaced by solenoid pins that can raise and lower their positions. The solution in [25] analyzes haptic information in handwritten signatures to achieve the goal of user identification. Some additional solutions were inspired by Undercover. At CHI'2009, De Luca et al. proposed a scheme called VibraPass, which uses the user's mobile phone as the receiver of hidden challenges (a signal telling the user to make a true or false response) to avoid possible manipulation of the haptic devices by attackers [28]. De Luca et al. noticed a possible timing attack related to "confused waiting" (the user responds slower to "false" hidden challenges due to confusion) that can lead to password disclosure. At CHI'2010, Bianchi et al. proposed a solution called Secure Haptic Keypad (SHK), which combines the tactile device and input buttons to make a uni-modal haptic password [27]. SHK can achieve similar usability to the original Undercover design in terms of average login time. In [23, 26], Bianchi et al. proposed a number of other uni-modal designs based on haptic and audio hidden cues (i.e., hidden challenges) to achieve user identification. Their user studies showed that a shorter average login time and a login error rate can be achieved with the uni-modal designs.

2.3 Timing Attack on Undercover

In theory, Undercover-like solutions can achieve perfect secrecy since the shared secret $\mathbf{S}$ can be perfectly "encrypted" by the hidden challenge $\mathbf{C}_h$. Unfortunately, this is not always true because careless designs could leak information about $\mathbf{S}$ and/or $\mathbf{C}_h$. Our study on the original Undercover design in [3] led to the discovery of such design flaws, which allow an attacker to reduce the password space if a sufficient number of login sessions are collected. We have developed a timing attack on the original Undercover based on a careless design flaw of the button layout, which leads to nonuniform behavior of the user's responses to hidden challenges. In the following, we describe the attack and its real performance verified via user studies on our own implementation of Undercover. We also developed an intersection attack on all Undercover designs [3, 4] that we separately describe in Section 2.8. The intersection attack is based on the design flaws of the original design.

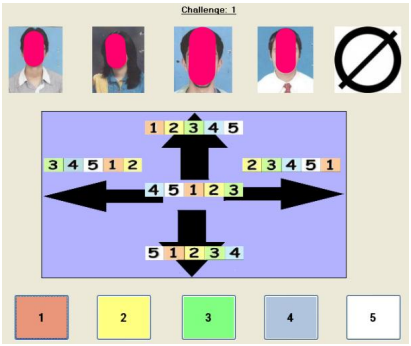


Figure 2.4. Our Undercover implementation.

2.3.1 Our implementation of Undercover

Before introducing the attack, we will first briefly describe how we implemented Undercover and the data for analyzing the performance of the attacks was collected.

To ease our study, we avoided using any special hardware and implemented the whole system in software. We used the audio channel to transmit the hidden challenges and Pass-faces [31] as the underlying graphical scheme. The same button layouts were used as in the original Undercover design. The five buttons are shown as push buttons on our software GUI. The users are allowed to make responses via mouse (by pressing one of the push buttons) or keyboard (by pressing <1>, <2>, <3>, <4> or <5>). Those changes have no influence on the security of Undercover against the proposed attacks. Figure 2.4 shows what a public challenge looks like in our implementation. We mask the faces in Figure 2.4 to avoid violating the affected people's privacy.

We performed user studies on our implementation at two universities located in two countries: the University of Split in Croatia and the National University of Science and Technology (NUST) in Pakistan. Neither university requires IRB reviews on research work involving human subjects, so the user studies were carried out without such a review. The University of Konstanz has no established policy on usable security research, but an approval from the Chair of the Ethics Committee was secured. Although a formal IRB review was not required, we took all possible measures to make sure that all legal and ethical issues we could think of were properly handled. For instance, all users were well informed in advance (before the user studies) about the purpose of the study and how the data would be processed and used.

Part of the reason why we ran the user studies in two different countries is to see if users with different cultural backgrounds and different races share similar nonuniform human behavior that makes the timing attack a universal attack. In total, 28 users participated. All users are university students and staff members in departments of electronic engineering and computer science. Among the 28 users, 19 performed the study at the University of Split in Croatia and 9 at the National University of Science and Technology (NUST) in Pakistan. Gender ratio is 22:6 (22 males and 6 females). The age of the 28 users ranges from 20 to

40. All participants were volunteers who were asked to help our research, and none of them was economically compensated/motivated strengthening our belief that our data is not biased towards positive results of our proposed attacks.

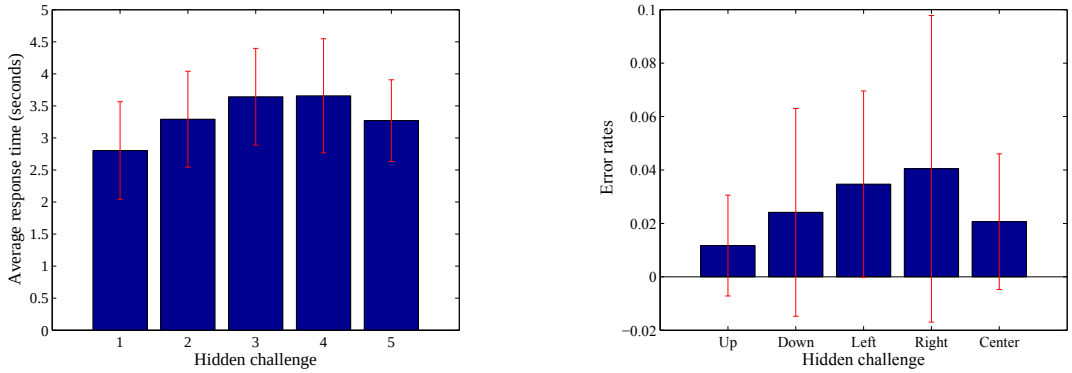
At the beginning of the user studies, the users were given a short tutorial of the system. A questionnaire was issued to each user to collect personal information and knowledge on computer/web technology and password security. Then, they were asked to log in at least once a day during a one-month period. To have a better control over the environment of the user studies, we set up a computer running the Undercover system in our labs and users had to physically come into our labs to perform the logins. Users who forgot to come within 24 hours were automatically reminded via emails. Despite the reminding mechanism, not all users followed our request strictly. Consequently, at the end of our user studies different users had different numbers of recorded login sessions. However, no user was dropped during the course of the user studies. The minimum number of user login attempts is 20 while the maximum number is 66 and the median is 26.5. In total, we collected 918 login attempts, among which 771 are successful ones, leading to an overall login success rate around 84%. The login success rates of all users range from 66.67% to 100% and the median rate is 84.82%. Among all the 28 users, 18 used the keyboard as the input device while others used the mouse. Login data from 28 users were stored in XML database for further processing.

Compared with the original Undercover implementation in [3], our implementation has comparable usability in terms of average login time. The median login time is 30.1 seconds, slightly shorter than the original Undercover implementation (32 seconds). Note that the average login time steadily decreased as the users became more familiar with the system. After 20 logins, the median login time decreased to 21.8 seconds.

2.3.2 Timing Attack on Undercover: Non-uniform human behavior in responding to different hidden challenges

Observing the five button layouts in Figure 2.1(a), we can see that the button layout corresponding to "Up" hidden challenge is "12345", exactly the original layout of the five buttons that the user needs to press. In comparison, the other four button layouts are all circularly rotated editions of the original button layout. Since users do not need to do button rotation for the original button layout, we hypothesized that they may make responses to "Up" hidden challenges faster and with a lower error rate, compared to the other four hidden challenges. Our user study confirmed this hypothesis. Figure 2.5(a) and Figure 2.5(b) show the average response times and error responses rates for all users to the five hidden challenges, respectively. Paired t-tests revealed that the difference between the user's responses to "Up" hidden challenges in relation to other hidden challenges is significant at 5% level.

This nonuniform human behavior in responding to different hidden challenges inspired us to propose a timing attack. Denoting the 28 pictures by 28 integers (from 1 to 28), the



(a) (b)

Figure 2.5. a) The nonuniform human behavior in the average response time to different hidden challenges, b) the nonuniform human behavior in the average error response rate to different hidden challenges.

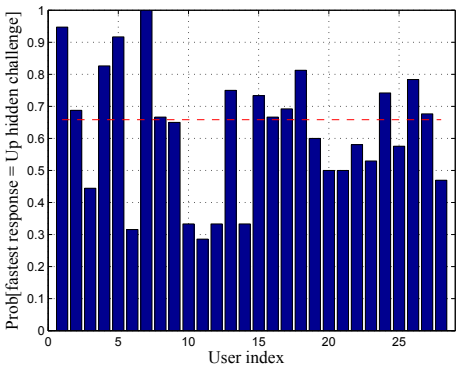
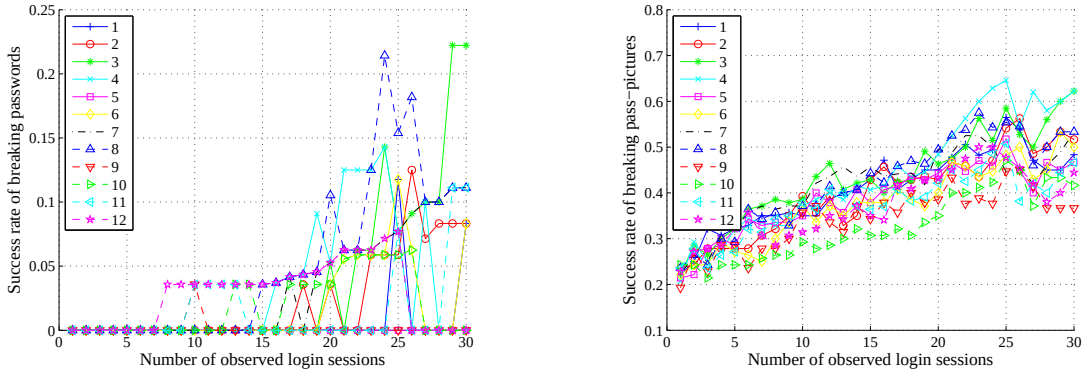


Figure 2.6. Probability that the fastest response in a login session corresponds to an "Up" hidden challenge.

attack works as follows.

- *Step 1*: Create 28 counters, $C_1,\ldots,C_{28}$, for 28 pictures, and initialize all of them with 0.
- *Step 2*: For each observed login session, take the fastest response and assume that it corresponds to an "Up" challenge. Then, if the corresponding public challenge contains a pass-picture i, increase C_i by one.
- *Step 3*: Rank all the pictures according to the values of the 28 counters, and take the top five pictures as the five pass-pictures forming the password. If there is more than one way to select the top five pictures (which can happen when some pictures have the same counter value), random shuffle all pictures with the same counter value as the fifth one, re-rank all the 28 pictures, and then take the new top five as the pass-pictures.



(a) (b)

Figure 2.7. Success rates of breaking passwords and pass-pictures by applying the timing attack to real login data.

The random shuffling process is to avoid the bias towards pictures with smaller indices.

Note that the random shuffling process in Step 3 means the timing attack may produce different results for different runs.

To further improve the performance of the aforementioned timing attack, two additional measures can be further adopted: 1) negative penalty mechanism - for each distinguished decoy picture i in *Step 2*, decrease C_i by one; 2) multiple fastest responses - use the fastest m=2 or 3 responses in *Step 2*. Both measures can potentially increase differences between counter values of pass-pictures and decoy pictures.

In the following, we describe the performance of the timing attack by applying it to the real login data collected in our user studies.

Real performance of the timing attack

We applied the timing attack to the login data collected in our user study, in order to verify its real performance under the following 3×2×2=12 different settings:

- Number of fastest response(s): m=1, 2, 3;
- Negative penalty mechanism: on, off;
- Login sessions used: all, successful ones only.

The performance of the above 12 settings of the timing attack on real login data is shown in Figure 2.7. Among all the 12 settings, Settings 3 (solid line marked with "*"), 4 (solid line marked with "x") and 8 (dashed line marked with "△") have a better performance, which correspond to "m=1, without negative penalty, successful logins only", "m=1, with negative penalty, successful logins only", and "m=2, with negative penalty, successful logins only", respectively.

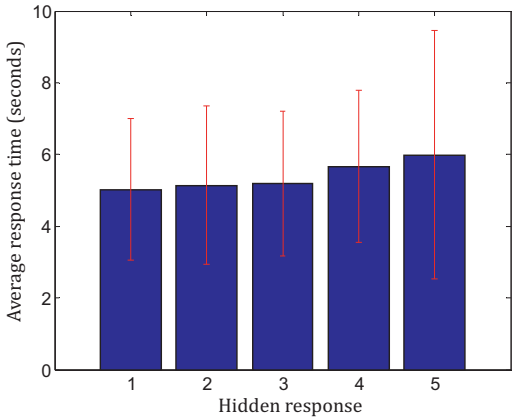


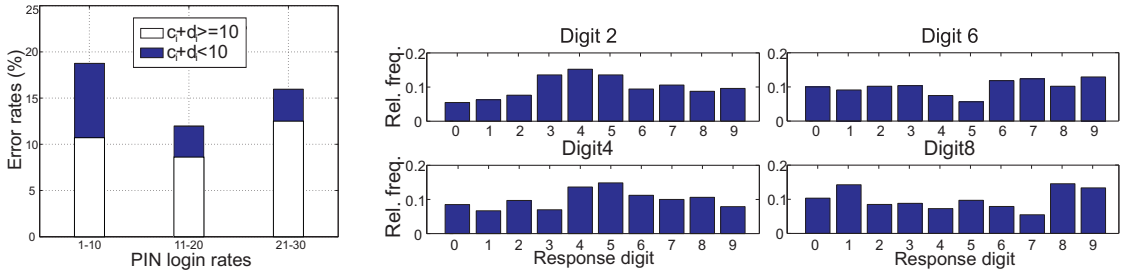
Figure 2.8. Average response times with respect to different hidden responses.

Yet another potential timing attack

The human behavior has many different kinds of nonuniformities we may exploit. Yet another nonuniformity we noticed is that most users tend to respond more slowly to public challenges with no pass-picture. Figure 2.8 shows the average response times of all users with respect to the five different hidden responses. One can see that the average response time is longer when the hidden response is "5", i.e., when the public challenge does not contain a pass-picture. This phenomenon can be explained by the fact that the user has to look at all four pictures (potentially twice) to make sure that there is indeed no pass-picture. We tested this new timing attack using the same strategy: 1) pick the m slowest responses in each login session; 2) assuming this response corresponds to the public challenge with no pass-picture, decrease the counters of the four distinguished decoy pictures by one; 3) rank the counters of all pictures and pick the top five ranked pictures to form the password. Simulated attacks on the real login data did not produce good results. None of the users' passwords was completely broken, and the success rate of breaking passpictures ranges from 0 to 0.4. We attribute the failure of the attack to the larger variance of the response time to the public challenge with no pass-picture (which can be seen in Figure 2.8). Although this timing attack was unsuccessful on our dataset, it remains a potential threat since some pass-pictures may still be broken.

2.4 Timing Attack on Mod10 Method

Mod10 scheme implements the *one-time pad* paradigm since the shared secret is perfectly "encrypted" by the hidden challenge. To calculate the response the user is required to perform modulo 10 computations of her secret digits with the received hidden challenges. However, there is always a potential risk that some kind of nonuniformity exists during modulo 10 computation so that an effective timing attack can be developed based on it. Our study



(a) (b)

Figure 2.9. (a) The average PIN-entry error rates (%) for 20 users using Mod10 method. (b) Relative frequency with which a given response digit appears within $\ell = 4$ fastest response digits, for PIN digits: 2, 4, 6 and 8.

of the Mod10 method in [7] led to the discovery of such flaws, which allows an attacker to reduce the PIN space if a sufficient number of login sessions are collected. We describe the attack and its real performance verified via user studies in our own implementation of Mod10.

2.4.1 Implementation of Mod10 method

We implemented the Mod10 method as a web application. For each participant, the same test statistics (overall login time, error rates) were collected and stored in a central database. The test evaluation consisted of two phases: a *training phase* and an *authentication phase*. In the training phase participants learned how to use the respective methods (five successful logins per method). The authentication phase served as the actual test authentication method. The participants were asked to successfully login 30 times per method; there have been no other incentives on the part of the testers (e.g., to achieve faster login times). To have a better control over the environment of the user studies, we set up a computer running the system in our labs.

In total, 58 users participated. All the users are university students or staff members in departments of electronic engineering and computer science. Gender ratio is 39:19 (39 males and 19 females). The ages of 40 users range from 18 to 25 years old, 10 users are within the age group 26-40, while 8 users were older than 40 years. All the participants were volunteers who were asked to help our research, and none of them was economically compensated/motivated. Here we present the results of the timing attack on Mod10 method. In total we collected 2491 login attempts, among which 2125 are successful ones, leading to an overall login success rate of about 85%. The average login time is 10.39 seconds, which places Mod10 method among the fastest authentication methods.

More detailed results from the performance test indicate that the major sources of errors in Mod10 scheme are the cases when the sum of two numbers exceeds 10 (Figure 2.9(a)). We hypothesize that these additions (over 10) have longer average response

times. Please note that the results of additions over 10 primarily consist of additions with large-valued numbers (e.g., 8+7). The study of these and similar cognitive processes while solving simple-arithmetic problem has been a subject of research in the last couple of decades [32, 33, 34, 35]. It is known that arithmetic operations with large-valued numbers achieve larger error rates and score slower response times (compared to arithmetic operations with small-valued numbers). Basically, simple arithmetic problems are usually calculated faster than large ones, because users apply different mental strategies: in small-valued problems users apply a direct mapping (retrieval) process which is faster than the actual addition procedure performed in solving large-valued problems [35]. Indeed, from the results of our study, the shortest response time (large bars in Figure 2.9(b)) for the respective PIN digit occurs in cases in which the users receive challenges from the set $\{0,1,2\}$ i.e. for "easy" additions with challenges 0, 1 and 2. More precisely, in Figure 2.9(b) we plot the corresponding relative frequency for PIN digits 2, 4, 6 and 8. The relative frequency is generated in such a way that for the fixed PIN digit d_i we count how many users (with PIN digit(s) d_i) have a given response digit within their ℓ "fastest" response digits (in our case $\ell = 4$). The shortest response is also achieved for cases in which the user's PIN digit equals to the received challenge value (e.g., 6+6), formally known as *a tie problem* [36].

These observations on nonuniform human behavior in Mod10 method inspired us to run a side-channel timing attack which allows the attacker to reduce the PIN digit space (the space of possible PIN digits as a solution). On the high level, the timing attack is based on the observation of two attributes: a vector $\mathbf{r} = (r_1, r_2, \ldots, r_\ell)$ representing the $\ell \leq 10$ fastest response digits, and a scalar t_r representing the fastest response time, both for the unknown/sought class (PIN digit). The timing attack presents a classification problem in which the observed data is assigned to the one of predefined classes (PIN digits) that represent different decisions. These decisions are based on two classifiers, where the first classifier exploits only the information conveyed by the vector $\mathbf{r}$ and the second is based only on the scalar t_r. We apply the approach based on the *naive Bayesian classifier*. To test the performances of our classifiers we used the technique known as *leave-one-out crossvalidation* technique, where every classifier is first trained on the login results from 57 users (out of the total of 58) and then tested on the single remaining user. In the sequel, we first describe each classifier, which is followed by the description of the classifier combination technique that combines the results of the individual classifiers. At the end, we present the obtained classification results.

2.4.2 Timing attack on Mod10

As already discussed, we used two classifiers trained over the training set. In our classification mechanism we used classification combination technique, that merges the results of individual classifiers to obtain a better estimation performance. Many approaches like boost-

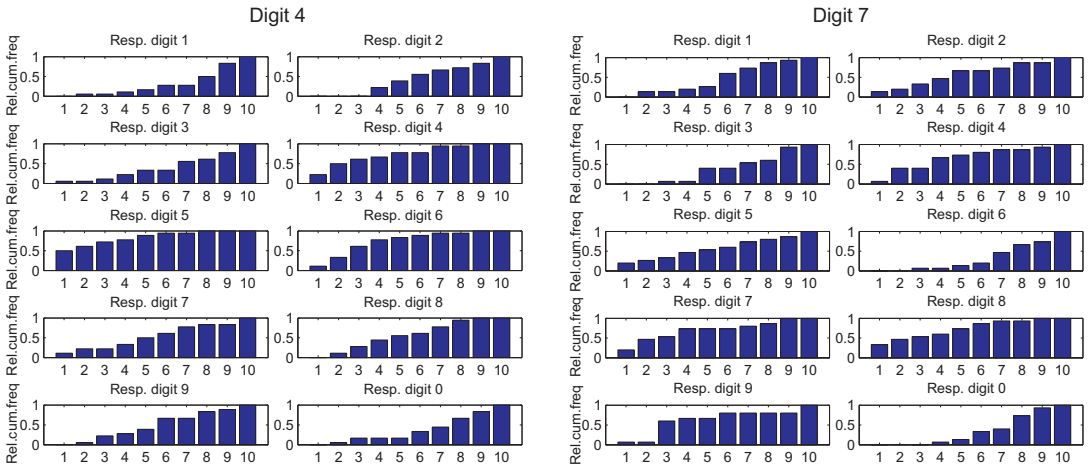


Figure 2.10. Relative cumulative frequency with which a given response digit appears within ℓ fastest response digits ($\ell \in \{1,\ldots,10\}$), for (a) PIN digit 4, and (b) PIN digit 7.

ing [37, 38] or bagging [39] exist that use classification labels. In our classification, we used a continuous output from each classifier that was normalized to the interval [0 1] (posterior probability). These values (for each class - PIN digit) were combined using an algebraic combination rule (e.g. majority voting, min/max/sum/product rule or other techniques) [40]. Here we used sum decision rule that combines posterior probability from each classifier, because it achieves the best classification results [40]. Before we describe the implementation of the classifier combination technique, we first describe in details the implementation of each classifier.

Classifier Trained on Fastest Response Digits

The first classifier exploits the property of "easy additions" with small-valued challenges (e.g., 0, 1 and 2) and tie problems (e.g., 7+7) that both achieve fast response times. As shown previously in Figure 2.9(b), easy additions with challenges 0, 1 and 2 will have different distributions of fastest response digits for different PIN digits. To learn these distributions of response digits for every class value (PIN digit d_i) the classifier is previously trained on the login results from 57 out of the total of 58 users (*leave-one-out crossvalidation* technique) in the following way. For every PIN digit d_i ($d_i \in \{0,\ldots,9\}$) repeat the following steps:

- for every response digit r^i_j ($r^i_j \in \{0,\ldots,9\}$) create 10 counters, $C^i_{j,1},\ldots,C^i_{j,10}$, and initialize them to 0,
- for all the users from the training set of 57 users whose PIN comprises digit d_i, and for all such digits d_i for the given user repeat the following steps:

- rank the response digits r^i_j (for the given PIN digit d_i) according to the values of their fastest response times,
- if the response digit r^i_j falls within ℓ fastest response digits, increment the counters $C^i_{j,\ell}$ to $C^i_{j,10}$ by one (note $\ell \in \{1,\ldots,10\}$).

As a result, we obtain, for every PIN digit d_i, the cumulative frequency $C^i_{j,\ell}$ that indicates the number of times a given response digit r^i_j falls within ℓ fastest response digits. Dividing $C^i_{j,\ell}$ by $C^i_{j,10}$ we get the corresponding relative cumulative frequency. Figure 2.10 shows the relative cumulative frequency of all the response digits for two PIN digits 4 and 7. It is interesting to observe in this figure that the response digits $4,5$ and 6 for the PIN digit 4, and the response digits $7,8$ and 4 for the PIN digit 7 are characterized by higher relative cumulative frequencies at small values of ℓ compared to the other response digits. This indicates that those response digits generally have faster response times for the corresponding PIN digit. Please note that these responses correspond to so-called *easy additions* (additions with $0,1$ and 2) and *ties* (e.g., $7+7$), for which users mainly achieve faster response times [36]. Our first classifier uses the resulting frequency distributions to perform the classification task of the given feature vector $\mathbf{r}$; recall, $\mathbf{r}$ represents ℓ fastest response digits corresponding to the given PIN digit position (but an unknown PIN digit value).

Given the feature vector $\mathbf{r} = (r_1, r_2, \ldots, r_\ell)$ of $\ell \leq 10$ fastest response digits (for the given PIN digit position), our classifier first estimates the posterior probability $P(d_i|\mathbf{r})$ for all PIN digits d_i and then selects into the final *candidate set* $n \leq 10$ digits d_i that have the highest posterior probabilities. In our classifier we use the classical naive Bayes technique/assumption: we assume that conditioned on a class d_i, the feature vector elements $(r_1, r_2, \ldots, r_\ell)$ are mutually independent. Using these assumption we can express $P(d_i|\mathbf{r})$ as follows:

$$P(d_i|\mathbf{r}) = \frac{\prod_{m=1}^{\ell} P(r_m|d_i) \cdot P(d_i)}{\sum_{l=1}^{10} \prod_{m=1}^{\ell} P(r_m|d_l) \cdot P(d_l)} = \frac{\prod_{m=1}^{\ell} P(r_m|d_i)}{\sum_{l=1}^{10} \prod_{m=1}^{\ell} P(r_m|d_l)}, \tag{2.1}$$

with $P(d_i) = 1/10$ for all digits d_i. Since the denominator in the expression (2.1) is the same for all the classes, maximizing the posterior probability $P(d_i|\mathbf{r})$ is equivalent to maximizing the numerator (*the likelihood function*), that is:

$$P(d_i|\mathbf{r}) \propto \prod_{m=1}^{\ell} P(r_m|d_i). \tag{2.2}$$

The class conditional probabilities $P(r_m|d_i)$ are estimated from the relative cumulative frequency distributions (obtained in the training phase), that is, we assume that the following holds:

$$P(r_m|d_i) \simeq \frac{C^i_{j,\ell}}{C^i_{j,10}}, \tag{2.3}$$

where the index j corresponds to the response digit r^i_j satisfying $r^i_j = r_m$. By plugging the approximation (2.3) into (2.2) we obtain the expression that for the observed feature vector $\mathbf{r}$ assigns a positive score to each class d_i, based on which we can rank the classes. Unfortunately, the straightforward application of this expression results in a great number of zero scores that are not very informative for our purposes (ranking of the classes). The main reason we observe so many zero scores is the fact that the product in (2.2) evaluates to 0 if $P(r_m|d_i) = 0$ (i.e., the corresponding counter $C^i_{j,\ell}$ is 0) for at least one response digit r_m (out of ℓ) from the observed feature vector $\mathbf{r}$. Due to the small number of login attempts and the fact that test users make errors during the login process, it is likely that a normally "slow" response digit will appear in the observed feature vector $\mathbf{r}$ normally comprising only the fastest response digits; it is likely that for this "slow" response digit the corresponding counter $C^i_{j,\ell}$ will be 0 in the training set (for the given PIN digit d_i). To avoid this problem, we modify our initial classifier in such a way that instead of calculating the scores according to the product rule (2.2), we calculate the score for each class (given the observation $\mathbf{r}$) by simply summing up the probabilities $P(r_m|d_i)$ ($m = \{1,\ldots,\ell\}$). More precisely, for the given feature vector $\mathbf{r}$ we assign a score $S(d_i|\mathbf{r})$ to each class (PIN digit) d_i, where $S(d_i|\mathbf{r})$ is calculated as follows:

$$S(d_i|\mathbf{r}) = \sum_{m=1}^{\ell} P(r_m|d_i) = 1/C^i_{j,10} \sum_{m=1}^{\ell} C^i_{j,\ell}. \tag{2.4}$$

Finally, for the observed feature vector $\mathbf{r}$ our classifier outputs into the final *candidate set* $n \leq 10$ digits d_i that have the highest score $S(d_i|\mathbf{r})$; the ties are broken by selecting candidates at random. Please note that we can normalize the obtained scores to convert them into proper posterior probabilities. More precisely, for each class d_i we define the corresponding posterior probability as follows:

$$\widetilde{P}(d_i|\mathbf{r}) = \frac{S(d_i|\mathbf{r})}{\sum_{i=i}^{10} S(d_i|\mathbf{r})}. \tag{2.5}$$

Ranking the classes according to the posterior probabilities $\widetilde{P}(d_i|\mathbf{r})$ is equivalent to ranking them according to the corresponding scores.

In Figure 2.11 we give a classification example for the observed feature vector $\mathbf{r} = (2,9,\ldots,3)$ (ℓ fastest response digits for the fixed PIN digit position). The trained classifier calculates, on the input feature vector $\mathbf{r} = (2,9,\ldots,3)$, the score $S(d_i|\mathbf{r})$ for all possible PIN digits $d_i \in \{0,\ldots,9\}$ and ranks them according to their score (the highest score goes first). In our example the highest scores are associated with the digits 1 and 9; we consider them to be the most probable solutions for the observed feature vector. Later in this section we study the effectiveness of this classifier, and show that it performs very well. Next, we describe our second classifier.

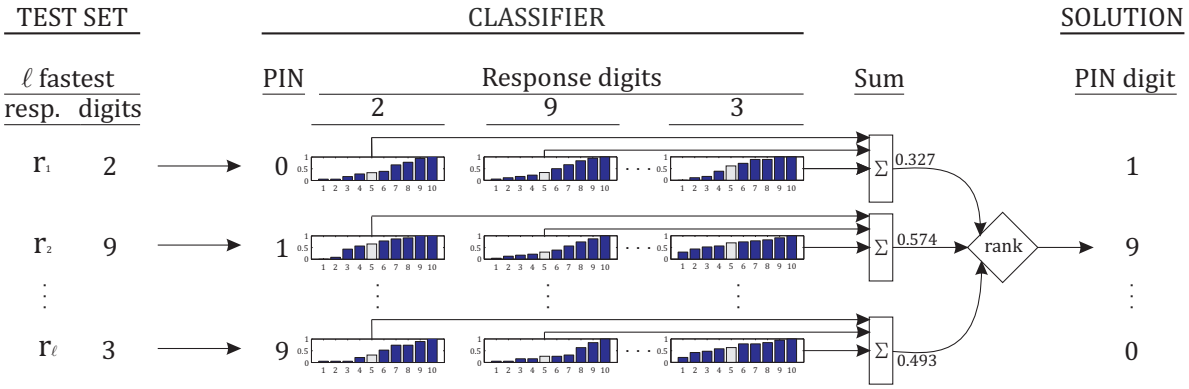


Figure 2.11. Classification of the input feature vector $\mathbf{r} = (2, 9, \ldots, 3)$ that holds $\ell = 5$ fastest response digits for an unknown class (sought PIN digit).

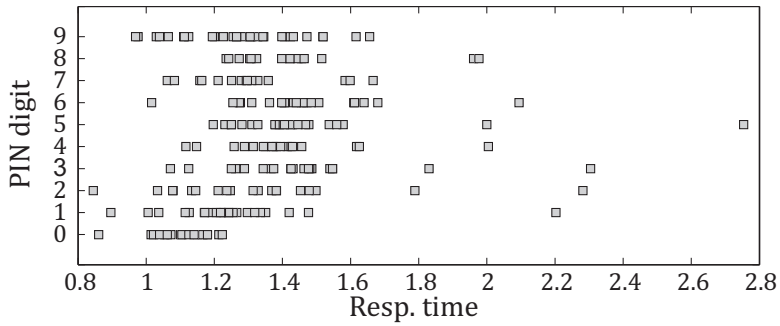


Figure 2.12. The calculation results indicate that additions with small single integers achieve fastest response times (PIN digits 0,1 and 2).

Classification Based on Fastest Response Times

In our second (Bayesian) classifier we also use response digits as an observed feature, but only implicitly: here we are only interested in the fastest response time (t_r) among all the response digits while we ignore the identity of the response digit for which the user achieves this response time. The second classifier uses the phenomenon called *problem-size effect* in simple arithmetic operations (with two single integers) [35]. Basically, arithmetic operations with small-valued numbers are solved faster (e.g., $2+3$) than with larger-valued numbers (e.g., $7+4$), simply because of different strategies applied during the response calculation; direct mental mapping technique that is characteristic for small-valued additions is faster than the procedural techniques of addition that use large-valued additions [32, 33]. In Figure 2.12 we show the distribution of the fastest response times achieved by our testers for single integer additions with different PIN digits. Indeed, the results of fastest response times for all PIN digits in Figure 2.12 show that additions with small single (one-digit) integers achieve faster response times. Please note that users with PIN digit 0 on average achieve faster response times. This is because the response equals to the received challenge value $c_i = r_i$ and users simply enter the response upon receiving the challenge. It is also interesting to observe

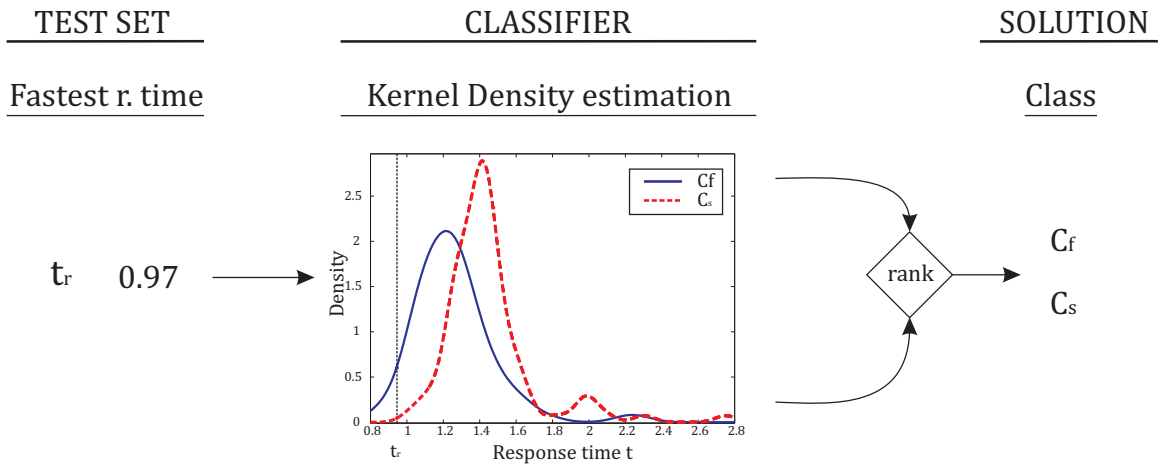


Figure 2.13. Classification of the input feature t_r - the fastest response time.

that some users with PIN digit 9 also achieve fast response times (faster than other large-valued digits). After the discussion with users we realized that they have developed a faster procedural technique (faster than the procedural operation of addition) while calculating the response. Users realized that the response r_i is actually a challenge value c_i (for challenges within $\{1,\ldots,9\}$) subtracted by one digit (e.g., $(9+c_i) \mod 10 = c_i - 1 = r_i$) and that this technique requires less time to calculate the response. Next we describe the implementation and the work of our second classifier trained on fastest response times.

If we take a look at Figure 2.12 we can see that multiple classes (PIN digits) have overlapping distributions of response times, and accordingly, the observed fastest response time t_r can be attributed to multiple basic classes (PIN digits). Similarly to [41], instead of assigning an observed feature t_r to a specific class d_i, it may be assigned to the meta-class C_j that comprises of multiple basic classes (PIN digits). In our classifier we used two meta-classes C_f and C_s, where meta-class C_f is comprised of "fast response digits" $\{0,1,2,7,9\}$, while C_s is comprised of "slow response digits" $\{3,4,5,6,8\}$.

To classify an observed feature t_r into one of the two above meta-classes, we first calculate the posterior probabilities $P(C_j|t_r)$ for $j \in \{f,s\}$ using Bayes' theorem, i.e.:

$$P(C_j|t_r) = \frac{P(t_r|C_j)P(C_j)}{P(t_r)}. \tag{2.6}$$

Note that the following holds for the prior probabilities: $P(C_f) = P(C_s) = 1/2$. We do not directly estimate the distribution $P(t_r)$ in the denominator, but simply normalize so that the sum of $P(C_f|t_r)$ and $P(C_s|t_r)$ is one.

Concerning the class-conditional probabilities $P(t_r|C_j)$ we take the same approach as in [42]. In that work the authors introduce a Flexible Bayes learning algorithm for estimating continuous distributions in Bayesian classifiers. While classical Naive Bayes approach uses a single Gaussian to model each class-conditional distribution (using the data from the training

set), in Flexible Bayes the conditional probabilities are estimated using the *kernel estimation* with Gaussian kernel (i.e., uses multiple Gaussian distributions). More precisely, for each meta-class C_j we obtain the probability density $gk_j(t_r)$ by training multiple Gaussian kernels on the training dataset that belongs to the meta-class C_j. We use Gaussian kernel estimation via diffusion [43] to obtain the probability density $gk_j(t_r)$. In Figure 2.13 we show two probability densities estimated using kernel density estimation trained on the training dataset from our tests with users. Then the class-conditional probability $P(t_r|C_j)$ can be estimated from $gk_j(t_r)$ as follows:

$$P(t_r \leq T_r \leq t_r + \Delta t|C_j) = \int_{t=t_r}^{t_r+\Delta t} gk_j(t)dt \approx gk_j(t_r)\Delta t \text{ (def. of definite integral and small } \Delta t). \tag{2.7}$$

Then for some very small constant Δt the following holds: $P(t_r|C_j) \approx gk_j(t_r)\Delta t$. Using this approximation, the fact $P(C_f) = P(C_s) = 1/2$ and by normalizing we obtain: $P(t_r) \approx \Delta t/2(gk_f(t_r) + gk_f(t_r))$. By plugging this into the expression (2.6) we finally obtain:

$$P(C_j|t_r) = \frac{gk_j(t_r)}{gk_f(t_r) + gk_s(t_r)},\ j \in \{f,s\}. \tag{2.8}$$

As shown in the example in Figure 2.13 our Bayesian classifier assigns the observed feature ($t_r = 0.97$s) to the meta-class (C_f) with the higher posterior probability. Finally, assuming that every digit d_i within the meta-class C_j is uniformly distributed [44], the posterior probabilities $P(d_i|t_r)$ for every class within the subset is obtained using the following rule:

$$\text{if}\quad d_i \in C_j \quad \text{then}\quad P(d_i|t_r) = \frac{P(C_j|t_r)}{|C_j|} = \frac{1}{5} \cdot \frac{gk_j(t_r)}{gk_f(t_r) + gk_f(t_r)},\ j \in \{f,s\}. \tag{2.9}$$

Later we show the effectiveness of the second classifier, but first we describe a classifier combination rule that is used to combine the results of the two presented classifiers.

Classifier Combination Using a Sum Decision Rule

We use the sum decision rule to combine the output posterior probabilities from each classifier, because the sum rule technique achieves the best classification results (compared to min/max/product rule) [40]. According to the sum rule, the posterior probabilities ($\widetilde{P}(d_i|\mathbf{r})$ and $P(d_i|t_r)$) from the first and the second classifier, respectively, and the observed features $\mathbf{r}$ and t_r can be combined as follows [40]:

$$P(d_i|\mathbf{r},t_r) \approx const. + \widetilde{P}(d_i|\mathbf{r}) + P(d_i|t_r). \tag{2.10}$$

As before, using the resulting posterior probabilities $P(d_i|\mathbf{r},t_r)$ we can rank the classes d_i and select the ones with the highest rank into the final candidate set. Next, we present the classification results for each classifier separately, and the classification results for the

Table 2.1. Effectiveness of the second classifier trained on the fastest response times.

meta-class\PIN d.	0	1	2	3	4	5	6	7	8	9
C_1	0.7847	0.6214	0.4820	0.4657	0.3831	0.3334	0.3740	0.5281	0.3553	0.5852
C_2	0.2153	0.3786	0.5180	0.5343	0.6169	0.6666	0.6260	0.4719	0.6447	0.4148

combined classifier.

Effectiveness of the Classifiers

In this section we study the effectiveness of the proposed classifiers. Recall, our classifier, based on the observed attributes $(\mathbf{r}, t_r)$ of an unknown PIN digit, outputs $n \leq 10$ most likely candidate PIN digits, i.e., a *candidate set* of size n. Then we can measure the effectiveness of a given classifier by estimating the probability that the unknown PIN digit will fall into the output candidate set. Please note that a random guessing strategy will be successful with probability at most $n/10$ for a candidate set of size n. So we can measure the effectiveness of our classifier by comparing it against a random guessing strategy. Figure 2.14 presents the effectiveness of the first classifier (the one that is based on first ℓ fastest response digits). To test the performances of our classifiers we used the *leave-one-out crossvalidation* technique, where every classifier is first trained on the login results from 57 users (out of the total of 58) and then tested on the single remaining user. All results presented in this section are averaged over all tested users (58 test users totally). It shows the probability that a tested unknown digit d_i will fall in a candidate set of size n. Figure 2.14 shows the results for $\ell = 2,3,4,5$ fastest response digits. As can been seen, the first classifier performs significantly better than a random guessing. For example, for $\ell = 5$ and the candidate set size $n = 4$ the first classifier achieves 65% better results than pure random guessing. Moreover, the candidate set of size $n = 7$ hold a sought PIN digit with probability around 0.88 (almost certainly).

The effectiveness of the second classifier is summarized in Table 2.1. It shows the probability that the PIN digit d_i will or will not be assigned to the correct meta-class. As expected,

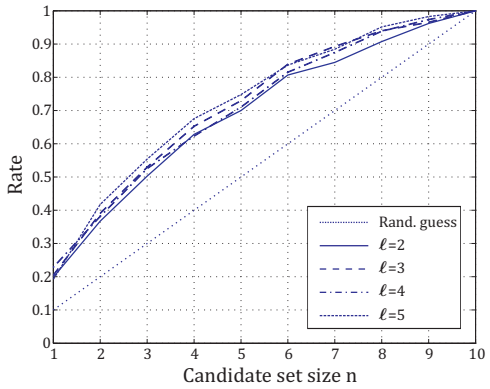


Figure 2.14. Effectiveness of the first classifier for $\ell = 2,3,4,5$ fastest response digits. Dotted line presents a random guessing attack.

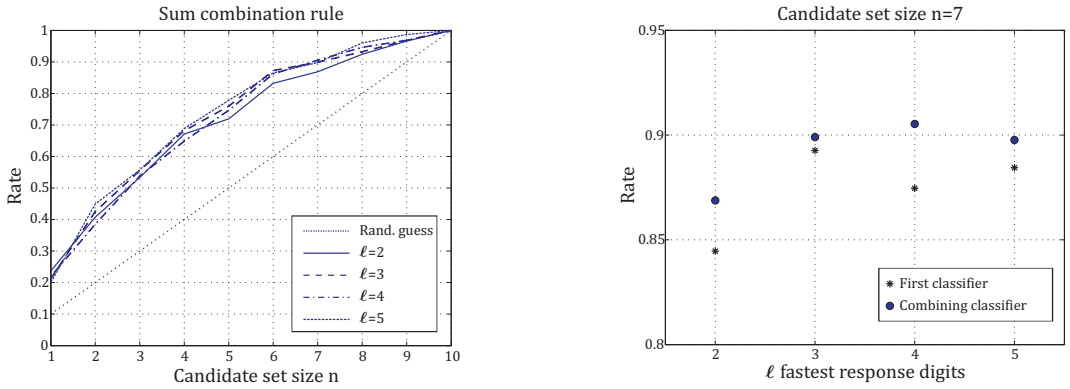


Figure 2.15. (a) Effectiveness of the sum combination rule that combines the results of two classifiers: one trained on fastest response digits and the second one on fastest response times. (b) Combining classifier (circle) achieves better results than the first classifier (star).

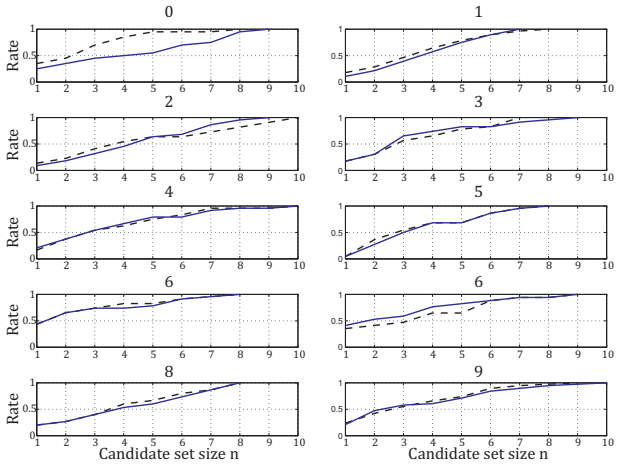


Figure 2.16. Effectiveness of the first classifier (solid line) and the sum combination rule (dashed line) for a specific PIN digit: probability with which a tested PIN digit d_i appears within n most-likely PIN digits.

digits with the fastest response times within the meta-class C_f (e.g., 0,1 in Figure 2.12) and digits with the slowest response times within C_s (e.g., 5,6 in Figure 2.12) have higher probability to be classified to the correct meta-class; on the other hand for PIN digits $2, 3$ and 7 the classifier does not perform better than random guessing.

After combining the results from each classifier using the sum combination rule, we can see a slightly better performance in the classification correctness as shown in Figure 2.15(a). For example, for the same parameters as before ($\ell = 5$ and $n = 4$) our combining classifier performs 72% better than random guessing (and 7% better than the first classifier). Generally, the combining classifier performs better than the first classifier for almost all candidate set sizes n; in Figure 2.15(b) we compare the combining classifier against the first classifier for a candidate set size of $n = 7$. A more detailed comparison is given in Figure 2.16, where

we compare the performances of the two classifiers for every PIN digit. In is interesting to observe in this figure that while the first classifier performs poorly for the PIN digit 0, the combining classifier significantly increases the predictive performances for the same digit thanks to the discriminatory power of the second classifier. Figure 2.16 also shows that this improvement in the discriminatory power for the PIN digit 0 does not come at the cost of reduced predictive power for other PIN digits.

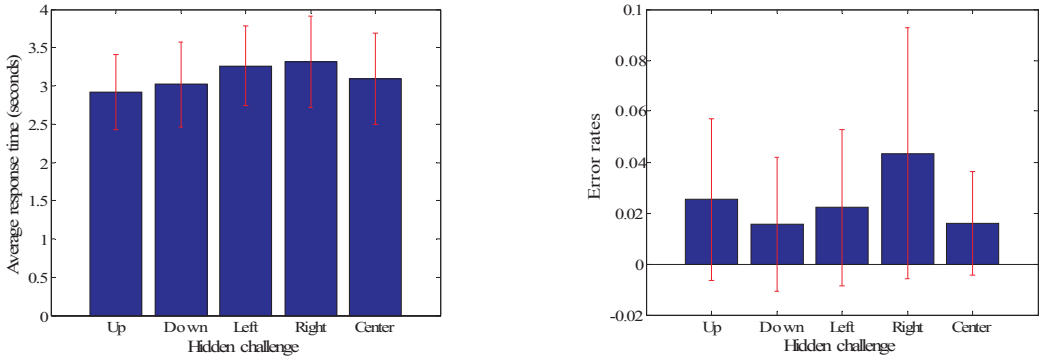
We next estimate the amount of information about an unknown PIN that an adversary can extract using our combining classifier. Let us denote with n^* a candidate set size that holds and unknown PIN digit with the probability above 90%. Then, $log_2(10/n^*)$ is the approximate number of extracted bits per PIN digit. Referring back to Figure 2.15(b) we can read that the combining classifier for a candidate set of the size $n = 7$ and $\ell = 4,5$ fastest response digits will comprise the sought unknown PIN digit with probability slightly over 90%. In Figure 2.16 we can see that a candidate set of size $n = 7$ will almost certainly hold a correct unknown PIN digit for almost all possible candidate PIN digits (except for the digits 2 and 8). Therefore, we conclude that an adversary can extract approximately $log_2(10/7) \approx 0.5$ bits per PIN digit. Accordingly, the adversary can extract approximately 2 bits of information about an unknown 4 digit PIN, i.e., the PIN entropy is reduced from $log_2 10 \approx 13.3$ bits to 11.3 bits. By observing more login session (but still a polynomial number) we expect even more significant reduction in the PIN entropy. Indeed, as can be seen from the aggregated results (over all the test users) in Figure 2.9(b), observed response times for fastest response digits completely reveal an unknown PIN digit, resulting in a zero entropy.

2.5 Enhancing Undercover

Analysis of the timing attack on Undercover in Section 2.3 reveals a general principle that is necessary to follow in order to eliminate the threat of non-uniformity in human behavior that leads to the timing attack on Undercover-based authentication methods.

1. *The information from public challenges should be made equally difficult for humans users to handle.*

In the case of Undercover we should randomly shuffle five buttons layout. Randomly shuffling them is a simple way to achieve this goal, and it can be done dynamically for each challenge in order to minimize any potential nonuniformity of users' response time to different hidden challenges. We developed an enhanced edition of our Undercover implementation by adopting this measure. A two-week user study with 22 participants was performed to verify its performance against the timing attack. We tested whether the users could respond to hidden challenges more uniformly. The same protocol as in the user studies of the original Undercover scheme was followed for this new user study. All the participants attended the



(a) (b)

Figure 2.17. Average response times and error response rates to different hidden challenges of the Undercover implementation enhanced by shuffling button layouts.

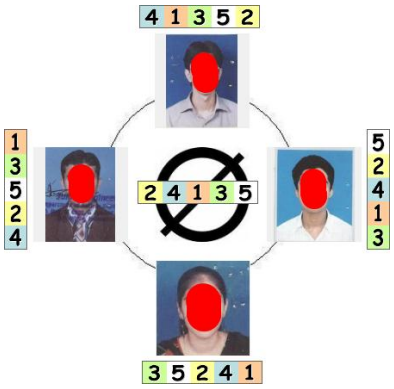


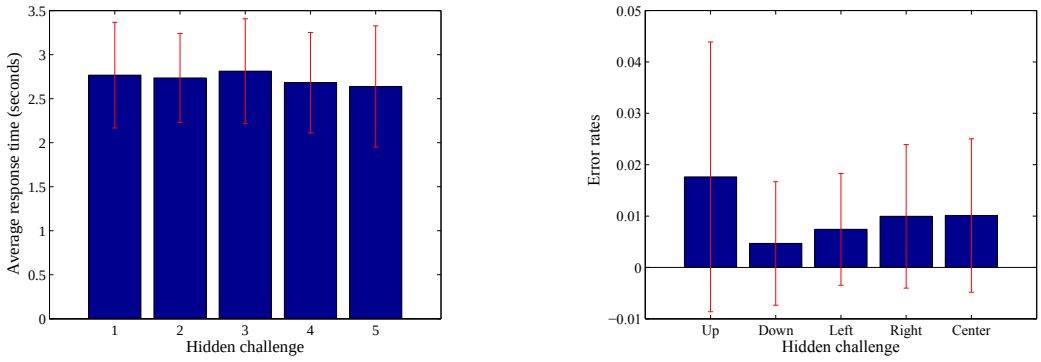
Figure 2.18. The new layout of our enhanced Undercover implementation.

previous user study, except one new user who was recruited for testing this new enhanced design. The average response times become flatter as shown in Figure 2.17(a). However, paired *t*-tests showed that the average response time to "Up" hidden challenges is still significantly shorter than the average response time to hidden challenges "Left", "Right" and "Center" (although with much smaller *p*-values).

After the discussion with some participants, we noticed a possible explanation to the still shorter response time to "Up" hidden challenges. By observing Figure 2.4, we can see that the button layout corresponding to "Up" hidden challenges is the closest to the public challenge. Some participants recalled that they had spent more time in locating other button layouts and verifying them.

To further remove the new kind of nonuniformity in human behavior, we realized that it is important to re-arrange the user interface so that the visual distance between each button layout and the public challenge is equal. This led to the definition of the second guideline that must be followed:

2. *The visual scan path between the public challenge and the response should be made*



(a) (b)

Figure 2.19. Average response times and error response rates to different hidden challenges, of the Undercover implementation enhanced with the new interface in Figure 2.18.

approximately equal.

In the case of Undercover we should equalize the distance between the pass-picture and the button layout used by the user to make the public response. This led to a new design of the interface of the Undercover as shown in Figure 2.18. Now we distribute the four pictures in each public challenge uniformly on a circle, and the "no pass-picture" icon at the center of the circle. The five button layouts are located in the same way as the five pictures. To further simplify the user interface, we also changed the hidden responses to "1", "2", "3", "4" and "5" and the user is asked to: 1) find the hidden response in the button layout near to the pass-picture or the "no pass-picture" icon; 2) press the button at the same location as the hidden response to make the public response. Aforementioned changes make the interface tighter and the user's task simpler, so we would expect the usability of the system could also be improved. As we can see, by following the above mentioned principles to counter the timing attacks on Undercover-based authentication methods, the resulting user-interface will have a symmetrical shape. This way, a symmetrical shape enables an equally difficult visual distance between the challenge and the response and makes the challenges equally difficult for human users to handle.

In the design process of the new enhanced Undercover implementation, we noticed a new kind of nonuniformity that may lead to a new timing attack: if all 5 pass-pictures have appeared in the first five or six public challenges, the user will know that all the remaining (one or two) public challenge(s) will contain no pass-picture, so he/she might be able to respond faster than in the default (usual) case. To avoid this problem, we changed the design so that the last public challenge always contains one pass-picture. This measure has a side effect on the success rate of random guess, which is increased from $1/\left(C_7^2 \cdot 4^5\right) = 1/21504$ to $1/\left(C_6^2 \cdot 4^5\right) = 1/15360$, around 1.4 times larger. Since 1/15360 is still smaller than 10^{-4}, the side effect is acceptable.

A one-week user study with 19 users was then performed to check if this new enhancement works effectively. All 19 users are previous users who had participated in earlier user studies. Unlike our previous user studies, each user was asked to login five to ten times per day so that we can collect enough data for analysis. Figure 2.19 shows the results obtained from the real login data. Now the paired t-test fails to reject the null hypothesis that the response time to hidden challenge "1" has the same mean value as the response time to other hidden challenges, thus leading us to believe that the response times to different hidden challenges are not significantly different. Simulated attacks on the enhanced Undercover implementation showed that none of the user passwords was broken. The success rate of breaking pass-pictures is always below 50%. In addition, as we expected, the average login times and the login error rates are both improved, compared to the original Undercover design: the average login time is reduced to less than 19 seconds after 20 logins and the error rate over all 19 users is just around 6%. We believe that the average login time can be reduced below 10 seconds as user becomes more familiar with the system and her password. While this is still significantly longer than the average time of entering a 4-digit PIN, it is likely that we have to pay some additional costs for getting the additional security against passive observers. It remains a question whether even a better design can be made to further reduce the average login time.

2.6 Enhancing Mod10

In Section 2.4, we showed that Mod10 as a cognitive authentication method is not secure against side channel timing attacks. Although in the previous section some guidelines were proposed to enhance the security (and even the usability) of interface-based methods, as such they cannot be applied to Mod10 method. This is because Mod10 is not an interface-based authentication method where uniform behavior is made by rearranging the user-interface. In fact, the non-uniform behavior in Mod10 method comes from time-dependent human cognitive capabilities to execute mental operations (e.g. mathematical addition) and the fact that users enter the response immediately after they calculate the response. Since it is generally difficult to rearrange interfaceless methods such that the challenges are equally difficult for human users, we will show that the only way to increase the security of interfaceless based methods is to perturb the usability (e.g. increase the overall login time). In the following, we will introduce to a guideline that may be followed not only to enhance the security of Mod10 method, but also the security of all other cognitive authentication methods. We will also discuss the implications of these enhancements on the cognitive workload and the usability.

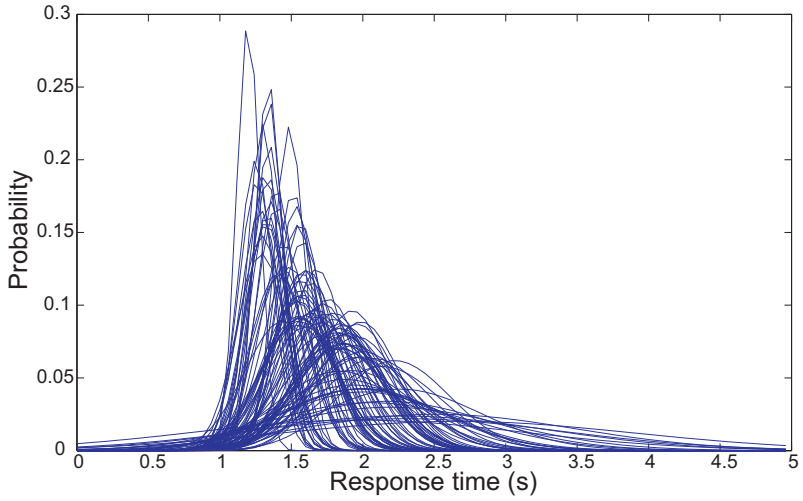


Figure 2.20. Estimated Gaussian distributions of all response times for all PIN digit and response digit pairs collected from the tested users.

2.6.1 Entering a Response After a Predefined Delay

One way to enhance the security of Mod10 method against the timing attack is to prevent users to enter the response immediately after she calculates it. In this way, the attacker will not be able to extract sufficient information about the PIN digit from the recorded response time. In the case of Mod10 method, the user could, for example, be presented with an on-screen keyboard only after some predefined delay (after she received the challenge). Please note that such a delay necessarily increases the overall authentication time; thus higher security comes at the cost of the decreased usability. To characterize a required delay, let us assume that response and PIN digit pairs have a Gaussian probability distribution of response times (as shown in Figure 2.20). We can see that the big majority of response times are not higher than 3.5 seconds (almost 90%). Therefore, by delaying the response at least 3.5 seconds (e.g., by hiding the virtual keyboard until 3.5 seconds has passed), we essentially eliminate the side-channel based on users' response times. Please note that in the case of Mod10 method the fixed delay of 3.5 seconds per response digit increases the response time from the initial 10.4 to approximately 15 seconds (4×3.5 + reaction time [sec]). This increase in the login time of over 40% significantly affects the usability of the enhanced Mod10 method. In the following section we study other approaches to increasing the security of Mod10 method against the timing attacks.

2.6.2 Increasing the PIN Size and Related Usability Cost

In the previous solution we increased the security of Mod10 method by significantly increasing the login time, but leaving the PIN size unchanged (i.e., 4 digits). In this section we explore the possibility of reducing this time cost by requiring the user to remember a

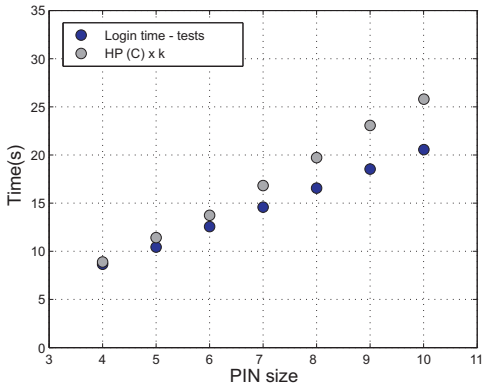


Figure 2.21. Comparison of human cognitive workload $HP(C)$ and average login time collected from the tests as a function of a PIN size.

slightly longer PIN. Please note that the minimum PIN size, denoted with k, has to satisfy the following expression:

$$k = \left\lceil \frac{4 \cdot \log_2 10}{H(D|\mathbf{r}, \mathbf{t_r})} \right\rceil, \tag{2.11}$$

where $H(D|\mathbf{r}, \mathbf{t_r})$ is the conditional entropy of the random PIN digit D given the vectors of observed response digits $\mathbf{r}$ and response times $\mathbf{t_r}$. Essentially, the reduction in entropy due to the time-based side-channel has to be compensated by extending the 4-digit PIN size. For example, with our two classifiers (used in combination) we were able to reduce the entropy by approximately 0.5 bits per PIN digit. So, to counter this particular attack (i.e., to preserve the entropy of a new PIN to be $4\log_2 10 \approx 13.3$ bits), the new PIN size should be (by the above expression) at least $k = 5$ digits long. In order to estimate the associated usability cost (memory and time) of adding an additional digit, we use the computational model proposed in [45]. In this model the process of human-computer authentication is decomposed into atomic cognitive operations in psychology. In this work we consider two types of atomic cognitive operations: free recall and simple cognitive arithmetic. Following [45], in our analysis we measure Cognitive Workload $HP(C)$ and Memory Demand $HP(M)$ (HP being *Human Power*). Cognitive load is is measured in terms of the time required by the user to perform cognitive operations. For Mod10 metod, the associated cognitive load $HP(C)$ is given by the following expression (for entering a single PIN digit):

$$HP(C) = (0.3694 + 0.0383 \cdot \varphi \cdot k) + \alpha_1 + \alpha_0, \tag{2.12}$$

where $(0.3694 + 0.0383 \cdot \varphi \cdot k)$ is the reaction time by the user to recall k PIN digits (her PIN), α_0 and α_1 are the average reaction times for modulo 10 reduction operations and small additions, respectively ($\alpha_0 = 0.738$ and $\alpha_1 = 0.773$), while $\varphi = 1.969$ is the ratio of cued recall compared to single item recognition [45]. When we apply this expression to Mod10 method, we obtain $HP(C) = 2.182$ seconds to recall a single PIN digit, perform one small addition with a challenge and finally reduce the result modulo 10. For comparison, in our

Table 2.2. Usability cost of the enhanced Mod10 method (a strategy based on increasing the PIN size).

		Tests			Model			
k	**Entropy per PIN digit**	**Round time (sec)**	**Login time (sec)**	**Login time + waiting cost (sec)**	**HP(C) per round (sec)**	**HP(C)×k (sec)**	**HP(M)**	**HP = HP(C)×HP(M)**
4	3.32	2.0984	8.3938	10.3907	2.1821	8.7282	4.7170	41.1708
5	2.6575	2.0984	10.4922	13.1548	2.2575	11.2873	5.8962	66.5526
6	2.2147	2.0984	12.5906	15.9190	2.3329	13.9973	7.0755	99.0372
7	1.8968	2.0984	14.6891	18.6831	2.4083	16.8580	8.2547	139.1582
8	1.6610	2.0984	16.7875	21.4472	2.4837	19.8696	9.4340	187.4492
9	1.4616	2.0984	18.8859	24.2113	2.5591	23.0320	10.6132	244.4437
10	1.3288	2.0984	20.9844	26.9754	2.6345	26.3453	11.7925	310.6754

tests with users, the average reaction time per challenge-response round was 2.098 seconds, which shows excellent match with the above human computational model. This is better seen in Figure 2.21 where we compare the cognitive load $HP(C)$ per login session with login times obtained in our tests with users, for different PIN sizes. Therefore, we can conclude that the time cost of an extra digit is around 2.1 seconds, totaling 13.1 seconds. Compared to the solution based on a fixed delay (around 15 seconds for 4 digit PIN), this solution (for the given attacker model) has significantly lower login time. However, in this solution the user has to remember a longer PIN. Therefore, it is interesting to estimate the overall expected human capability requirement (*the usability cost*) for the given authentication system. Following [45], the usability cost of a given authentication method can be estimated using a *human power score* $HP = HP(C) \cdot HP(M)$, where $HP(M)$ represents the memory demand. $HP(M)$ can be calculated as follows:

$$HP(M) = k/\lambda_{op}, \tag{2.13}$$

where λ_{op} is the accuracy rate of the corresponding memory retrieval operation within a fixed memorization time; for Mod10 method $\lambda_{op} = 0.848$ [45]. Table 2.2 shows the comparison of the basic (4 digits) and enhanced ($k > 4$ digits PIN) Mod10 method in terms of the usability cost. As can be seen in the table, adding an extra digit increases the required human capability (decreases the usability) by over 60% (the first two rows in the table, $k = 4$ and $k = 5$). Comparison of the usability cost between this solution and the one introduced in the previous section (based on a fixed delay per digit) is not possible in the used human computational model because the delay in the second solution does not reflect the human reaction time. We plan to investigate this aspect in future; a promising framework for evaluating the usability cost of delayed actions is the one of *prospective memory* from the cognitive psychology [46].

Thus far, we assumed that the maximum amount of information that the attacker can learn by observing the feature vectors $\mathbf{r}$ and $\mathbf{t_r}$ is 0.5 bits per PIN digit. This was based on the results obtained by applying our combining classifier. However, our classifier may not be the optimal one (the one extracting the most information out of the available information).

Since the entropy $H(D|\mathbf{r},\mathbf{t_r})$ affects the PIN size of the enhanced Mod10 method (through the expression 2.11), we need to estimate $H(D|\mathbf{r},\mathbf{t_r})$ more accurately. In this direction, we first estimate the conditional entropy for the given response digit r_i and response time t_r:

$$H(D|R=r_i,T_r=t_r) = -\sum_{j=1}^{10} P(d_j|r_i,t_r)\log_2 P(d_j|r_i,t_r). \tag{2.14}$$

The probability $P(d_j|r_i,t_r)$ can be calculated as follows:

$$P(d_j|r_i,t_r) = \frac{p(t_r|d_j,r_i)P(d_j|r_i)}{p(t_r|r_i)} = \frac{p(t_r|d_j,r_i)P(d_j)}{\sum_{l=1}^{10} p(t_r|d_l,r_i)P(d_l)} = \frac{p(t_r|d_j,r_i)}{\sum_{l=1}^{10} p(t_r|d_l,r_i)}, \tag{2.15}$$

where we use the fact that $P(d_j|r_i) = P(d_j) = 1/10,\ \forall i,j$. We estimate the probability $p(t_r|d_j,r_i)$ using the data collected in our tests, by assuming that they have Gaussian distribution (as shown in Figure 2.22, the first column). In Figure 2.22 we plot the resulting conditional entropies $H(D|R=r_i,t_r)$ for three response digits, namely $r_i = 2,6,8$. The results indicate that the conditional entropy significantly varies with the response times; here we focus on the range between 0.8 and 3.5 seconds (the range of response times corresponding to the majority of users). The entropy varies between 1 and 3.3 bits depending on the observed response time. We can further observe that the entropy also significantly varies for different response digits and a fixed response time. This implies that some response digits reveal more information about the PIN than some others. To estimate the expected conditional entropy $H(D|R,t_r)$ over all possible response digits R we do the following:

$$H(D|R,T_r) = \frac{1}{10}\sum_{i=1}^{10}\int_{t_r\in[0.8,3.5]} p(t_r|R=r_i)\cdot H(D|R=r_i,T_r=t_r)dt_r \tag{2.16}$$

$$= \frac{1}{100}\sum_{i=1}^{10}\sum_{j=1}^{10}\int_{t_r\in[0.8,3.5]} p(t_r|R=r_i,D=d_j)\cdot H(D|R=r_i,T_r=t_r)dt_r. \tag{2.17}$$

Please note that in the estimation of the entropy we restrict response times to the range $[0.8,3.5]$. By evaluating the expression (2.16) numerically, we finally obtain $H(D|R,T_r)\approx 2$ bits. Thus, to be on the safe side concerning the timing attacks, the PIN size should be at least $k = \left\lceil \frac{4\cdot\log_2 10}{2} \right\rceil = 7$ digits. Please note that this implies that the user has to remember and enter 3 extra PIN digits during the login process. Referring to Table 2.2, the overall authentication time increases to approximately 16.8 seconds (counting the cognitive cost $HP(C)$ only). Likewise, the memory demand $HP(M)$ increases by 75% compared to the basic 4 PIN digit. The overall usability cost $HP = HP(C)\times HP(M)$ increases by over 3 times compared to the basic case. In the table we also show the usability cost related to 8,9 and 10-digits long PIN. As we can see from this table, the threat of side-channel timing attacks can only be mitigated by increasing the overall usability cost.

A general conclusion that results from this study can be summarized in the following

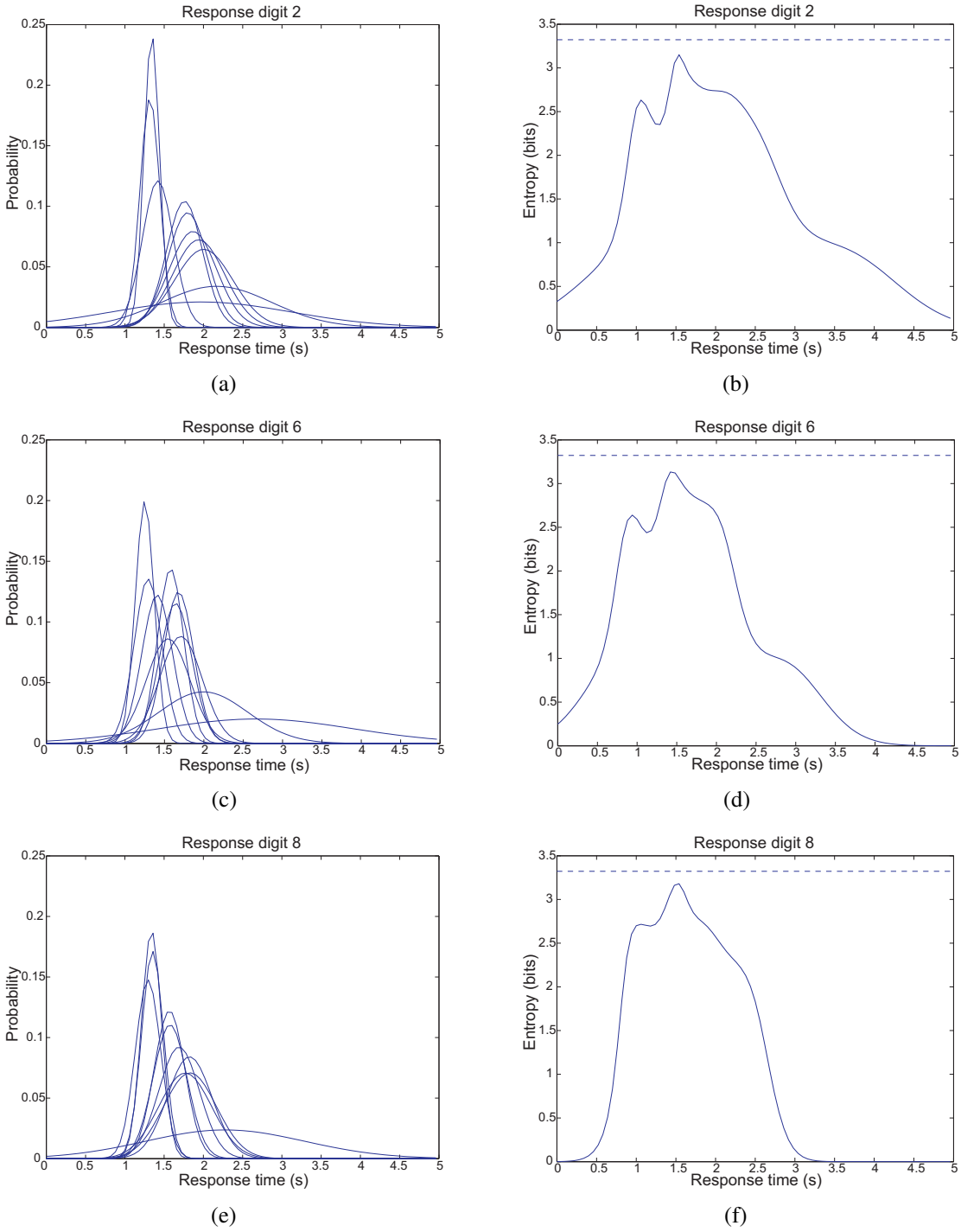


Figure 2.22. Distributions of response times for different (PIN digit, response digit)-pairs $p(t_r|d_j, r_i)$ and the corresponding conditional entropy $H(D|R = r_i, T_r = t_r)$; the distributions are estimated using the 50 fastest response digits over all the tested users.

guideline:

3. *When designing new cognitive authentication methods designers should pay particular attention to potential asymmetry in both the cognitive load and the physical interface of different elements (mental and physical) of their methods.*

To recap, Mod10 method is a good example of a cognitive authentication method where non-uniformity comes from human nature related to mental operations such as recognition, memory retrieval, small additions, etc. The facts that Mod10 lacks a visual interface and that a user cannot perform any better than her cognitive skills allow her, its resistance against the timing attacks cannot be increased in usual ways (e.g., rearranging a visual interface). As we showed in this section, the only way to do this, without changing the nature of the original method, is to sacrifice the usability (e.g., increase the overall login time by a fixed delay and/or increase the PIN itself). The important conclusion from the presented results is that researchers designing and evaluating cognitive authentication schemes should exercise caution when claiming superiority of their proposals in terms of usability with respect to existing proposals. High usability scores are usually highly correlated with short authentication times. However, potential vulnerabilities of those authentication methods to timing attacks could easily render the claimed "short" authentication times not so short in the end.

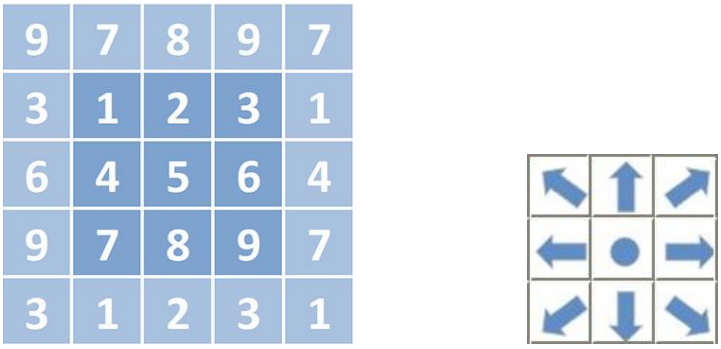
2.7 Enhancing the Security of Mod10 Metod with Table Look-ups

Based on the three principles that need to be followed to eliminate the threat of side-channel timing attacks, in this section we introduce another scheme aimed at improving the security of Mod10 approach [7]. Unlike Mod10, our method does not require users to perform any mathematical or mentally demanding operations. On contrary, it requires the user to perform nothing more than a simple table lookup. We call this new method a Simple Table Lookup (STL) login method.

The STL method can be treated as an extension of the Mod10 method, with the important difference that it eliminates the need to perform mathematical calculations. Recall, in the Mod10 method, in order to enter the ith digit d_i of his/her PIN, the user receives a challenge c_i (one digit long) selected uniformly at random from the set $\{0,\ldots,9\}$ via a protected channel (e.g., earphones). In turn, the user adds the two digits modulo 10 (i.e., $r_i = (c_i + d_i) \bmod 10$) and enters back the outcome r_i of this addition via the public channel (e.g., a numerical pad).

The STL scheme is designed to work in the partially observable model where the adversary can only partially observe the PIN-entry procedure. STL implements the challenge-response paradigm and comprises three major components: (i) *a protected channel* ensuring secrecy and integrity of challenge values, (ii) *a simple lookup table* - a table of digits from 1 to 9 organized in such a way that each digit i is an immediate neighbor to the other 8 digits from the set $\{0,\ldots,9\}$ (Figure 2.23(a)) and (iii) a set of *response buttons* (Figure 2.23(b)).

The STL method proceeds as follows. Let us assume that the user wants to authenticate to a computer using the following PIN: 46548 (please note that the PIN is 5 digit long on purpose). Let us denote the PIN digits as $d_0 = 4$, $d_1 = 6$, $d_2 = 5$, $d_3 = 4$ and $d_4 = 8$. The



(a) (b)

Figure 2.23. STL interface: (a) In the simple lookup table each digit i is an immediate neighbor to the other 8 digits from the set $\{1,2,\ldots,i-1,i+1,\ldots,9\}$; (b) A user enters his/her response via 8 arrow buttons and one center button.

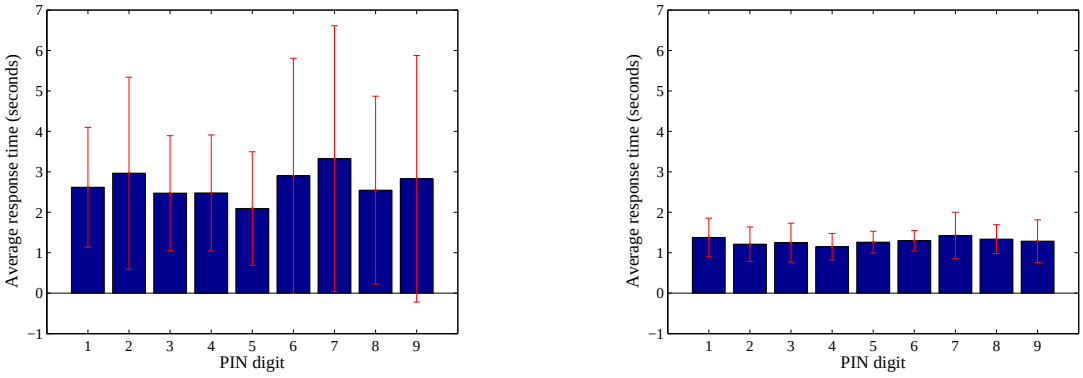
computer will display the STL table on its screen as shown in Figure 2.23(a). These two components of STL are public (observable by an adversary). At time instant t_0, the user will receive a random challenge (one digit long) c_0 selected from $\{1,\ldots,9\}$. Let us assume $c_0 = 9$ in our example. The user will receive the challenge over a protected channel (e.g., over *earphones* plugged into the computer). We assume that the adversary cannot learn this challenge (the partially observable model).

Upon receiving the secret challenge $c_0 = 9$ from the interrogator, the user looks in the darker area of the STL table (Figure 2.23(a)) and locates (visually) the first digit of his/her PIN, $d_0 = 4$. The user then locates (visually) the challenge $c_0 = 9$ in the immediate (one-hop) neighborhood of previously located digit $d_0 = 4$. Note that this is always possible, as the digits in the STL table are arranged in such a way that each digit i (located in the darker area) is an immediate neighbor to all the other digits from the set $\{1,2,\ldots,i-1,i+1,\ldots,9\}$. Finally, the user answers the challenge by clicking a response button (Figure 2.23(b)) that shows the relative position of the challenge c_0 with respect to the corresponding PIN digit p_0. In our example, the user clicks the "south-west" arrow, that is, he/she responds with $r_0 = \swarrow$. It is easily seen that the response r_0 unambiguously links the challenge with the corresponding PIN digit. Note that the adversary can observe the user's response r_0.

At this stage the first PIN digit has been entered and the whole procedure repeats for the remaining PIN digits. At time instant $t_1 > t_0$ the user receives the challenge $c_1 = 6$ to enter the PIN digit $p_1 = 6$. Therefore, he/she responds by clicking the center button (Figure 2.23(a)), that is, $r_1 = \bigcirc$. The whole PIN-entry procedure is summarized in the following table:

time	t_0	t_1	t_2	t_3	t_4
PIN	4	6	5	4	8
challenge values	9	6	2	1	6
user's response	$\swarrow$	$\bigcirc$	$\Uparrow$	$\Uparrow$	$\nearrow$

Note that the STL method does not require any numerical computation on the part of the



(a) (b)

Figure 2.24. Average response times for different digits of the STL implementation in (a) shorterm and (b) longterm study.

human user. Moreover, the number of challenge-response rounds equals to the size of the PIN. It is these two features that make the STL method usable.

The size of the PIN space. The digits in the STL table are arranged in a special way in order to ensure that each digit i is an immediate neighbor to all the other digits from the set $\{1,2,\ldots,i-1,i+1,\ldots,9\}$. The price that we have to pay to accomplish this is the reduced PIN entropy. Indeed, in our solution every PIN digit can take one out of 9 values compared with one out of 10 in Mod10 and classical methods. In order to compensate for this loss, we suggest using somewhat longer PINs. For example, in our experiments each user has been given a 5 digits long PIN compared to 4 in classical solutions; note that $9^5 > 10^4$. As we report in Section 2.7.1, this increase did not significantly affect the usability of our scheme.

From the above description of STL method we can see that method follows the first two principles for the design of a secure authentication method. The visual distance between the visual challenge (located digit d_i) and the response (r_i) is equally difficult (in the immediate one-hop neighborhood). From the design of STL table (Figure 2.15) we can see that users should be able to handle equally the information from public challenges.

A one-day study with 20 users was performed to check the security of STL method. A more detailed usability analysis of STL method can be found in Section 2.7.1 where we provide a comparative usability evaluation of STL and Mod10 methods, while here we provide the security analysis against the timing attack. All users were university staff members in departments of electronic engineering and computer science. All the participants were volunteers who were asked to help our research, and none of them was economically compensated/motivated, so we believe that our data is not biased towards positive results of our proposed attacks. During the authentication phase the users were asked to successfully login at least 30 times per method.

We can see in Figure 2.24(a) that the average response times for digit "5" are shorter.

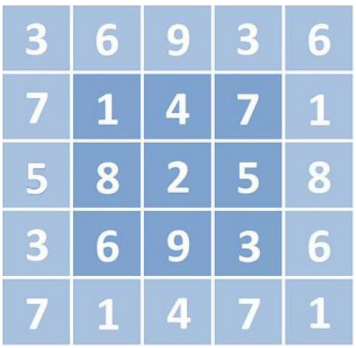


Figure 2.25. Interface of our enhanced STL implementation.

Moreover, paired t-tests showed that the average response time for digit "5" is significantly shorter than the average response time for other digits. After the discussion with users, we noticed a possible explanation of the faster response time for digit "5". Observing the STL table in Figure 2.23(a), we can see that the order of secret challenges in one-hop neighborhood around the digit "5" is exactly the order on the numbers on a regular keypad (e.g. ATM), and that users actually do not even have to look at the STL table to retrieve the response. In comparison, the responses around other eight digits are possible editions of the original layout. Therefore, the non-uniformity in response times comes from the fact the users are not familiar with the layout of response digits within the one-hop neighborhood, as they are with the layout of response digits in the case of digit "5".

We wanted to see the user reaction time once they become more familiar with the method and the STL table. By following aforementioned approach we conducted a 30-day user study with 15 new participants (students) where each user had to login successfully at least once a day. If we take a look at Figure 2.24(b) that shows the average response times of last 10 successful logins (from the last 10 days), we can see that once users become more familiar to the STL table they respond to challenges more uniformly and faster. As a result, paired t-tests now fails to reject the null hypothesis that the response time to digit "5" has the same mean as the response time to other digits, thus leading us to believe that the response times to different hidden challenges are not significantly different. However, this still does not prevent the attacker to record the response times during the "training" phase (e.g., first 20 logins) and learn something about the user's secret PIN. To further eliminate non-uniformity in response times, it is necessary to modify the STL table in such a way that the layout of responses is equally difficult for every PIN digit (first principle). A possible solution could be to randomly shuffle the digits as shown in Figure 2.25. Please note that we still keep all the challenges c_i in the immediate (one-hop) neighborhood of previously located digit d_i (in the dark area). Unfortunately, due to the lack of time we did not test the security of the enhanced STL method. In future work we plan to test the security of the enhanced method against the timing attack.

Another Timing Attack: Correlation Between Identical PIN Digits. In Figure 2.26,

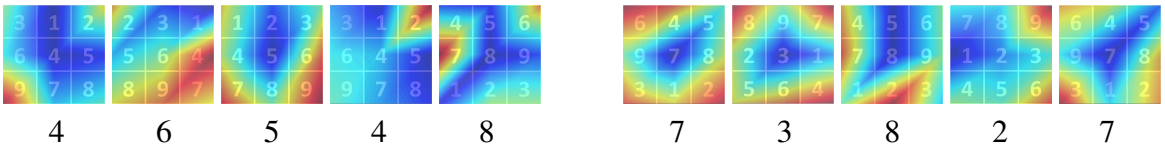


Figure 2.26. Patterns showing different response times (Δt) by two users with PINs: 46548 and 73827.

we plot the response time Δt for two STL users from our evaluation study (Section 2.7.1). The patterns in Figure 2.26 are generated as follows. For each user we recorded 30 successful logins and for each successful login we calculated the response time Δt taken for entering a given PIN digit. Recall that for a given PIN digit, the response time is a function of a random challenge (Figure 2.26). Since there are only 9 different challenge values and we recorded 30 login sessions, each challenge value has been generated approximately 3 times on average for the fixed PIN digit. Next, we average these response times for the fixed PIN digit and each different challenge value. Note that the attacker can do the same, since the same PIN digit and the same challenge value always imply the same public response by the user. These 9 average response times are stored in a 3×3 matrix to which we applied the MATLAB functions `contour()` and `shading()` [47] to finally obtain the patterns as shown in Figure 2.26. Darker areas symbolize shorter response times.

By analyzing the patterns in Figure 2.26 an adversary can extract significant amount of information about the secret PINs. For example, for the first user, the attacker can observe that patterns corresponding to the first and the fourth PIN digit are highly correlated. Similarly, for the second user, there is a very high correlation between the first and the last PIN digit. Based on this information, the attacker can conclude that the respective PIN digits are the same, therefore reducing the security factor from 9^5 to 9^4 (i.e., approximately 89%!).

2.7.1 STL vs Mod10 Usability Evaluation Study

In previous sections we tested the security of STL method against side-channel timing attack. Now, we compare in detail the usability of the Mod10 method and its enhancement - the STL method. We carried out experiments in order to study different usability aspects of the Mod10 and STL login methods. Each test took 30 minutes per method. A total number of 20 participants took part in the usability study[1].

Implementation and Test Procedure

We implemented the STL and Mod10 methods as a web application, where each participant in the experiment was assigned an unique pair of login credentials. The user interface for

[1] It is well-known that a usability study performed by 20 participants captures over 98% of usability related problems [48].

Table 2.3. Summary of the users' demographics.

	Age			**Using PC (hours/week)**				**Using web (hours/week)**			
	18-25	**26-40**	**>40**	**$\geq$ 30**	**15-30**	**6-15**	**$\leq$5**	**$\geq$ 30**	**15-30**	**6-15**	**$\leq$ 5**
STL vs Mod10	18	2	0	11	6	2	1	7	7	4	2

the STL is shown in Figure 2.23(a), while for the Mod10 the participants were shown a blank page. For each participant, the same test statistics (overall login time, error rates) were collected and stored in a central database for later processing.

For all methods, the users receive a random challenge over their headphones. The users themselves initiate the transfer by pressing a combination of Ctrl+Alt+C keys on a keyboard. Each time a participant presses this combination and/or enters a response via the keyboard, the system (participant's browser) logs the current time which is then transmitted to and stored by a central server for later processing.

Prior to the experiment, in STL vs Mod10 evaluation the participants were asked to memorize a five-digit long PIN for STL, and a four digit PIN for Mod10. Please recall, we used a 5 digit long PIN in STL because in this method a PIN is selected from the set of 9 values, compared with classical methods where PIN can take one of 10 values. The usability evaluation per each of the PIN entry methods consisted of two phases. A *training phase* and an *authentication phase*. The training phase served the purpose of teaching the participants how to use the respective methods. This phase consisted of five successful logins for each method. The authentication phase served as the actual test authentication methods (STL with Mod10 method).

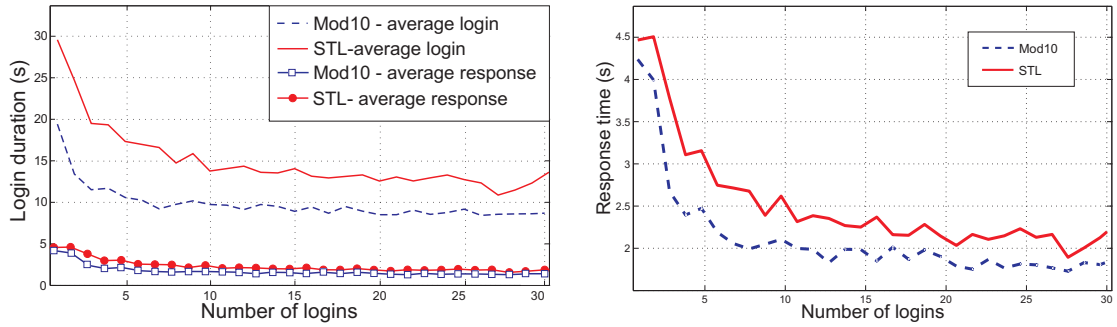
At the end of each usability test for each login method, the users completed a post-test questionnaire. The System Usability Scale (SUS) [49] test was used to numerically express the usability of each method. Within the post-test questionnaire, the users were asked for their opinion regarding the usability and perceived security for each tested method.

A total number of 20 testers (13 males, 7 females) participated in the study. All participants were in their early twenties. Table 2.3 summarizes user's age, the distribution of the time that the participants spent using a computer and internet on a weekly basis.

In the usability test, each participant went through the same set of experiments, as described above. During the authentication phase the users were asked to successfully login at least 30 times per method; each test lasted around 30 minutes (per method). The participants were taking a break for about half an hour between the tests. Next, we presents results from the study.

Login Time

In Figure 2.27(a), we plot the average curves for the login times taken by the 20 participants over 30 successful logins, for both STL and Mod-10 methods. The results in Figure 2.27(a)



(a) (b)

Figure 2.27. (a) The PIN-entry time: average login times from the experiment with 20 STL and Mod10 users. (b) The average user's response time per PIN digit for STL and Modulo-10 method.

reveal two important facts about the STL and Mod10 methods. They both have a very steep learning curve, but in a positive sense. Indeed, already after the first few successful login trails, the login time decreases quickly. The overall login times are reasonably short; only 12.5 and 9.5 seconds and with standard deviation of 7.41 and 2.54 seconds on average for STL and Mod10 methods, respectively. Higher login time with STL can be explained through the size of the PIN (5 digit PIN compared to 4 digit PIN in the Mod10 method) and the fact that the users were not fully accustomed with the table lookup operations. Later in the section, we show that shorter login time with Mod10 comes at the cost of increased error rate.

Next we study the average *user response time*, meaning the time it takes for a user to enter a single PIN digit. As the user response time we take the period from the time instant at which the user receives the challenge value to the time instant at which the user enters the respective response. The user response time does not include the time between the previous response and the subsequent challenge. The average user response time for both STL and Mod10 are given in Figure 2.27(b). As we can see from the figure, towards the end of the testing session, the average user response time for STL is under 2.25 seconds with standard deviation of 1.17 seconds, while for Mod10 the time does not exceed 1.8 seconds with standard deviation of 0.54 seconds. Somewhat longer time to enter a single digit with STL is due to the fact that the users were not fully accustomed with the table lookup operations at the end of the tests.

Error Rates

Figure 2.28 shows average PIN-entry error rates for STL (Figure 2.28(a)) and Mod10 (Figure 2.28(b)) methods over the period of 30 consecutive successful logins. The error rates are shown for 3 equal subsequent periods. Each period includes 10 successful logins. In the first period (for the first 10 successful logins), the error rates are approximately the same for both methods. For the two subsequent periods, Mod10 has a higher error rate than STL, in spite

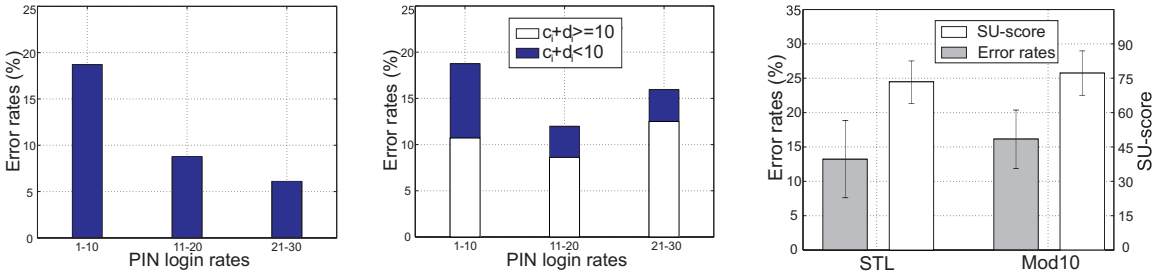


Figure 2.28. The average PIN-entry error rates (%) for 20 users using (a) STL and (b) Mod10 method divided into three consecutive periods. Each period counts 10 successful logins. (c) The average PIN-entry error rate (%) and SU-score for 20 users for 30 STL and Mod10 successful logins; 90% confidence intervals are used.

of the shorter PIN to be entered (4 digits vs. 5 with STL). This difference is better seen in Figure 2.28(c), where we show the average error rate after 30 successful logins with 90% confidence intervals. The difference in the error rates is due to the fact that STL requires only a simple table lookup operations, while Mod10 involves mathematical computations. In addition, for Mod10 we can see the increase in the error rate in the third period (please refer to Figure 2.28(b)). This effect can be explained through the testers' fatigue, caused by a high number of subsequent mathematical calculations required from the participants within a short time span. Similar effect is not seen in the case of STL.

It is very interesting to observe from Figure 2.28(b) that the major source of errors with the Mod10 method are cases in which the sum of the challenge and the respective PIN digit exceeds 10. These type of errors account for more than 70% of all errors. Indeed, this type of non-uniform behavior inspired us to mount a side-channel timing attack on Mod10 method previously introduced in Section 2.4.

Usability

Figure 2.28(c) shows the SUS scores (shown with 90% confidence intervals) provided by test participants for STL and Mod10 methods. The average SU-score for STL and Mod10 is 73 and 78 (out of 100). Thus, it seems that Mod10 is rated by the participants as slightly more usable than STL (although we did not find any statistical difference, as discussed below). In spite of higher error rates, the participants perhaps evaluated Mod10 as slightly more usable because of the shorter login time (9.5 seconds vs 12.5 seconds). Finally, Table 2.4 summarizes the users' answers on the following two questions: "Do you find the methods usable?" and "Do you feel secure while using the methods?" Generally, the results show that majority of participants considered both methods easy-to-use as well as secure.

Table 2.4. Summary of responses from users about how acceptable and secure they found using the STL vs Mod10 methods (5-strongly agree, 1-strongly disagree).

	Using					**Feel secure**				
	5	**4**	**3**	**2**	**1**	**5**	**4**	**3**	**2**	**1**
STL vs	10	4	6	0	0	11	5	4	0	0
Mod10	9	4	7	0	0	9	9	0	0	0

Table 2.5. Cross-Correlation of different measures for STL and Mod10 methods, respectively.

	STL			**Mod10**	
	Average Login Time	**SUS**		**Average Login Time**	**SUS**
SUS	0.0586	-	**SUS**	-0.11715	-
Error Rate	-0.1385	0.0658	**Error Rate**	-0.06847	-0.05023

Within Subject Analysis

Paired t-tests revealed that subjects achieve significantly higher error rates (p=0.0892) and significantly faster login time (p=0.0023) using the Mod10 method as compared to the STL method. However, there was no significant effect on SU-scores while comparing these two methods. The means of the error rates were 16.50% and 11.60% and of the login times were 8.72515 and 13.6327 seconds, respectively, corresponding to the two tested methods.

Although the users achieved faster times with Mod10 method, they did not consider this method to be significantly more usable (SU-score) than the STL method (p=0.1068).

Possible Correlations

Next, we investigated the existence of possible correlations among the login times, error rates and usability scores for each method (STL and Mod10). Table 2.5 shows the correlation coefficient for these two methods. As shown, none of the measures is sufficiently correlated with others to be considered as strong. However, it should be noted, for STL that there is weak negative correlation between error rates and login times, meaning that longer login times imply a higher likelihood of errors.

2.8 Intersection Attack

In this chapter, we also report an intersection attack on Undercover [3] that can also be generalized to the alternative designs in [4]. The intersection attacks are based on flaws in the Undercover design. The experimental analysis show that the attacks can recover the

password or part of it with considerably high probability with less than 10 observed login sessions.

Intersection attack is not new and has been reported in previous research on other human authentication systems especially graphical passwords [50]. The basic idea behind intersection attack is to fuse the information obtained in multiple observed login sessions to reduce the space of password space (i.e., the password entropy). This subsection presents intersection attacks on the original and alternative designs of Undercover in [4, 3].

2.8.1 Breaking the original Undercover design with randomized public challenges

In [3], the system is designed so that each pass-picture and decoy picture is shown once and only once in a single authentication process. Unfortunately, showing each picture only once is not a sufficient condition to maintain the security. In fact, how the public challenges are generated also matters. In this sub-subsection, we show that the password can be exposed with $O(10)$ observed login sessions if randomized public challenges are used.

In [3] it was not made clear how public challenges should be generated. Our communications with the authors of [3] revealed that they implemented their prototype system with fixed public challenges, so their prototype does not suffer from the security problem discussed in this sub-subsection. However, since this issue was not discussed in [3], a reader might assume that randomizing public challenges is still fine or even beneficial because randomness often helps enhance the security of a system. Therefore, the intersection attack in this sub-subsection shows how important such small design details are for a secure system.

Each public challenge exposes a significant amount of information about the password due to the following fact: each public challenge (i.e., a set of four pictures) contains at most one (i.e., either none or one) pass-picture. This means that a candidate password can be excluded if two or more pass-pictures in this candidate password appear in a public challenge. In other words, observation of one public challenge can lead to a reduction of the password space. Therefore, as the number of observed public challenges increases, the password space will become smaller and smaller and finally the real password will be revealed after a number of login sessions are observed. For n given observed public challenges, the reduced password space can be mathematically calculated as the intersection of the reduced password spaces corresponding to the n public challenges; hence we call this attack an "intersection attack". The real attack is performed in a simpler way:

- *Step 1*: Set $\mathbf{P}$ to be the space of all possible passwords.
- *Step 2*: For each observed public challenge, reduce the space of candidate passwords $\mathbf{P}$ by checking each password in $\mathbf{P}$ and removing invalid ones.

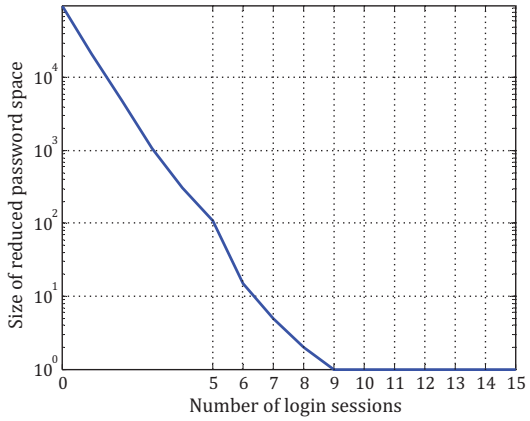


Figure 2.29. The size of reduced password space in an intersection attack on the original Undercover design.

- *Step 3*: Repeat *Step 2* until all observed challenges are processed or the size of **P** becomes 1.

To verify the real performance of the above intersection attack, we did MATLAB simulations on the original Undercover system with 15 randomly generated login sessions (i.e., 105 public challenges). The experimental results showed that the actual number of observed login sessions to uniquely reveal the password is seven to ten in most cases. A typical simulation result is shown in Figure 2.29. We performed the intersection attack on real login data collected in our user studies, and the passwords of all 28 users were successfully broken. The number of required login sessions ranges from eight to eleven, and the median number is nine.

2.8.2 Breaking alternative Undercover designs

For the two alternative designs proposed in [4], the same intersection attack still works but in a slightly different way. Now no public challenge is available, but the user's response becomes the source of information leakage. This is due to a flaw in the alternative designs: for different PIN digits and hidden digits, the user needs to press different sequences of arrow buttons to make a correct response. As a result, the buttons presses and their order can leak information of the PIN and hidden digits.

This problem can be best explained by an example. Assume the PIN digit is 2 and the hidden digit is 6. To make a correct response, the user needs to press Button "Left" (←) and Button "Down" (↓) (the order does not matter). Obviously, pressing Button "Down" leaks the information that the PIN digit is in the first row. Similarly, pressing Button "Left" reveals that the PIN digit must not be 0. As a whole, the number of possible PIN digits is reduced from ten to only four (1, 2, 3 or 4).

Table 2.6 shows a list of different button press patterns that can leak information about

Table 2.6. Information leaked from different button presses.

Button press pattern	Possible PIN digits	Possible hidden digits
↓	0, 1, 2, 3, 4	5, 6, 7, 8, 9
↑	5, 6, 7, 8, 9	0, 1, 2, 3, 4
←	1, 2, 3, 4, 6, 7, 8, 9	0, 1, 2, 3, 5, 6, 7, 8
→	0, 1, 2, 3, 5, 6, 7, 8	1, 2, 3, 4, 6, 7, 8, 9
←←	2, 3, 4, 7, 8, 9	0, 1, 2, 5, 6, 7
→→	0, 1, 2, 5, 6, 7	2, 3, 4, 7, 8, 9
←←←	3, 4, 8, 9	0, 1, 5, 6
→→→	0, 1, 5, 6	3, 4, 8, 9
←←←←	4, 9	0, 5
→→→→	0, 5	4, 9

PIN digits, where the occurrence probability of each case assuming that each PIN digit and each hidden digit distribute uniformly in $\{0,\ldots,9\}$. Here, we ignore button presses that cancel each other, e.g. one "Left" followed by one "Right" or one "Up" followed by one "Down". From Table 2.6, we can see that a combination of some button press patterns can lead to a unique determination of the PIN digit.

We did a large number of MATLAB simulations to test the real performance of the intersection attack. For a PIN "1236", 1000 random attacks showed that the median number of login sessions needed to uniquely reveal the whole PIN is eleven. For a PIN "0459", the median number is nine. Since we did not implement this alternative design of Undercover, the attack was not validated by real login data from a user study. However, the attack does not depend on human behavior at all, so a re-validation via a user study is not really necessary.

2.9 Related Work

In the challenge-response protocol that we have described above, the key is to find a good mapping f so that the computation $\mathbf{R}=f(\mathbf{C},\mathbf{S})$ can be easily handled by an average user, while at the same time it can maintain the expected security level. If a hardware device is available to assist the user, it is not difficult to choose a strong trapdoor one-way function f, thus leading to a cryptographically strong system. Unfortunately, to protect the device from unauthorized access, a password/PIN is still required, which is again vulnerable to observation attacks. If the hardware device is a general-purpose one, like a mobile phone, then mobile malware could be another potential threat [18].

If auxiliary hardware devices cannot be used, the mapping f has to be sufficiently simple for users to mentally calculate the correct responses. While the user has his/her own brain as the only computational resource, the attacker can access a supercomputer or even a large number of distributed computing resources (e.g., a botnet). Furthermore, to make a human authentication system usable in reality, the average login time and the error rate should be small. In contrast, the attacker can wait for a long time to break a victim's secret. Intuitively argued, it is non-trivial to find a mapping f that makes the constructed human authentication system both sufficiently secure and highly usable. Since the 1990s, there have been a number

of attempts in this field, but they are either insecure or not usable in terms of average login time.

To the best of our knowledge, the first solution against observation attack was proposed by Matsumoto and Imai in 1991 [51]. The solution tries to hide the user's secret in the response by using a question alphabet and a randomized answer alphabet. Unfortunately, a few years later Wang et al. pointed out [52] that Matsumoto-Imai scheme is not sufficiently secure if the same challenge can be replayed several times by an active adversary. In addition, to achieve a high level of security, Matsumoto-Imai scheme has to use large question and answer alphabets, thus compromising usability [53]. Wang et al. also proposed an improved scheme to enhance the security, but the usability is much worse.

Matsumoto later proposed several other solutions based on inner products of secret and public vectors [54]. As pointed out in [55, 53], these solutions cannot resist multiple observations because the secret vector can be solved from $O(N)$ observations, where N is the size of the secret vector.

Li and Teng proposed a new solution based on lexical shifting and matching in [56]. Although no cryptanalysis has been reported so far, its usability is not good enough since the user needs to remember a long 3-tuple secret.

Hopper and Blum proposed two solutions based on hard mathematical problems in [57]. The main problem with these solutions is again the usability: the password has to be long enough to ensure security, which makes usability relatively low. According to the user study reported in [57], the average login time of one solution (the less complicated one) is around 160 seconds, which is too long for a practical system. One solution also requires the users to make intentional errors with probability h, which may not be an easy task for them.

Sobrado and Birget proposed several novel graphical password schemes against observation attacks in [58]. One typical scheme called CHC (convex hull click) asks the user to click a random point inside a convex hull formed by three or more secret icons in the password. This scheme was later tested in a user study reported by Wiedenbeck et. al. in [59]. A similar scheme called S3PAS was proposed in [60]. Two attacks on CHC were recently reported in [61]. In addition, the usability shown in [59] is not encouraging: the average login time is longer than 70 seconds. The user study was performed on a small password space of size $C(112,5) \approx 2^{27}$, so the usability would be much worse if the password space had to be enlarged significantly.

In [62], Li and Shum suggested some basic principles of designing challenge-response protocols against observation attack. They also proposed two general protocols called Twins and Foxtail, which are based on making balanced errors and hiding direct responses to attackers, respectively. A Foxtail protocol and a graphical implementation were also reported. No cryptanalysis has been reported, but the usability of the graphical implementation is questionable, since the average login time is considerably long.

Jameel et al. proposed a new image-based solution in [63] and shortly after extended

it for devices with limited display [64]. This solution is based on a hidden rule classifying an image pool into two different sets. One major problem with this design is the conflict between the automation of the classification process and security against automated attack. However, if the classification process has to be done manually by the user, the usability will be low since the image pool needs to be large.

In [65], Weinshall proposed two new solutions based on image recognition capabilities of humans. Golle and Wagner showed that both solutions are insecure against SAT (satisfiability solver) attack [66]. This attack requires only a small number of observations. In addition to the security problem, the usability of Weinshall's solutions is also questionable: the user has to remember more than 30 pictures as the password.

Bai et al. proposed a new observation-resistant human authentication scheme called PAS in [22]. PAS uses different parts of the password for different login sessions and the user's responses are obfuscated by randomized challenge and response tables. In [67], Li et al. showed that part of the password can be revealed with a number of observations, thus leading to a degradation of the PAS scheme to a common OTP (one-timepassword) system but with worse usability.

In [68] Lei et al. proposed a virtual password system against observation attacks. Their system is based on a randomized linear function. However, in [69] Li et al. pointed out that this virtual password system is not secure because an equivalent password can always be derived with only two or a few more observations.

Very recently Asghar et al. proposed a scheme in [61], which is based on a many-to-one nonlinear mapping to hide the direct response. While it is still too early to say if this solution is indeed secure, its usability does not seem to be very encouraging: the average login time was estimated to be 213 seconds, even slower than Hopper-Blum protocol proposed in [57].

Instead of trying to design a solution secure against general observation attacks, some solutions mitigate the security requirements to target only the weakest observation attack: shoulder surfing with a very limited number (say, three) of passive observations. Examples of these solutions include some graphical passwords [24, 50, 70], that offer limited security against observation attacks by exposing only partial information of the password in each login session. An interesting comparative study on simple shoulder surfing performed by human observers on Passfaces (a commercial graphical password scheme [71]) and textual passwords was reported in [14], which reveals that Passfaces with keyboard input is the strongest setting, whereas strong textual password is the weakest one.

While most previous work does not require any hardware device, some other solutions employ special devices so that challenges and/or responses are completely or partially unobservable. Devices of this kind include eye-gazing devices [72], haptic/tactile input devices [23, 26, 27, 4, 29, 30, 20, 21, 3], headphone/earphone[26, 4], mobile phones [26, 28], and so on. The use of eye-gazing devices can obviously make the responses **R** invisible to human observers, but it is still possible to install hidden eye-tracking devices to read the

user's eye movements. Solutions based on other partly/completely unobservable devices have close links to Undercover [3, 4], the observation-resistant solution studied in this paper, and are described in the next section.

It is a known fact that users do not reliably protect their passwords [73]. Previous research has also shown that different kinds of insecure human behavior can compromise the security of some password systems [74, 28, 75]. However, how human behavior affects the security of many ad hoc designs of human authentication systems still remains largely unexplored.

2.10 Summary

In this chapter we report successful side-channel timing attacks on two authentication methods, Undercover and US patent Mod10, which were believed to be secure against observation attacks. The main security weakness in these methods is due to non-uniform human behavior. We also proposed some enhancements to make Undercover more secure against the proposed attacks. User studies were carried out to verify both our proposed attacks and the performance of the suggested enhancements.

Our work reemphasizes that designers of security systems should pay special attention to the security of cognitive authentication systems against timing attacks. More specifically, the attacks proposed in this chapter demonstrate that, if meticulous care is not exercised in measuring how human users will perceive and operate a security system, user behavior can reveal sensitive information that can be used to break the system.

We introduced three guideline principles the designers of future secure authentication mechanisms should follow. The proposed security enhancements of authentication methods show the existence of a strong tradeoff between security and usability indicating that sometimes a secure human-identification system as a consequence needs to perturb the usability (e.g. by increasing the overall login time). The general conclusion is that designers of cognitive authentication schemes should pay attention when claiming superiority of their proposals in terms of usability with respect to existing proposals.

In addition, we report an intersection attack on Undercover which can be generalized to the alternative designs.

3 FORTUNE COOKIES AND SMARTPHONES: WEAKLY UNRELAYABLE CHANNELS TO COUNTER RELAY ATTACKS

3.1 Introduction

Smartphone devices are rapidly gaining momentum in our lives, thanks to their diverse capabilities (gaming, sensing, information availability, positioning). It is only the matter of time when these devices will be turned into fully equipped payment devices that will transfer payments from the existing credit cards to mobile phones. Indeed, thanks to NFC (near-field communication) technology, already used on some credit cards [76], Google has seen the potential of mobile payments and started to push NFC chips on smartphone devices (such as Nexus S [77]). In the absence of devices that have integrated NFC technology, three mobile operators AT&T, Verizon and T-Mobile have recently formed an initiative named ISIS, that recommends manufactures to push NFC on their smartphones in order to increase mobile (proximity) payments [78]. More notably, Spanish bank La Caixa has installed contactless terminals that allow using NFC based smartphone devices [79]. While most existing NFC technologies are RF based, some alternative NFC communication technologies have also emerged in the context of mobile transactions. Naratte's Zoosh is one such technology based on ultrasound which enables users to exchange data between devices using the speakers and microphones [80]. A recent solution from NRC allows users to perform ATM cash withdrawal using their smartphones. The transaction is completed upon scanning a 2D QR code presented on the ATM screen with a smartphone camera [81]. One advantage of this software-based solution is that it operates with existing devices - requiring no hardware changes. Another alternative short range technology is the *intra-body communication* (IBC) that uses the electrical conductivity of human skin to exchange data between devices attached to or in contact with a person [82]. With this technology a user can perform a payment simply by touching an IBC compliant device.

All these NFC[1] technologies look promising as they can move payments from existing credit cards to smartphone devices. Yet all these (and similar) technologies have one problem in common: they are all vulnerable to a tricky form of relay attack - *mafia fraud attack* [83].

[1] We slightly abuse the term NFC.

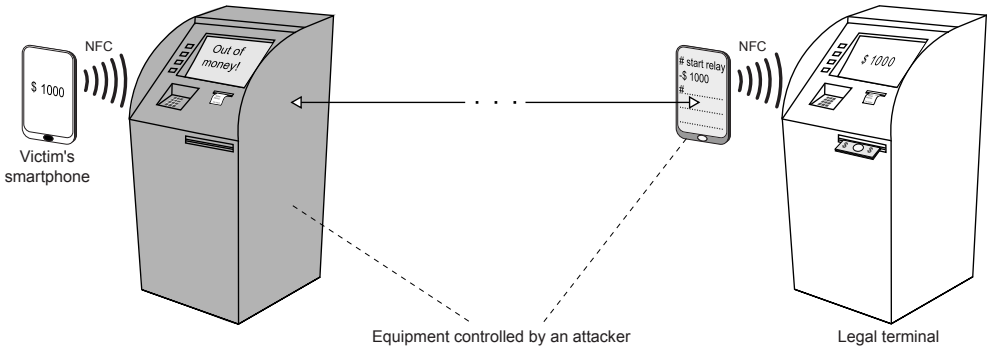


Figure 3.1. The NFC relay attack. A regular customer A wants to pick up money from the ATM. He prepares the transaction on his smartphone and taps the fake ATM (operated by the attacker) to establish NFC communication. At the same time, the attacker, with his smartphone, taps the trusted terminal B. The transaction from smartphone A is wirelessly relayed to B allowing the attacker to pick up the requested money from the trusted terminal B.

In order to live up to the above expectations, these NFC technologies have to address the threats posed by the mafia fraud attack.

Let us briefly review the attack using the scenario shown in Figure 3.1. Customer equipped with a smartphone (prover) approaches a fake cash machine that is under the control of the attacker. S/he prepares the transaction on her/his smartphone device (1000 USD) and taps the ATM operated by the attacker to initiate the NFC-based transaction. At the same time, the attacker, with his smartphone, taps the trusted terminal (verifier) and simply relays the transaction details between the fake cash machine and his/her smartphone and picks up all the money. On the other side, the victim receives the message "Out of money" to avoid any suspicion. Please note that the attacker does not have to know the transaction details (i.e. the secret key shared between the honest smartphone and the cash machine) in order to succeed in this attack - the messages between the victim's smartphone and cash machine are simply relayed in their original form. This is what makes this attack so subtle.

Please note that this scenario is not so far fetched, as similar mobile phone based withdrawal services such as M-PESA exist. This is a SMS-based money transfer service, that allows individuals to deposit, send and withdraw funds using only their cell phone [84]. Therefore, this system is even more vulnerable than the NFC-based solution envisioned in Figure 3.1. M-PESA is a very popular solution reaching more than 40 percent of Kenya's adult population.

The problem of relay attack has long been recognized by the research community. Drimer and Murdoch [85] introduced a practical implementation of a mafia fraud attack on Chip and PIN system. In 2007 Anderson [86] pointed out at the ease of mounting a successful relay attacks using future NFC-enabled mobile phones. Indeed, Francis et al. [87, 88] showed a practical implementation of peer-to-peer NFC relay attack using off-the-shelf equipment, while Roland [89] went one step further with a practical and cheap implementation of relay

attack on Google Wallet - a smartphone based payment service that uses NFC to transmit payment to merchants.

Hence, a challenge is to find a solution against relay attacks in such scenarios. Most existing solutions are based on RF distance bounding protocols that calculate the delay between sending out a challenge bits and receiving back the corresponding response bits between two legitimate parties [90]. However, the proposed protocols are still not practical, as they require modification of the existing hardware. Moreover, these solutions are mainly focused to the specific short-range contactless technology (e.g. RFID). Ideally, the solution should cover a range of proximity technologies (e.g. NFC, Bluetooth, WiFi, IBC - Intrabody Communication). One such attempt was the work of Stajano et al. [1] who suggest augmenting the channels normally used for the transaction with an additional, specific channel that attackers will not be able to relay. However, their solutions are highly impracticable and serve as illustrations of their multichannel-based relay protection.

Inspired by the recent work of Stajano et al. [1] on *unrelayable* channels, in this chapter we show that a secret message printed on a piece of paper and appropriately folded (hence *fortune cookie*) can implement a protection against relay attacks (a *weakly unrelayable channel*). As we will show, weakly unrelayable fortune cookies in combination with omnipresent smartphones provide strong protection against mafia fraud attacks.

Our mechanism is based on a novel (multichannel) protocol, called Forces (**For**tune **C**ooki**es**) that shares some similarities with multichannel protocols proposed by Stajano et al. [1], but with one important difference. Essentially, we reduce the timeframe within which the attacker can relay a message transmitted over the special (paper) channel. This is what makes our solution more practical compared to other solutions outlined in [1].

The proposed solution is suitable for scenarios that involve transaction (paper) receipts such as ATM money withdrawals and payment terminals. Compared to existing solutions for contactless systems, mainly based on RF distance bounding, our solution is independent of the used contactless/wireless technology (e.g., NFC, WiFi, Bluetooth, infrared) or contact-based technology (e.g., intra-body communication). Moreover, we show that it requires minimal or no hardware changes to the existing equipment (particularly on the user's side).

We implemented two instantiations of our solution and performed an extensive user performance study with 54 participants. The study indicates that the proposed solution is both easy to understand and to perform by the end users. Moreover, our study shows that it is possible to eliminate critical errors that rely on the alertness of the honest user. Finally, we prove formally the security of the proposed Forces protocol.

Chapter Outline. In Section 3.2 we give an overview of our solution. In Section 3.3 we present the Forces protocol, while Section 3.4 deals with the attacker and protocol timing model. Security of our weakly unrelayable channel is described in Section 3.5, while in Section 3.6 we state the main result. User performance study is presented in Section 3.8. Related work is provided in Section 3.9, and we conclude in Section 3.10.

3.2 Solution Overview

In the recent work [1] Stajano et al. propose a general paradigm to counter relay attacks, which is based on multichannel protocols (e.g., a radio channel). They suggest to augment the channels normally used for the transaction with the additional special channel (out-of-band) that attackers will not be able to relay. Two honest endpoints can use that special *unrelayable* channel in order to verify that they are talking directly to each other. Stajano et al. [1] argue that the following properties are required to implement unrelayable channel:

- **unclonability**: it must be prohibitively difficult to produce a copy of some physical aspect of that channel,
- **unsimulability**: it must be prohibitively difficult to simulate some physical aspect of that channel,
- **untransportability**: it must be prohibitively difficult to manufacture a data pipe device capable of transporting a detectable physical aspect of that channel from one location to another.

Stajano et al. [1] also present several examples of unrelayable channels that are highly impracticable and serve as illustrations of their multichannel-based relay protection. They conclude their work by stating that more robust and practical implementations of unrelayable channel are required [1]. In this work we propose one such practical implementation of unrelayable channel along with associated authentication protocols. Our unrelayable channel is based on a plain paper.

We relax two properties required to implement unrelayable channel as originally proposed by Stajano et al. in [1]. We require the properties of unclonability and untransportability to hold for only a limited time period Δt. More precisely, we define:

- **weak unclonability**: it must be prohibitively difficult to produce a copy of some physical aspect of the special/urelayable channel within a predefined timeperiod Δt
- **weak untransportability**: it must be prohibitively difficult to manufacture a data pipe device that is capable of transporting a detectable physical aspect of that channel from one location to another within a predefined timeperiod Δt.

We keep the original unsimulability property unchanged. A channel satisfying the above relaxed properties is said to be *weakly unrelayable*.

Next, we briefly explain how a secret message on a piece of paper and appropriately folded can serve as a basis for the implementation of such a channel, as shown in Figure 3.3. In our solution, the message printed on a paper represents a secret challenge in a challenge-response protocol executed between a prover (e.g., a smartphone in Figure 3.1) and a verifier

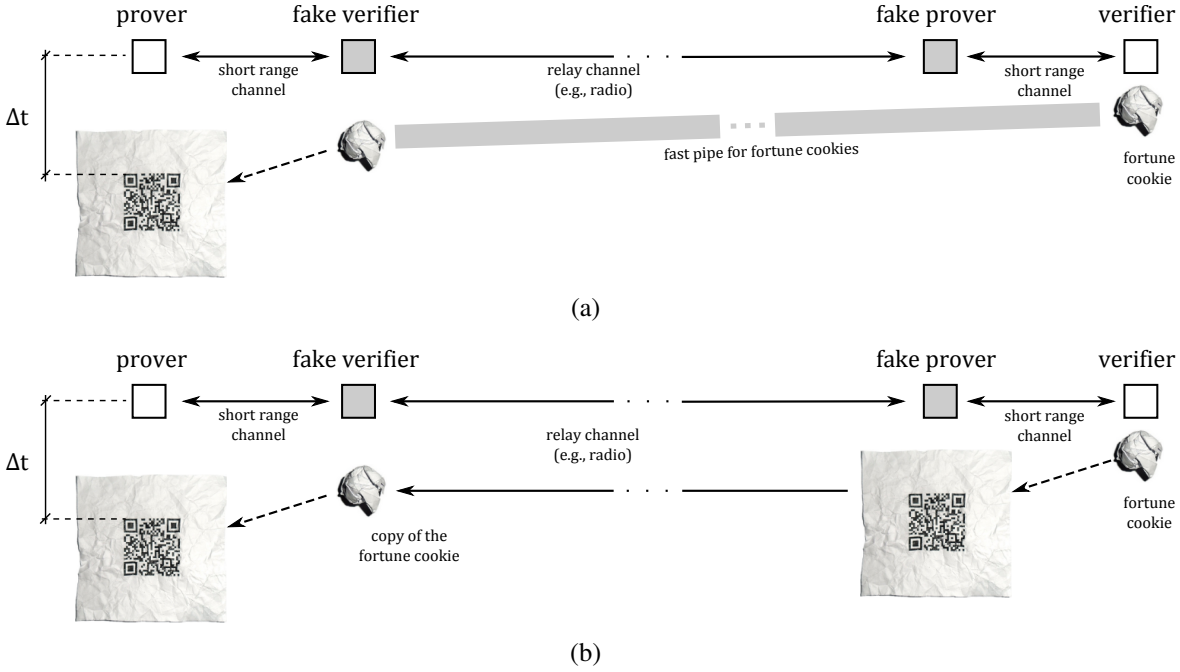


Figure 3.2. Two possible relaying/attacking strategies: the attacker can either (a) physically transfer a secret message printed on a piece of paper between the two honest endpoints, or (b) somehow read the secret nonce from the paper, forward it using a fast relaying channel and finally clone it on the side of the honest receiver.

(e.g., a cash machine in Figure 3.1); we elaborate this protocol in detail later in Section 3.3. In this protocol, the verifier issues the secret challenge to the prover (the user's smartphone). Essentially, without knowing that secret challenge, the prover cannot authenticate successfully with the verifier. To succeed with the relay attack the adversary can either: (i) physically transfer the original paper imprinted with the secret challenge between the two honest endpoints as shown in Figure 3.2(a) or (ii) to somehow read the challenge from the original paper, forward it using a fast channel and finally clone it on the side of the honest receiver (Figure 3.2(b)).

Now, if the delay induced by either of the above mentioned attacker strategies exceeds some predefined and detectable delay Δt, the attack will fail. Our solution is the reminiscent of regular distance-bounding techniques, but with an important difference that it is independent on the underlying communication technology. In this work we argue that a "paper" channel, as described above (Figures 3.2(a) and 3.2(b)), is indeed difficult to relay within a time period of the order of 100 ms (e.g., $\Delta t \approx 100$ ms); such relatively short delays can be reliably detected by standard smartphones. Indeed, transporting or unwrapping a tightly folded/crumpled piece of paper (i.e., relaying it) within such a short period of time (in our case $\Delta t = 70$ ms) requires a great deal of sophistication, as we show in Section 3.5.

Please note that in the practical solution it would be highly impracticable to work with a crumpled paper, such is the one shown in Figure 3.2. Therefore, we propose the usage of

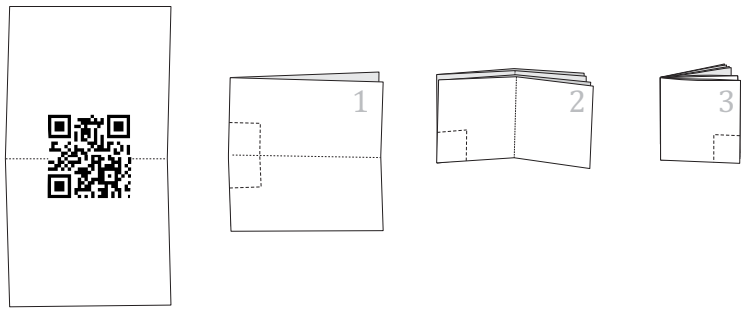


Figure 3.3. A secret message (nonce) printed on a piece of paper and appropriately folded can serve as a basis for the implementation of weakly unrelayable channel - a fortune cookie.

regularly folded paper (e.g., folded 2-3 times) as shown in Figure 3.3. We assess the security of the proposed solution in a formal model in the Appendix.

3.3 Authentication Protocol for Weakly Unrelayable Channels

In this section we provide details of a novel authentication protocol, designed to work with two channels, where one channel is weakly unrelayable (paper channel in our case). As stated before, we propose the implementation of such a channel using a secret message printed on appropriately crumpled paper - a fortune cookie. We call our protocol Forces (**For**tune **C**ooki**es**). The Forces shares some similarities with existing multichannel protocols, as proposed by Stajano et al. [1], but still differs with one important difference: it requires a weaker notion of uncloneability and untransportability. Essentially, in our case, we reduce the timeframe within which the attacker can relay a message transmitted over the special (paper) channel. It is exactly this property that makes our solution more practical compared to highly impractical solutions outlined in [1].

The proposed authentication protocol is shown in Figure 3.4. We use the following scenario in the description of the protocol. A user, equipped with a smartphone, wants to use a service offered by some terminal (e.g., withdraw some money, buy a train ticket). The user authenticates with the terminal using her smartphone. Please note that in the description of the protocol we distinguish two entities: entity A, representing the user and her smartphone, and entity B, the terminal the user interacts with. As shown in Figure 3.4, the Forces protocol evolves in four phases. We assume that the smartphone A and the terminal B share a secret key K.

Phase I: Initialization. The user enters the transaction details into her smartphone (i.e., PIN, desired amount of money) as well as the ID of the terminal B (entering the terminal's ID can be performed via NFC or QR codes). We use dashed arrows to denote message transfer

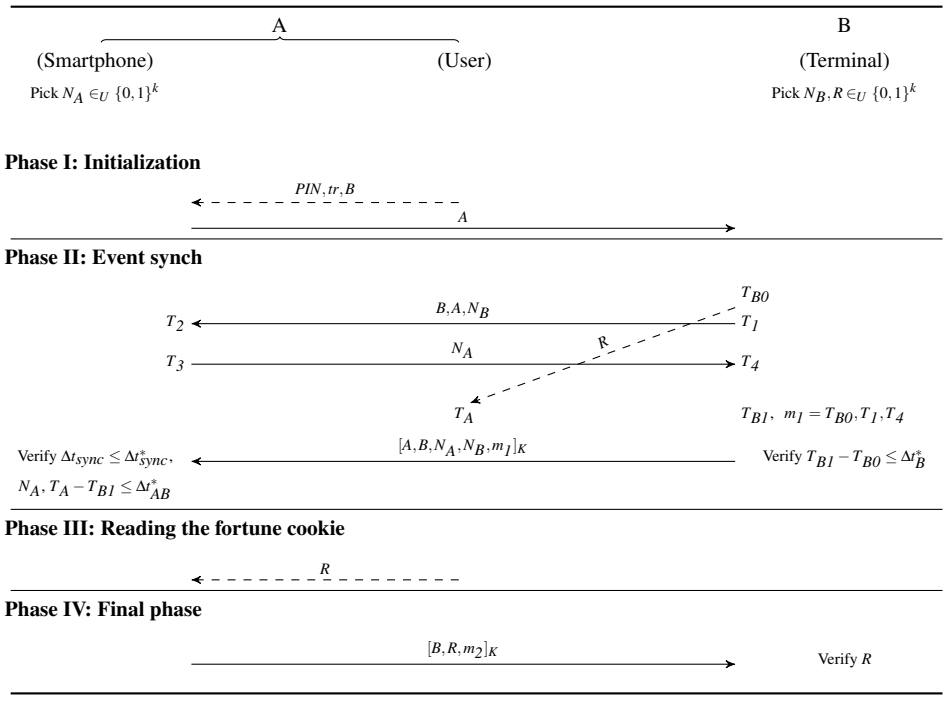


Figure 3.4. Authentication protocol for weakly unrelayable channel based on fortune cookies - Forces. A dashed arrow denotes message transfer between the user and the device, while a communication over a regular channel is denoted with a solid arrow.

between the user and a device. Communication over the regular channel (e.g. WiFi, NFC, Bluetooth) is denoted with solid arrows. Having entered all the transaction details, the user instructs her smartphone (a push of the button) to initiate the Forces protocol. At this stage, the smartphone A transmits the user/smartphone's identity over the regular channel to the terminal B. The initialization phase is finished when the terminal successfully verifies the user identity.

Phase II: "Event" synchronization. With successfully verified user's ID, the terminal B prepares the fortune cookie - i.e., generates a random nonce R, prints it on a piece of paper, appropriately folds the paper (Figure 3.3) and exposes it to the user via appropriate dispenser. At this stage, the user pulls the fortune cookie out (this is marked with the dashed arrow in Figure 3.4) and at the same time, using her smartphone, records the moment when the cookie leaves the ATM dispenser. This is the crucial stage for our protocol, where the user A (her smartphone) and the terminal B synchronize with respect to the moment when the fortune cookie left one *trusted zone* (the terminal) and entered into another trusted zone (the user's hand) - making this the *event synchronization*. During this stage, the terminal B records the time instant (denoted with T_{B0}) when the user begins to draw the fortune cookie and the time instant (denoted with T_{B1}) at which the fortune cookie leaves the terminal (its trusted zone). At the same time the user, using her smartphone, also timestamps the very moment (T_A in Figure 3.4) at which the cookie leaves the terminal; later in Sections 3.3.1 and 3.3.2 we study in detail three approaches by which the user can accomplish this in a secure and usable way.

As recorded times T_A and T_B are to be compared against each other, the smartphone and the terminal have to synchronize their clocks. For our purposes, a rather weak synchronization is sufficient (e.g., the corresponding clock should be within 10 ms). To accomplish this, we use the time synchronization protocol similar to [91].

The coarse clock synchronization is initiated by the terminal B at time instant T_1 (the moment at which the user begins to draw the fortune cookie). As shown in Figure 3.4, the terminal B transmits the random nonce N_B and its identity; B also records this moment T_1. The smartphone A recieves this message at time T_2 and quickly responds with its own random nonce N_A at T_3; the smartphone records T_2 and T_3. The terminal B receives this message at time T_4 and replies with the signed message $[A,B,N_A,N_B,m_1]_K$, where $m_1 = (T_{B0},T_1,T_4)$ and the message is signed with the shared key K.

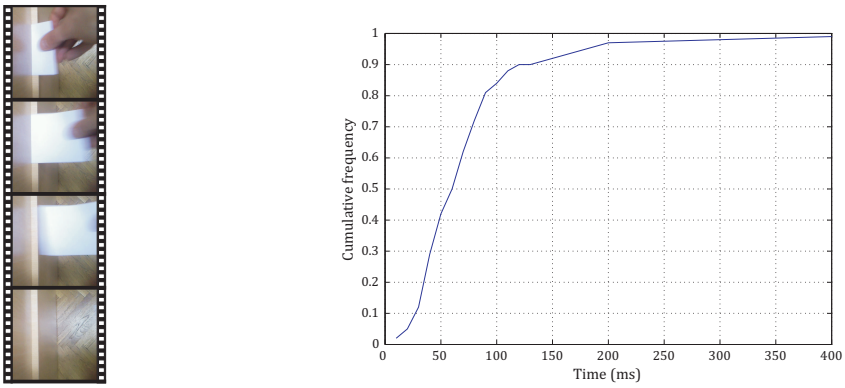
Upon a reception of the message $[A,B,N_A,N_B,m_1]_K$, the smartphone A verifies the message authenticity and that the end-to-end delay $\Delta t_{sync} \triangleq (T_2 - T_1 + T_4 - T_3)/2$ is bounded as follows $\Delta t_{sync} \leq \Delta t^*_{sync}$. If all the verifications are successful, the smartphone calculates the clock offset $offset_A \triangleq T_1 + \Delta t_{sync} - T_2$, and adjusts accordingly its clock. Having synchronized its clock with the terminal B, the smartphone verifies that they are *event synchronized*, meaning that the times T_A and T_{B1} are within the expected bound, i.e., $T_A - T_{B1} \leq \Delta t^*_{AB}$. For the reasonably small Δt^*_{AB}, this last verification essentially means that the smartphone A and the terminal B are synchronized with respect to the same event (the fortune cookie extraction). From the user performance study presented in Section 3.8, we obtain the value of Δt^*_{AB} to be 70 ms. For security reasons the terminal also limits the cookie extraction time to Δt^*_B, i.e., $T_{B1} - T_{B0} \leq \Delta t^*_B$. We study in detail the security implications of these time bounds later in Section 3.5. If all the verifications are successful, the protocol continues to Phase III.

Phase III: Reading the fortune cookie. At this stage the user opens (unfolds) the fortune cookie and reads its content (the nonce R) using her smartphone (this is denoted with a dashed arrow in Figure 3.4). The random nonce R can be encoded with QR codes (as shown in Figure 3.3) to make this process more user-friendly.

Phase IV: Final phase. In the last phase, the smartphone A sends to the terminal B the signed message $[B,R,m_2]_K$ (which can also be encrypted to ensure privacy), where m_2 represents transaction details (i.e., the amount of money, the account, the user's ID, etc.). Upon a successful verification of the last message, the terminal B provides the service (e.g. gives out the money, of course if the user has sufficient funds).

3.3.1 Event Synchronization with a Smartphone

We describe a method used by the user A to precisely timestamp the moment (T_A) at which the fortune cookie leaves the terminal. We search for a method that will allow the user to perform the event synchronization in a secure and user-friendly manner. The proposed method utilizes a smartphone equipped with a regular camera. Please note that, in our model,



(a) (b)

Figure 3.5. (a) A sequence of recorded frames that capture the fortune cookie extraction as seen by the user on her smartphone. (b) The cumulative frequency distribution of the cookie extraction time (for the 2 cm long QR encoded secret nonce R).

a smartphone is used as the replacement for a regular smartcard and as such is an additional hardware requirement.

At the end of the initial phase of the Forces protocol (Phase I in Figure 3.4), the user is instructed by her smartphone (i) to point the smartphone's camera towards the terminal and (ii) to begin pulling the fortune cookie (recall from the description above, this marks the beginning of Phase II of the protocol in Figure 3.4). The smartphone starts recording (without any user's intervention) upon the reception of the first clock synchronization message, i.e., $((B,A,N_B)$ in Figure 3.4). The recording ends with the last clock synchronization message (that is $[A,B,N_A,N_B,m_1]_K$). Hence, the terminal B controls the recording process of the user's smartphone (through these two message flows). At this stage the user holds the fortune cookie in her hand (the cookie resided in the user's trusted zone), while the terminal and the smartphone have synchronized clocks. What remains to be done on the user's side is to determine the moment T_A at which the cookie actually left the terminal and compare it against T_{B1}, the moment at which the terminal detected the same event - event synchronization. This is accomplished by presenting to the user a sequence of recorded frames on the smartphone as shown in Figure 3.5(a). The user has to simply select the first frame on which she sees the fortune cookie completely extracted from the terminal[2]. As each frame is timestamped relative to the beginning of Phase II, by selecting the appropriate frame, the smartphone can learn the time T_A at which the cookie left the terminal. Of course, the precision of this timestamping process will depend on the camera's frame rate. We provide a detailed security assessment of this method in Section 3.4. We also report the results of related user performance study in Section 3.8. At this stage of the Forces protocol, the smartphone and

[2]Already a large number of easy-to-use applications allow the user to grab frames from the recorded video [92, 93]. In these applications, the user simply slides through the frames and selects the most appropriate one.

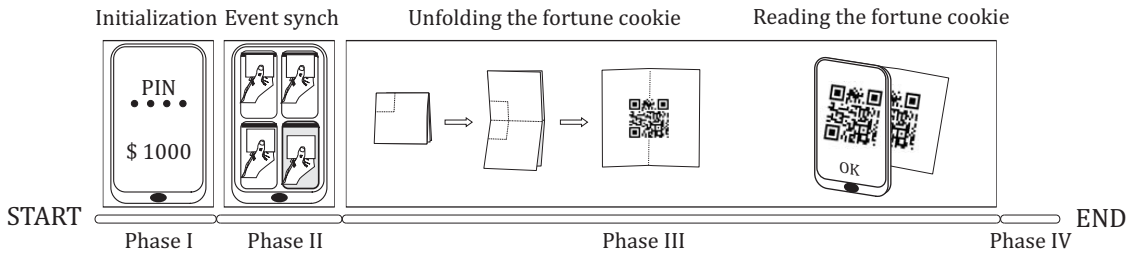


Figure 3.6. The timeline of the Forces protocol.

the terminal are *event synchronized* (they are synchronized with respect to the time at which the fortune cookie left the terminal's trusted zone and entered into the users's trusted zone - the user's hand). The protocol continues with Phase III, where the user opens and reads the fortune cookie; the complete protocol timeline is shown in Figure 3.6.

3.3.2 Alternative Event Synchronization Solutions

Here we briefly outline two alternative methods that can be used for the event synchronization purposes: pushbutton-based event synchronization and accelerometer-based event synchronization.

In the pushbutton method the smartphone learns the time T_A, by making the user push the button once she observes that the cookie left the terminal. Although this method is quite simple, it is prone to errors that can make the whole protocol insecure. For example, the adversary can set up a fake terminal with a long empty paper, acting as a fortune cookie. The user begins to withdraw the fake cookie and pushes the button on the smartphone while the cookie is still inside the terminal; this happens because she got used to shorter cookies (in normal conditions). The attacker stops the cookie extraction process for a short moment, reads the content of the genuine fortune cookie and prints/copies the genuine nonce R on the part of the fake fortune cookie which is still inside the fake terminal. The fake terminal finally releases the fake cookie for the user to pick it up. The effect of this attack is that the attacker extends the distance from which he can execute the relay attack. In our performance study with users (Section 3.8), we show that this kind of user mistakes can easily go undetected. We managed to trick 25 percent of tested users.

In an attempt to further increase the usability of the event synchronization process, we can use the accelerometer that is included as standard equipment for most smartphones. In this method, the user extracts the fortune cookie and, similarly to the previous method, briefly shakes his smartphone once she observes that the paper left the terminal. The smartphone logs the acceleration data and appropriately estimates the time T_A. While being highly usable, this method suffers the same problems as the pushbutton-based event synchronization. Moreover, this method is prone to false positives, where the smartphone can wrongly conclude (due to highly sensitive accelerometer) that the event synchronization has taken place

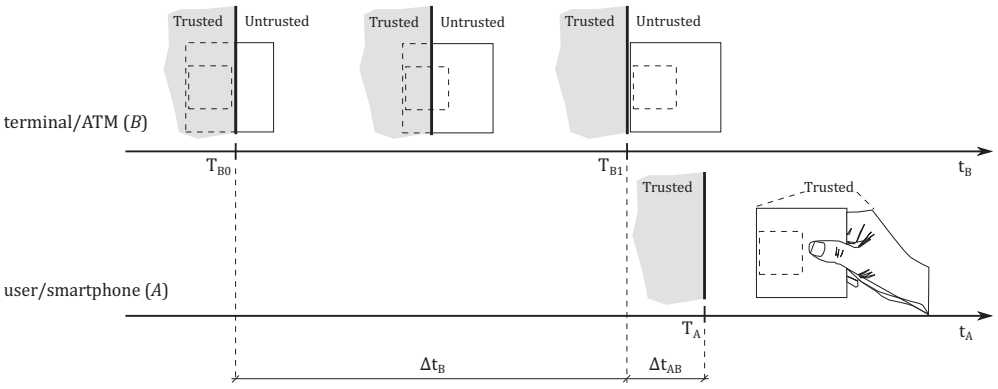


Figure 3.7. Unfolding of the cookie extraction from the terminal B. The part of the cookie carrying the secret nonce (dashed square) initially resides within the terminal's dispenser (the shaded part marked as a trusted zone*). As the user A begins to draw the fortune cookie (at T_{B0}), the hidden secret nonce enters a* untrusted zone*. The cookie completely leaves the terminal's trusted zone at T_{B1}, while the smartphone A records this event at T_A.*

while it has not.

3.4 Attacker and Protocol Timing Model

In this section, we introduce the attacker model, as well as the protocol timing model that will be used later in the formal security assessment of the Forces protocol. Forces is a multichannel protocol where information is sent over two channels: a regular radio channel and a special weakly unrelayable channel. We consider a strong attacker model in which the attacker has a full control over both channels, thus he can eavesdrop, drop, delay, relay, replay and modify messages sent over both channels. Although very powerful, we assume a realistic attacker in the sense that he has to perform some work (i.e., invest some time) to mount any of the above attacks. In other words, each attacking action costs the attacker some nonzero time.

As previously stated, Stajano et al. [1] argue that the attacker can not create a clone of the message that is transmitted over a special channel satisfying untransportability, uncloneability and unsimulability properties. In our model, we relax the properties of untransportability and uncloneability by restricting the time available to the attacker to realize the relay attack. For this reason, we assume only the existence of weakly unrelayable channels.

As outlined in Section 3.2, we propose to implement a weakly unrelayable channel using a plain paper (called a fortune cookie). The fortune cookie is created by the terminal B: it generates the secret nonce R, prints it on the paper, appropriately folds the paper and finally presents it at the terminal's dispenser (as shown in Figure 3.7). As can be seen from Figure 3.7, the part of the fortune cookie holding the secret nonce (the dashed square) resides inside the terminal B (the shaded part denoted as *trusted zone*). We assume that the paper is folded once or multiple times, depending on the desired security level (Figure 3.3). At

this stage the attacker does not have access to the part of fortune cookie residing inside the terminal and hence to the secret nonce R. We say that the fortune cookie resides in the trusted zone (of the terminal). The part of the fortune cookie outside of the terminal is within the untrusted zone. We assume that the attacker can manipulate the part of the fortune cookie that resides in the untrusted zone. Figure 3.7 shows the unfolding of the fortune cookie extraction process over time (as seen by the terminal and the user's smartphone). Recall, here T_{B0} marks the beginning of the Phase II of the protocol (see Figure 3.4); it is the moment (as recorded by the terminal) at which the user begins to draw the fortune cookie out. Likewise, T_{B1} is the time at which the fortune cookie completely leaves the terminal's trusted zone (Figure 3.7). We denote the cookie extraction time with Δt_B, i.e.,:

$$\Delta t_B = T_{B1} - T_{B0} \leq \Delta t_B^*. \tag{3.1}$$

For security reasons, we limit the cookie extraction time Δt_B to be smaller than Δt_B^*, where Δt_B^* is approximately 120 ms (this is obtained from the user performance study described later in Section 3.8). This security check is enforced by the terminal; the terminal will abort the protocol if this limit is violated. As we discuss below, relatively small Δt_B^* limits the attacker's ability to successfully clone the fortune cookie. Referring back to Figure 3.7 and the description of the Forces protocol, T_A represents the moment at which the cookie left the terminal (its trusted zone), as recorded by the user's smartphone. Please recall from Section 3.3.1, the smartphone learns T_A from the timestamp of the recorded video frame that the user has selected in Phase II (see Figure 3.5(a)). Here we assume that no human error has taken place (e.g., the user was not tricked by the attacker into selecting a wrong frame); this is related to the simulability attack discussed in Section 3.5.3 in greater detail. Please note that from this moment on (i.e., at the time $t \geq T_A$), the cookie resides in the user's trusted zone (in the user's hand as shown in Figure 3.7). This essentially means that from T_A on, the attacker cannot replace nor modify the legitimate fortune cookie. By the time T_A the terminal and the user's smartphone are securely synchronized, as we prove in the Appendix. Moreover, we prove that the smartphone holds the authentic value of the random challenge R issued by the legitimate terminal.

As discussed earlier, we require the user's smartphone to learn the moment when the fortune cookie leaves the terminal; the terminal timestamps this event with T_{B1} (Figure 3.7). Due to the limited framerate of the smartphone's camera, the smartphone will detect this event with some delay (proportional to its framerate) at time T_A. We denote the relative difference between T_A and T_{B1} with Δt_{AB}, that is:

$$\Delta t_{AB} = |T_A - T_{B1}| \leq \Delta t_{AB}^*. \tag{3.2}$$

In general, we expect T_A to be greater than T_{B1} as the terminal and the smartphone will be tightly synchronized (so we could drop the absolute value signs). Again, for security

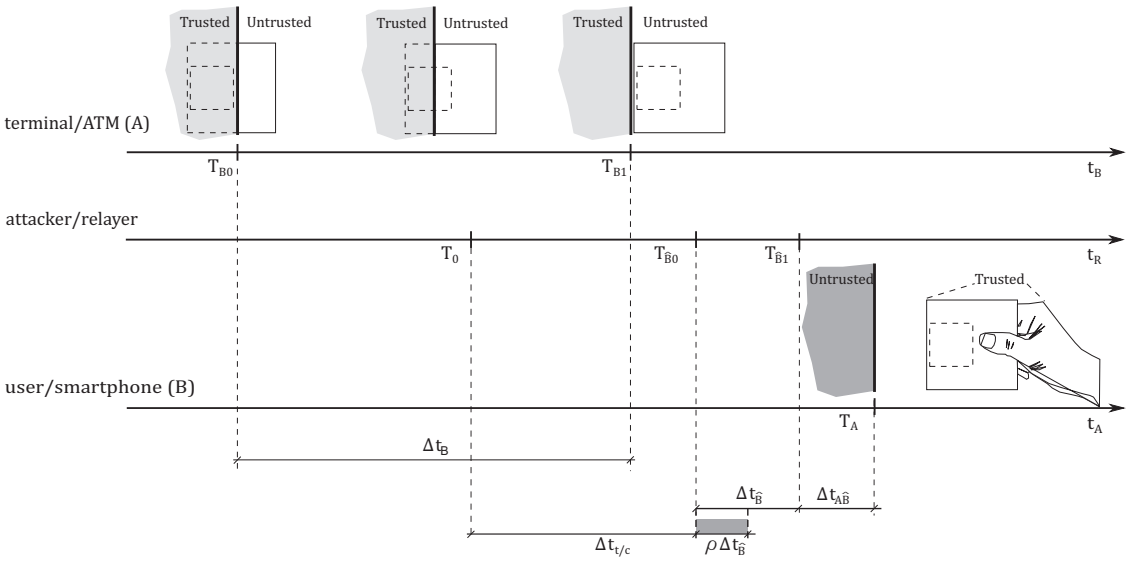


Figure 3.8. An example of the relay attack on the Forces protocol. A successful attack implies either (i) the transfer of the original fortune cookie from the trusted terminal to the fake/untrusted terminal under the attacker's control and/or (ii) reading the random nonce R from the original cookie, relaying the nonce R over to the fake terminal, cloning it (printing R on the fake cookie) and finally having the legitimate user extract the fake cookie from the fake terminal, all this within the period $(\Delta t^*_{AB} + \Delta t^*_B)$.

reasons (discussed in Sections 3.5.1 and 3.5.2), we limit Δt_{AB} to be smaller than some predefined value Δt^*_{AB} (around 70 ms for the camera model introduced later in this section). The smartphone aborts the protocol if this time difference is greater than Δt^*_{AB}.

To recap, together Δt_B and Δt_{AB} represent the period during which the fortune cookie (the part of the cookie holding the secret nonce R) resides in the untrusted zone. That is, within this time period the cookie travels from the terminal's trusted zone into the user's trusted zone (her hand) (Figure 3.7). Later in this section we provide detailed characterization of Δt_B and Δt_{AB}. Next, we derive an expression for the effective period available to the relaying attacker.

Relay attack timing model

In the Forces protocol the secret nonce R (sent within the fortune cookie) is used by the terminal to authenticate the user. The nonce R is a secret challenge that the legitimate user has to sign. Without knowing R, the user cannot authenticate successfully with the terminal (as we prove in the Appendix). Therefore, to be successful, the relaying attacker has to somehow transfer the random nonce R from the issuing terminal over to the honest user. Moreover, this has to be performed in such a way that the bounds (3.1) and (3.2) are met. Clearly, the attacker has only two possibilities: either (i) to physically transport the original fortune cookie from the legitimate terminal over to the honest user (*the transport attack*), or (ii) to guess and/or somehow read a part or the whole secret nonce R from the original

fortune cookie and clone it (produce a fake cookie holding the original random nonce R) at the user's side (*the cloning attack*). As stated earlier both attacking strategies are subject to the tight time constraints (3.1) and (3.2). However, the attacker can potentially gain an extra time by mounting a successful *simulability attack*; here the attacker attempts to simulate some physical aspect of the special weakly unrelayable channel. Recall, from Section 3.3.1, the smartphone learns the time T_A from the video frame selected by the user in the Phase II of the Forces protocol. So the attacker can try to trick the user into selecting an earlier frame (or pressing the button at much earlier time in the push-button event synchronization method). In this way, the bounds (3.1) and (3.2) will be both met, yet the attacker has gained an extra time to relay the fortune cookie and/or its content. We assess the security against the simulability attack in Section 3.5.3, while here we focus on the transport and cloning attacks.

Figure 3.8 shows the unfolding of the fortune cookie extraction over time in the presence of an active attacker. In the figure, the time T_0 represents moment at which the attacker begins with his relaying activity. In our model we assume $T_0 \geq T_{B0}$. As shown in Figure 3.8, until the moment T_{B0}, the random nonce R resides in the terminal's trusted zone and as such is not available to the attacker. The honest user begins pulling the fortune cookie (original or fake depending on the attacking strategy) from the fake terminal (controlled by the attacker) at the moment $T_{\widehat{B}0}$; we use "hat" to signify that this time is affected by the attacker. Likewise, the time $T_{\widehat{B}1}$ is the moment at which the fortune cookie (original or fake) completely leaves the fake terminal. As before, T_A represents the moment at which the cookie left the fake terminal as recorded by the honest user's smartphone. Using this timing model, the time available to the attacker for the cloning and/or transport attack equals $(T_A - T_0)$. Referring back to Figure 3.8, we can write the following equality:

$$T_A - T_0 = \Delta t_{t/c} + (1-\rho)\cdot \Delta t_{\widehat{B}} + \Delta t_{A\widehat{B}}, \tag{3.3}$$

where $\Delta t_{t/c}$ is the time the attacker invests in relaying activities (e.g., unfolding the fortune cookie, physically transferring it, printing and creating a fake cookie, any other relaying action), $\Delta t_{\widehat{B}} \triangleq T_{\widehat{B}1} - T_{\widehat{B}0}$ is the cookie extraction time (by the honest user) at the fake terminal, and $\Delta t_{A\widehat{B}}$ is the delay introduced by the legitimate smartphone's camera (not controlled by the attacker). The factor $\rho \in \{0,1\}$ is used to mathematically represent the possibility that the attacker's relaying/cloning activities can take place in parallel/simultaneously with the fortune cookie extraction by the honest user. We call ρ the *simultaneity factor* and define it as follows:

$$\rho = \frac{\Delta t_{t/c} - \left(T_{\widehat{B}0} - T_0\right)}{\Delta t_{\widehat{B}}}\ . \tag{3.4}$$

For example, $\rho = 1$ implies simultaneous activities by the attacker and the honest user. It also implies longer $\Delta t_{t/c}$ (i.e., $\Delta t_{t/c} = T_{\widehat{B}1} - T_0$) for the given $(T_A - T_0)$, or in other words, more time available for the attacker (Figure 3.8)). On the other hand $\rho = 0$ gives $\Delta t_{t/c} = T_{\widehat{B}0} - T_0$

and implies serial actions (first the attacker then the honest user). By combining the time constrains (3.1), (3.2) and the expression (3.3), we can finally derive the following bound on $\Delta t_{t/c}$ - the time window during which the attacker can mount the relaying attack:

$$\Delta t_{t/c} \leq \left(\Delta t^*_{AB} - \Delta t_{A\widehat{B}}\right) + \left[\Delta t_B - (1-\rho)\cdot \Delta t_{\widehat{B}}\right] - (T_0 - T_{B0})\,. \tag{3.5}$$

We use the equation (3.5) to reason about different attacking strategies and their implications in the context of the transport and cloning attacks. Detailed analysis is deferred until Section 3.5, where we provide formal definitions of the weak untransportability and the weak unclonability properties of the paper channel. Before moving on with this analysis, we first characterize the time periods Δt_B and Δt_{AB} (and $\Delta t_{A\widehat{B}}$) that appear in the equation (3.5).

Distribution of the Cookie Extraction Time (Δt_B)

Recall, from the previous discussion and the equation (3.1), Δt_B represents the fortune cookie extraction time. It is a user-dependent (random) variable, which reflects the speed with which users can pull the fortune cookie out of a terminal. In order to assess the distribution of Δt_B, as well as other aspects of the proposed protocol, we conducted an extensive user performance study. In this section, we only present the results related to the cookie extraction time Δt_B, while in Section 3.8 we provide more detailed description of the experimental setup and other results obtained from this user study. Figure 3.5(b) gives a frequency distribution of the cookie extraction time Δt_B; the results obtained from the study were scaled to 2 cm long physical representation of the nonce R, i.e., 2cm long QR code. As can be seen in Figure 3.5(b), in 90% of cases, the tested users extracted the fortune cookie within 120 ms. For this reason, we upper bound Δt_B by $\Delta t^*_B = 120$ ms. If the terminal measures the cookie extraction time to be greater than 120 ms it will abort the protocol. Recall, this bound is obtained for 2 cm long physical representation of the nonce R. If a smaller bound Δt_B is required, it is possible to further reduce the physical representation of the secret nonce or require the users to pull out the cookies faster thus increasing the probability of false positives.

Characterization of the Delay Induced by a Camera (Δt_{AB})

As discussed above (see also Figure 3.7), Δt_{AB} is the time difference between T_A (the moment at which the user's smartphone recorded the act of cookie leaving the terminal) and T_{B1} (the actual time at which the cookie left the terminal; T_{B1} is recorded by the terminal itself). Please note that $\Delta t_{A\widehat{B}}$ is similarly defined (i.e., $\Delta t_{A\widehat{B}} = T_A - T_{\widehat{B}1}$), the only difference is that the user interacts with the fake terminal in this case (Figure 3.8).

Given that the smartphone and the legitimate terminal are synchronized (as proved in the Appendix; the synchronization error being in the order of couple of milliseconds [91]), the

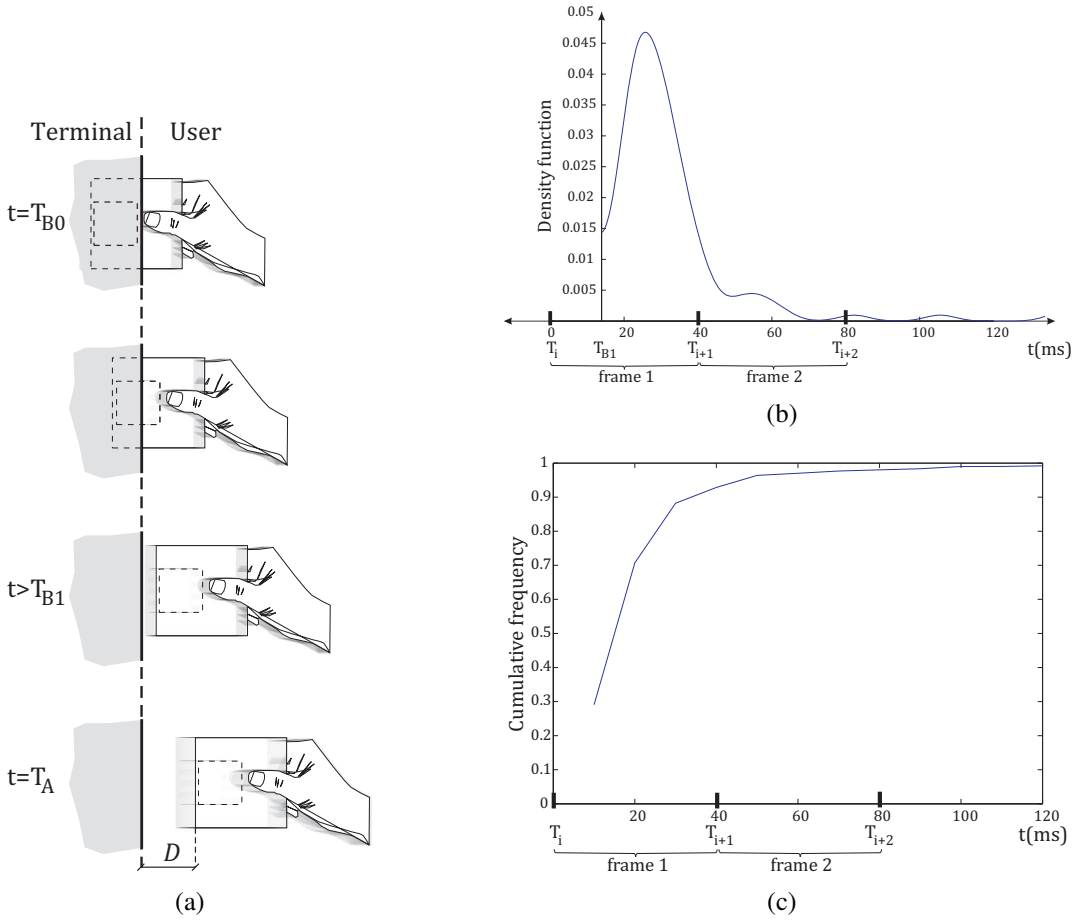


Figure 3.9. Motion blur effects on the delay Δt_{AB} (and $\Delta t_{\widehat{AB}}$): (a) A sequence of motion blurred frames as seen by the user on her smartphone, (b) estimate of the distribution of the time at which the cookie reaches the distance $D = 0.5$ cm from the terminal, and (c) the cumulative frequency distribution of $\Delta t(D)$ - the time it takes the cookie to trave the distance $D = 0.5$ cm.

delay Δt_{AB} is influenced primarily by the limited framerate of the smartphone's camera and the related motion blur effects (due to the moving cookie). This can be seen in Figures 3.5(a) and 3.9(a) that show the sequence of four frames recorded by the user's smartphone. As discussed in Section 3.3.1, the smartphone learns T_A (the moment the cookie leaves a terminal) from the frame selected by the user during the event synchronization phase of the Forces protocol. Thus, in the examples shown in Figures 3.5(a) and 3.9(a), the user should select the 4th frame, where she can clearly see that the cookie has left the terminal. This is in spite of the fact that the cookie actually left the terminal at the earlier time corresponding to the 3rd frame (Figure 3.9(a)). It is the motion blur that introduces this uncertainty in the user's decision and thus in the delay Δt_{AB}. In what follows, we will show that for a standard 25-30 fps camera, with a high probability we can say that Δt_{AB} is smaller than 70 ms or approximately 1.75 video frames.

Motion blur occurs when a recorded object (the cookie in our case) moves during the camera exposure time or the camera moves itself (we do not consider this case in our sim-

plified model). We assume the following motion blur model [94, 95]. Let us consider the 3rd frame shown in Figures 3.5(a) and 3.9(a). The opaque object (the fortune cookie) moves (from left to right) during the process of image capture. In the captured image, the cookie boundary on the left and right partially occludes the background, and is shown as *semitransparent*. The center of the cookie, in the captured image, is not mixed with the background since during the camera exposure time background is covered by the cookie. The transparency for each image pixel is determined by the proportion of the time that the background is exposed to the camera. Now, the resulting captured image I is a linear combination of the foreground image F (the moving object) and the background image B (e.g., the floor of the terminal) through the *alpha channel*, that is, for the given image pixel p we have:

$$I(p) = \alpha(p)F(p) + (1-\alpha(p))B(p),\ \alpha(p) \in \{0,1\}\ . \tag{3.6}$$

The $\alpha(p)$ value corresponds to the fraction of the exposure time (the frame duration) during which the moving object projects to (is captured by) the pixel p. Thus, $\alpha(p)$ close to 1 implies that the pixel p captures only the cookie (p is clear/not mixed with the background), whereas lower positive $\alpha(p)$ values result in the pixel p blended to the background scene.

We use the following *user model* to describe the user's behavior. The user, who starts pulling out the cookie, will see the cookie completely detached from the terminal (on her smartphone) when the alpha value $\alpha(p_D)$ of the image pixel p_D that captures the physical scene at the distance D from the terminal, is smaller than some predefined value α_D, i.e., $\alpha(p_D) < \alpha_D$ (see Figure 3.9(a), the last frame at $t = T_A$). Let us denote the frame selected by the user (using the above frame-selection convention) as frame j. The frame j begins at T_j and lasts Δt_f; we assume all frames to be of the same duration. Following the above model of motion blur, the moving cookie will project to the pixel p_D during at most the initial α_D fraction of the frame j. Then we know that the cookie (its left side) must have already left the terminal at the time $T_j + \alpha_D \Delta t_f$. We conveniently take this time as T_A, that is, $T_A = T_j + \alpha_D \Delta t_f$. We are now ready to derive an upper bound on Δt_{AB}.

To be more specific, let us set D to 0.5 cm in our user model. It will be helpful to consider the scenario shown in Figure 3.9(b), where we overlay the probability density function of the time $\Delta t(D)$ it takes the cookie to travel the distance $D = 0.5$ cm, on top of two video frames of the fixed duration Δt_f. In this scenario, the terminal recorded that the cookie left it at T_{B1}; T_{B1} happened to fall in the interval corresponding to the frame i (i.e., $T_i < T_{B1} < T_{i+1}$). The pulled cookie will reach the distance $D = 0.5$ cm at the moment equal to T_{B1} plus the time $\Delta t(D)$ sampled from the overlayed distribution; the distribution is obtained by scaling the distribution of the cookie extraction time Δt_B to 0.5 cm. The random time $(T_{B1} + \Delta t(D))$ will land in either the frame i or some other frame ℓ following the frame i. Then, according to our user and motion blur models, if $(T_{B1} + \Delta t(D) - T_\ell)/\Delta t_f < \alpha_D$, the user will select the frame ℓ as the one that shows the cookie detached from the terminal. In turn, the smartphone

will set $T_A = T_\ell + \alpha_D \Delta t_f$, as discussed in the previous paragraph. Otherwise, the user will select the next frame, frame $(\ell+1)$ and hence we will have $T_A = T_{(\ell+1)} + \alpha_D \Delta t_f$. It is easy to check that in general T_A can be expressed as follows:

$$T_A = (k + \alpha_D)\,\Delta t_f + T_i, \text{ with } k = \left\lceil \frac{T_{B1} - T_i + \Delta t(D)}{\Delta t_f} - \alpha_D \right\rceil . \tag{3.7}$$

Using the equation (3.7), we can derive the following bound on Δt_{AB}; in the derivation we assume $T_A > T_{B1}$ (which implies the smartphone and the terminal to be well synchronized).

$$\begin{aligned}
\Delta t_{AB} &= T_A - T_{B1} && \text{(by definition)} \\
&= \left(\left\lceil \frac{T_{B1} - T_i + \Delta t(D)}{\Delta t_f} - \alpha_D \right\rceil + \alpha_D \right) \Delta t_f + T_i - T_{B1} && \text{(by equation (3.7))} \\
&\leq \left(\frac{T_{B1} - T_i + \Delta t(D)}{\Delta t_f} - \alpha_D + 1 + \alpha_D \right) \Delta t_f + T_i - T_{B1} && (\lceil x \rceil \leq x + 1) \\
&= \Delta t(D) + \Delta t_f \ .
\end{aligned}$$

Let us plug in some numbers into this bound to put it into perspective. The CDF of the random variable $\Delta t(D)$ with $D = 0.5$ cm is shown in Figure 3.9(c), from which we can read $P(\Delta t(D) \leq 30 \text{ ms}) > 0.88$. In addition, for 25 fps cameras we have $\Delta t_f \leq 40$ ms. We can conclude that, for the selected parameters, $\Delta t_{AB} \leq 70$ ms or about 1.75 video frames, with probability at least 0.88. Based on this argument we set Δt^*_{AB} in the expression 3.5 to 70 ms, i.e., $\Delta t^*_{AB} = 70$ ms.

3.5 Security Properties of a Paper-Based Weakly Unrelayable Channel

The goal of this section is to show that the appropriate combination of three elements (weak uncloneability, weak untransportability and unsimulability) can indeed implement a weakly unrelayable channel. The proof based on the random oracle model is available in the Appendix. In the rest of this section we first analyze weak uncloneability and weak untransportability properties of the special channel, and later we focus on unsimulability property.

3.5.1 Transport Attack and Weak Untransportability

The weak untransportability property is required to mitigate the transport attack. Recall, in the transport attack the attacker physically transports the original fortune cookie holding the secret nonce R between the legitimate terminal and the honest user. Referring back to Figure 3.8 this attack is characterized by letting T_0 be equal to T_{B1}. This means that the distance over which the original cookie is relayed/transported is measured from the outermost point

of the terminal dispenser (see Figure 3.8). Please note that we do not consider the tricky form of the attack in which the attacker would simply cover the legitimate terminal with a fake-terminal mask. No existing solution can prevent this type of attacks, since the attacker operates within tolerable error margins of the given protocol (e.g., for the protocol in [90], the timing error of only 1 ns corresponds to the distance of 30 cm). Plugging $T_0 = T_{B1}$ into the inequality (3.5), it simplifies to:

$$\Delta t_t \leq (\Delta t^*_{AB} - \Delta t_{A\widehat{B}}) - (1-\rho) \cdot \Delta t_{\widehat{B}} \tag{3.8}$$

Here, Δt_t is the length of a time window during which the attacker has to transport the original cookie; longer transport times would violate the Forces timing requirements, thus resulting in the aborted protocol. Please note that we drop the subscript c from the expression (3.5), since we consider only the transport attack in this subsection.

The first important observation is that the bound on Δt_t does not depend on Δt_B (the cookie extraction time); that is, the attacker gains no extra time by quickly pulling (i.e., $\Delta t_B \to 0$) the fortune cookie. Furthermore, the simultaneity factor ρ is 0 in the case of the transport attack; i.e., the user cannot begin extracting the original cookie while the attacker still relays/transports it. Indeed, in the transport attack the honest user, by assumption, always extracts the original (physically untampered) cookie; the cloning attack, discussed later, involves physical manipulation of the legitimate cookie. Without physically modifying the original cookie, $\rho > 0$ would mean that the relaying distance must be very short (in the order of the length of the cookie). This corresponds to the situation in which the user would interact with the legitimate terminal through a fake mask mounted on the legitimate terminal. So, by conservatively setting $\Delta t_{A\widehat{B}} = 0$ and with $\rho = 0$ we get:

$$\Delta t_t \leq \Delta t^*_{AB} - \Delta t_{\widehat{B}} \,. \tag{3.9}$$

To put this bound into perspective, let us plug-in some numbers in the last inequality. In the previous section we derived $\Delta t^*_{AB} = 70$ ms, while from the cumulative frequency distribution of the cookie extraction time $\Delta t_{\widehat{B}}$, given in Figure 3.5(b), we have $P(\Delta t_{\widehat{B}} > 25 \text{ ms}) > 0.9$. Therefore, $P(\Delta t_t < 45 \text{ ms}) > 0.9$. We argue that 45 ms is a way too short period for the attacker to physically transfer a piece of paper/cookie over larger distances. Within such a short time the attacker has to physically transfer the fortune cookie, load it into the fake terminal, make sure that the user has extracted it from the terminal and that the user's smartphone recorded this. For comparison, the shutter speed of a regular (25 fps) camera is 40 ms. The average (human eye) blink duration is estimated to about 300 ms, ranging from 1/6 of a second (160 ms) to 2/3 of a second (600 ms) [96]. According to [97], typically the drivers airbag inflates within approximately 60-80 ms after the first moment of vehicle contact. In addition, in [89] one can find estimates of the delay times induced by relaying NFC communication over various channels. For example, it is shown that the on-device access by an

application to a secure element on a NFC-enabled phone takes 50-80 ms. *We therefore claim that a successful transport attack within 45 ms is unrealistic, even for very short distances.*

Next we provide a more precise characterization of distances over which the attacker can potentially transport a fortune cookie within the time Δt_t. Since this attack involves physical transfer of a piece of paper, the attacker and the special paper channel are subject to certain physical laws and constraints such as the maximum velocity v_m, the initial and the terminal velocity of the paper, as well as the maximum acceleration/deceleration a. In this model, we assume that the initial and the terminal speed are both zero. Under the above assumptions, the maximum distance $d(\Delta t_t)$ that a fortune cookie can travel within Δt_t is obtained by solving the following optimization problem:

$$d(\Delta t_t) = \max \int_0^{\Delta t_t} v(t)dt \quad \text{subject to:} \quad v(t) \leq v_m,\ |\dot{v}(t)| \leq a\ .$$

It is well known that the optimal strategy, maximizing $d(\Delta t_t)$, is so called "bang-bang" strategy [98], where the paper cookie is accelerated (with a) from the initial zero speed to the maximum velocity v_m, then it keeps moving at v_m and finally it decelerates (with a) to the zero terminal speed (at the fake terminal). The "bang-bang" strategy yields the following optimal solution for $d(\Delta t_t)$:

$$d(\Delta t_t) = \begin{cases} v_m(\Delta t_t - v_m/a) & \text{for } v_m \leq a\Delta t_t/2; \\ a\Delta t_t^2/4 & \text{otherwise (i.e., } v_m \text{ not attainable within } \Delta t_t)\ . \end{cases} \tag{3.10}$$

This expression gives us insight into physical constraints and possible tradeoffs on the attacker's side. Let us use some numbers to get better idea about attainable distances. Let us assume that the cookie can reach the maximum speed of around 70 m/s or 252 km/h (being the release speed of modern slings) with the maximum acceleration of 2000 m/s^2 (around 200 G). Using the above expression for the optimal $d(\Delta t_t)$, we can see that assumed v_m is not attainable within $\Delta t_t = 45$ ms, so we use the bottom expression to get $d(\Delta t_t) = 1.01$ m. By decreasing Δt_t to 30 ms, the distance reduces to 45 cm. Even with such advantageous assumptions for the attacker (i.e., a rather large acceleration and conservative bounds on the available time Δt_t) the maximum reachable distance is rather limited.

For an arbitrary large (but finite) time Δt_t, we can describe the advantage of the attacker in the transport attack as shown in Figure 3.10(a). *The advantage $\varepsilon_t(d, \Delta t_t)$ of the attacker is defined as the probability that he successfully relays/transports a fortune cookie over the fixed distance d within the time Δt_t.* We know from the expression (3.10), that for the constrained attacker (i.e., fixed v_m and a) and every Δt_t, there exists the corresponding maximum relaying distance $d(\Delta t_t)$. Distances larger than this one are not within the attacker's reach. In Figure 3.10(a) we can qualitatively see the advantage of the attacker for two values of Δt_t with the corresponding maximum reachable distances d_1 and d_2. In this example, we clearly

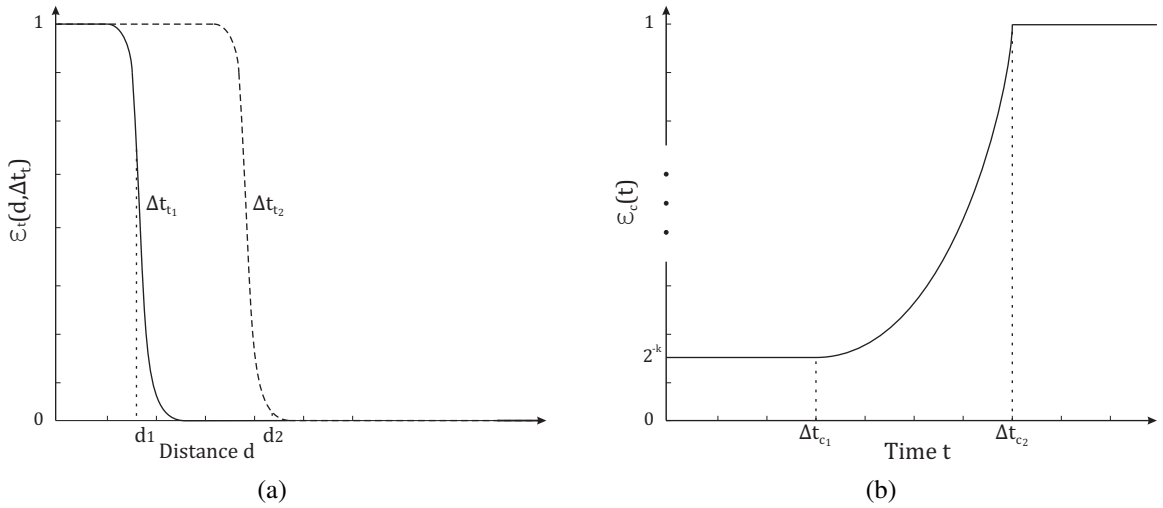


Figure 3.10. Qualitative advantage of the attacker in: (a) the transport attack, and (b) the cloning attack.

have $\Delta t_{t1} < \Delta t_{t2}$. Please note in Figure 3.10(a) that $\varepsilon_t(d,\Delta t_t)$ satisfies another reasonable assumption, namely, $d_2 > d_1$ implies $\varepsilon_t(d_2,\Delta t_t) \leq \varepsilon_t(d_1,\Delta t_t)$ for the fixed Δt_t.

Finally, observe that the weak untransportability property is directly related to the advantage $\varepsilon_t(d,\Delta t_t)$. *We say that, for the targeted distance d, a given (paper-based) channel satisfies the weak untransportability property if the advantage* $\varepsilon_t(d,\Delta t_t)$ *is satisfactorily small.* We will use this security definition later in the Appendix, where we formally prove the security of the Forces protocol.

3.5.2 Cloning Attack and Weak Uncloneability

The cloning attack is a more general attacking strategy compared to the transport attack. In the cloning attack the attacker guesses and/or reads the value (a part or the whole) of the secret random nonce R, relays R (using a fast (radio) channel) close to the victim (the honest user) and produces a fake fortune cookie imprinted with the nonce R. As before, the attacker has to perform the attack while meeting the time bound given by the expression (3.5). It follows from this expression that the optimal strategy (timewise) is characterized by the following constraints:

- $T_0 = T_{B0}$ (i.e., the cloning attack starts at the early stage of the fortune cookie extraction);
- the simultaneity factor ρ equals 1 (the attacker's relaying/cloning activities extend over to the moment $T_{\widehat{B}1}$, the moment at which the cloned cookie leaves the fake terminal - see Figure 3.8);
- and $\Delta t_B = \Delta t_B^*$ (essentially meaning that the attacker uses all the available time to

extract the legitimate fortune cookie - fast extraction only reduces the available time).

Inserting these constraints into the expression (3.5), it simplifies to:

$$\Delta t_c \leq \left(\Delta t^*_{AB} - \Delta t_{A\widehat{B}}\right) + \Delta t^*_B \,. \tag{3.11}$$

We drop the subscript t from the expression (3.5), since we consider only the cloning attack. *By comparing Δt_t (the expression (3.9)) with Δt_c, we can see that timewise it is more optimal for the attacker to launch the cloning attack.*

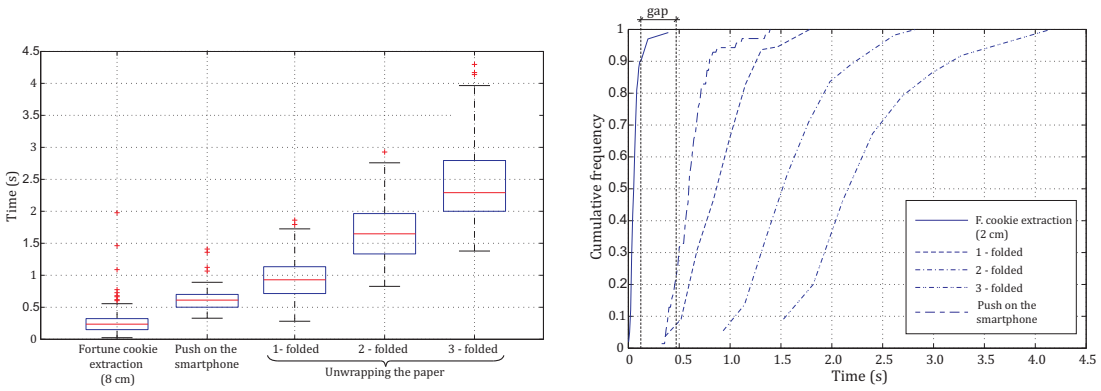
We next discuss a potential work required on the side of the attacker to successfully mount the cloning attack within the time given by the expression (3.11):

- **Naive guessing attacker:** The simplest attack would involve guessing the secret nonce R and preparing in advance a fake fortune cookie; the attacker the fake cookie well ahead. "All" the attacker has to do during the attack is to relay the legitimate messages transmitted over a regular channel (while making sure that the extraction of the legitimate fortune cookie and of the fake one is synchronized). While the work by the attacker is relatively low, the probability that he will succeed is only 2^{-k} (in a single attempt), where k is the size of the secret nonce R.

- **Very strong X-Ray attacker:** We next consider a very strong and sophisticated attacker, equipped with a X-Ray scanner, who tries to learn the content of the fortune cookie without actually opening it. To read the content of the cookie the attacker can apply X-Ray tomography [99]. We know, from the expression (3.11), that X-Ray recording and processing, relaying the learned nonce, printing it on the fake cookie and any other required action have to be accomplished within the time Δt_c. For $\Delta t^*_{AB} = 70$ ms (as derived in Section 3.4), $\Delta t_{A\widehat{B}} = 0$ and $\Delta t^*_B = 120$ ms (Section 3.4), the expression (3.11) evaluates to $\Delta t_c \leq 190$ ms.

  We claim that performing all these actions in such a very short time could be very challenging. As discussed in the context of the transport attack, the average (human eye) blink duration is estimated to about 300 ms, ranging from 1/6 of a second (160 ms) to 2/3 of a second (600 ms). Moreover, the WiFi relay path was estimated in [89] to introduce the delay in the range of 100 to 210 ms.

  Even if we quite conservatively assume that we will face such a sophisticated attackers in reality, the paper based channel can be strengthened in a cheap and a simple way. We could fold the fortune cookie multiple times, or even better make it into a crumpled paper ball (as the one shown in Figure 3.2). As discussed in [99], in this way we can significantly reduce the chance for the attacker to probe the interior of the fortune cookie. In this way we can make the task of reading the content of the cookie without actually opening it, impossible. At last, since Δt^*_{AB} and Δt^*_B are system parameters, we



(a) (b)

Figure 3.11. Manual cookie opening experiment: (a) Box plot representation of the times of fortune cookie extraction, push on the smartphone after the extraction and unfolding a 1-folded, 2-folded and 3-folded fortune cookie, and (b) the corresponding cumulative frequency distribution.

can always reduce them (so that Δt_c is for example 100 ms) at the cost of the increased rate of false positives.

- **Realistic cloning attacker:** A realistic attacker is equipped with a regular camera and controls a fake terminal. This attacker extracts the legitimate cookie from the legitimate terminal and manually unfolds it. The attacker then uses his camera to record a part or the whole secret nonce R, relays it to the fake terminal where R is printed on the fake fortune cookie. We conservatively assume that the attacker does not fold the cookie at the fake terminal; moreover, we assume that the honest user will not notice this (a human error).

  For this attacker model, we are interested in the speed with which the attacker can manually unfold a fortune cookie. We challenged 20 students to open/unfold a folded paper (one, two or three times) as fast as they could. We motivated the students to give their best by rewarding the top three *fastest hands*; the first place reward was 40 USD, the second place 20 USD and the third place 10 USD. Each student had three chances to win the prize for each 1-time, 2-times and 3-times folded paper test (totaling 9 opening attempts per student). For each student we noted only the fastest times. The results of this competition are presented in Figure 3.11. As expected, the unfolding time increases with the number of times the paper is folded. The best result over all the tests ($20 \times 3 = 60$) is 281 ms (for the one-time folded paper). This result exceeds by far the conservative limit $\Delta t_c \leq 190$ ms.

  This is nicely shown in Figure 3.11(b) where, in addition to the cumulative frequency distributions of fastest opening times, we also plot the CDF of the cookie extraction time Δt_B^*. It is easily seen that the gap between the CDF curve of Δt_B^* and the onset of

the CDF curve for the 1-time folded paper is over 300 ms. Based on these experiments we conclude that this attacking strategy involving (manually) opening the cookie will fail with a high probability.

- **Powerful realistic cloning attacker:** The difference between this attacker and the previous one is that here we assume that the attacker holds more expensive cameras (e.g. high speed, pipe snake-like cameras, cameras that use very bright lights). With such equipment the attacker can increase the probability of a successful attack by trying to probe inside a closed cookie. However, as discussed above, we can reduce this probability by adjusting the way and the number of times the paper is folded. Likewise, we can reduce Δt_B^* (the maximum allowable extraction time) by reducing the size of the fortune cookie, or requiring a faster user's camera (thus decreasing Δt_{AB}^* and Δt_c).

Based on the previous analysis of the whole spectrum of possible attackers (ranging from naive to very strong ones) we can qualitatively represent the advantage of the cloning attacker as shown in Figure 3.10(b). We use $\varepsilon_c(\Delta t_c)$ to denote this advantage: *the probability that an attacker successfully mounts the cloning attack within the time* Δt_c. It is interesting to observe the important difference between $\varepsilon_c(\Delta t_c)$ and the attacker's advantage $\varepsilon_t(d, \Delta t_t)$ in the case of the transport attack. Thus, $\varepsilon_c(\Delta t_c)$ does not depend on the relaying distance d, only on the available time Δt_c; this is because the cloning attacker does not relay any information over the special paper channel. Since the attacker can always try to guess the random nonce R (with probability 2^{-k}) the following must hold: $\varepsilon_c(\Delta t_c) \geq 2^{-k}$ for any $\Delta t_c > 0$. Moreover, following our previous discussion, it is reasonably to assume that $\varepsilon_c(\Delta t_{c1}) \leq \varepsilon_c(\Delta t_{c2})$ for $\Delta t_{c1} < \Delta t_{c2}$ (see Figure 3.10(b)).

We say that a given special channel satisfies the weak unclonability property if $\varepsilon_c(\Delta t_c)$ *is satisfactorily small.*

3.5.3 Unsimulability

Unsimulability property refers to hardness of simulating some physical aspect of a special weakly unrelayable channel. In the context of our Forces protocol, we realize such a channel by appropriately orchestrating the three elements: a piece of paper (fortune cookie), a smartphone and the end user. The user records the cookie extraction using her smartphone and in turn selects the frame (from the recorded video) that corresponds to the moment (T_A) at which the paper/cookie left the terminal. In this way, the user's smartphone learns time T_A which it uses (along with T_{B1}) to decide if the protocol has to be terminated or not (Section 3.4, expression (3.2)). Consequently, the security of our protocol to some extent relies on the alertness of a user taking part in the protocol. We stress here that this is the only step in Forces protocol where we rely on user's alertness.

A potential attacker can try to exploit this fact by tricking the honest user into selecting a wrong frame, the one that does not correspond to the actual moment at which the cookie left

the terminal. For example, the attacker can try to mimic/simulate the situation in which the cookie already left the terminal by coloring the outer half of the (fake) cookie using some dark (background) color. This would cause the effect similar to the blurring effect shown in Figure 3.9(a), the third frame. The attacker hopes that the honest user will not notice this deception and that she will select, as the synchronizing frame, the one that corresponds to the moment at which the fake cookie has not yet left the fake terminal. In this way, the attacker gains an additional time to relay the content (the nonce R) of the legitimate cookie, while all the timing constraints will be met.

Similar simulability attack is possible against the pushbutton event-synchronization method presented in Section 3.3.2. There, the attacker can exploit user's expectation of a small cookie by using a larger paper for a fake fortune cookie. The user, used to short cookies, begins to pull the paper and pushes the button on the smartphone while the (larger) fake cookie is still inside the terminal. The attacker blocks the paper extraction for a while, thus gaining the additional time to relay the content of the legitimate cookie to the fake one. In our performance study (presented in Section 3.8), we managed to trick 25% of users in this way.

In the context of the Forces *protocol the unsimulability property refers to the difficulty of mounting such attacks.*

The advantage of the camera-based event-synchronization (compared to other synch methods) is that the user can possibly review the extraction process (multiple times if needed) on her smartphone (e.g., by simply sliding through the recorded frames). The user will drop the ongoing session if she observes something unusual. This will especially be the case with "highly motivated" users that use the Forces protocol to authorize a high value transactions in a unknown environment (e.g., withdrawal of a larger amount of money from a given terminal in a foreign country).

We use ε_s to denote the probability that a given user will be successfully deceived by the attacker who launches some form of th simulability attack. As discussed earlier, a successful simulability attack only buys the attacker an extra time to perform either the transport or cloning attack. As we prove in the Appendix, the simulability attack increases the overall attacker's advantage against our protocol by the factor $\varepsilon_s \cdot \left(\varepsilon_t(d,\Delta t_t^{'}) + \varepsilon_c(\Delta t_c^{'})\right)$, where $\Delta t_t^{'} < (t + \Delta t_{out} - T_{B1})$ and $\Delta t_c^{'} < (t + \Delta t_{out} - T_{B0})$, t being the time instant at which the protocol session begins and Δt_{out} the allowable session duration. Please note that $\Delta t_t < \Delta t_t^{'}$ and $\Delta t_c < \Delta t_c^{'}$; in other words, a successful simulability attack buys the attacker an extra time to perform either the transport or cloning attack. Moreover, the bounds on $\Delta t_t^{'}$ and $\Delta t_c^{'}$ are not tight, as the user has to extract the cookie, open it and have it read by her smartphone all before the session times out at $(t + \Delta t_{out})$.

In the following section we state our main security result.

3.6 The Main Security Result

In this section, we first formally define what we mean by a secure (*mafia-fraud resistant*) protocol. Then we state our main security result in Theorem 1. We use two parameters $d \geq 0$ and $\varepsilon \in [0,1]$ to characterize the resistance of the given protocol against the mafia fraud attack. The interpretation of these parameters at a high level is as follows: d represents the maximum distance over which a mafia fraud adversary is *successful* against the given protocol, with the probability at most ε. Definition 1 clarifies the meaning of *attacker being successful*.

Definition 1. *We say that a given protocol, executed between two honest parties A and B, is* (d,ε)-mafia fraud resistant*, if the following holds except for the probability* ε*: a honest party B* accepts *to provide the service (e.g., the terminal accepts a request for money as legitimate) at time instant* $(t+\Delta t_s)$ ***only if*** *(1) a honest party A (e.g., the user) has requested that service from B during period* $[t,t+\Delta t_s)$*, and (2) A and B have been within the distance d of each other during that period; here* Δt_s *represents the allowable session duration.*

The first condition implies that B has successfully authenticated the user A, while the second condition asserts that the user A was located in the close vicinity of the terminal B at the time she issued the request to B. Please note that the second condition in the definition does not imply that A and B have to be located within distance d of each other during the *whole* session period $[t,t+\Delta t_s)$. This last condition has to hold for only an arbitrary short, but non-null time period within $[t,t+\Delta t_s)$.

It is important to stress at this point that we are not concerned with protecting the honest party A's privacy; we focus only on the authenticity of A's transaction details. In this way we avoid potential complications related with formal definitions and treatment of confidentiality properties of our protocol. Thus, in the security proof in the Appendix, we only use the notion of *random functions* (as opposed to *random permutations*). As a result, in real practical scenarios, our Forces protocol can be implemented using regular *message authentication codes (MACs)*. If in addition the transaction privacy is required, we can replace the MAC with an appropriate *non-malleable authenticated encryption* (AE) scheme; *non-malleability* property appears to be essential for the security of our and similar distance bounding schemes.

Definition 1 is quite general as it characterizes an arbitrary secure authentication protocol (not necessarily designed with the mafia fraud attack in mind) as a mafia fraud resistant, but with potentially unfavorable parameters (d,ε). To confirm this and to show that our definition correctly captures the mafia fraud resistance property, let us consider EAP-TLS protocol [100]. For example, EAP-TLS protocol *is not* $(100m,1)$-mafia fraud resistant; that is, EAP-TLS cannot prevent relaying attacks in which the attacker relays over distance of $100m$. Indeed, nowadays EAP-TLS can be used to authenticate communicating parties in wireless LANs over distances greater than $100m$ (e.g., the distance between the given client

and the corresponding authentication server). Therefore, for $d = 100m$, EAP-TLS violates the second condition in Definition 1, that is, using the standard implementation of EAP-TLS B will provide the service to A even if they are separated by $100m$ or more apart during the session period $[t, t + \Delta t_s)$. Generally, the probability ε will depend on tolerable message timeouts in a given protocol. Thus, for example, by imposing a very short and stringent delays on EAP-TLS messages (e.g., $0.5\mu s$ or less), EAP-TLS can be made $(100m, \varepsilon)$-mafia fraud resistant, with ε being satisfactorily small (e.g., 10^{-4}). In this case, it follows from Definition 1 that the honest terminal B will provide the service only if the honest user A was located within $100m$ at the time the service request had been issued, except with the probability at most $\varepsilon = 10^{-4}$. Whether such a protocol would be of any use or not depends on the given application context.

Similarly to EAP-TLS, using Definition 1 we could, for example, classify dedicated distance-bounding protocols utilizing nano-precision clocks (such as [90]) as $(10cm, \varepsilon)$-mafia fraud resistant, with $\varepsilon \ll 10^{-4}$. Please note that depending on spatial resolution of such protocols, the attacker could still mount the attack successfully if he stays within the bounds of the error margin (e.g., within 5-10 cm due to the clock drift). In the same vein, we argue that no existing solution can prevent attacks where an adversary tampers with a legitimate terminal and tricks the user into accepting it as being a genuine one. The adversary can for example cover the door at the terminal where the money goes out and wait until the honest party successfully authenticates with the terminal. If necessary, the adversary can also install a fake display over the authentic one. *In other words, we claim that* $(0, \varepsilon)$*-mafia fraud resistant protocols, with* $\varepsilon < 1$*, are hardly attainable, at best; even by employing strategies suggested in [90].*

Therefore, the best we can hope for in terms of protection against mafia-fraud attacks is to force the adversary to be very close to both legal parties. We cannot rely on end users to verify the authenticity of the terminal. Therefore in the present chapter we consider the situations where the adversary does not try to modify in any way the legal terminal. However the adversary may install a new fake ATM/terminal in our attacker model. As the final note on the implications of Definition 1, please note that any secure authentication protocol (not necessarily designed with the mafia fraud attack in mind) is $(\infty, 0)$-mafia fraud resistant.

Next, we state the main result. We postpone its formal proof (in the random oracle model) for the Appendix.

Theorem 1. *The Forces protocol is* (d, ε)-mafia fraud resistant*, with* $\varepsilon \leq q \cdot \left[(q+1) \cdot 2^{-k} + \varepsilon_t(d, \Delta t_t) + \varepsilon_c(\Delta t_c) + \varepsilon_s \cdot \left(\varepsilon_t(d, \Delta t_t') + \varepsilon_c(\Delta t_c') \right) \right]$.

where $\varepsilon_t(\cdot)$, $\varepsilon_c(\cdot)$ and ε_s are the probability of the successful transport, cloning and simulability attack, respectively, d is the relaying distance, Δt_t and Δt_c are given/bounded by expressions (3.8) and (3.11) (Section 3.5), $\Delta t_t' < (t + \Delta t_{out} - T_{B1})$ and $\Delta t_c' < (t + \Delta t_{out} - T_{B0})$ with t being the time instant at which the protocol session begins and Δt_{out} the allowable

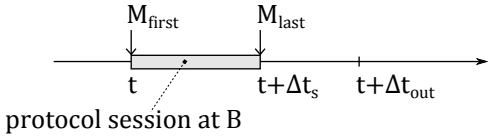


Figure 3.12. Protocol session duration at the genuine party B. The protocol session begins at the time instant t at which it receives the first/initial protocol message M_{first} and lasts until B accepts a service request from the honest party A at the time instant $(t + \Delta t_s)$ at which B receives the last message in the protocol M_{last}.

session duration, while q is the number of oracle calls made by the attacker.

Please note that $\Delta t_t < \Delta t_t^{'}$ and $\Delta t_c < \Delta t_c^{'}$; in other words, a successful simulability attack buys the attacker an extra time to perform either the transport or cloning attack. Moreover, the bounds on $\Delta t_t^{'}$ and $\Delta t_c^{'}$ are not tight, as the user has to extract the cookie, open it and have it read by her smartphone all before the session times out at $(t + \Delta t_{out})$.

3.7 Proof of Theorem 1

The timing model. We first briefly describe the timing model (Figure 3.12) used in our security proof. We assume that genuine party B (as well as genuine party A) can participate in a single session at a time. From the party B's perspective, the protocol session begins at the time instant t at which it receives the first/initial protocol message (denoted M_{first} in Figure 3.12). Recall from the protocol description, this message comprises the identity ID_A of some honest party A.

In a regular protocol run, we say that B accepts a service request from the honest party A at the time instant $(t + \Delta t_s)$ at which B receives the last message in the protocol (M_{last} in Figure 3.12) - Δt_s thus represents the current session duration. Please note that Δt_s does not account for the time it takes B to actually deliver the requested and accepted service (e.g., the time it takes to B to prepare and deliver the requested amount of money). In case of any inconsistencies or errors in the ongoing session (authentication failures, missing/incomplete messages, etc.), B does not accept and finishes the current session. The maximum session duration is bounded by Δt_{out}, i.e., $\Delta t_s \leq \Delta t_{out}$. This is the latest time at which B can accept a service request in the ongoing session.

Advantage of a mafia fraud attacker. Using Definition 1, we can characterize a successful mafia fraud attacker as follows. First, we observe that the honest party B provides the service only if the requestor's identity in the initial message that B receives belongs to a legitimate user. In practice, the requestor ID is used by B to lookup the corresponding authentication credentials. Thus it is pointless for the attacker to make requests with non-existing IDs[3].

[3]We are not concerned with possible DoS attacks here.

For a given security protocol SP and a fixed relaying distance d, the advantage $\mathbf{Adv}^{mafia}_{\mathsf{SP}}(d)$ of a mafia fraud attacker against SP is the probability that any of the following holds: *the honest party B accepts a service request with a legitimate requester identifier ID_A at some time instant $(t+\Delta t_s)$ while (1) the honest party A with identifier ID_A has not output the service request during period $[t, t+\Delta t_s)$, or (2) the requesting honest party A (with identifier ID_A) was located outside the distance d from B throughout the session period $[t, t+\Delta t_s)$.*

Therefore, for a given (d,ε)-mafia fraud resistant protocol we have $\mathbf{Adv}^{mafia}_{\mathsf{SP}}(d) = \varepsilon$.

Bounding the advantage $\mathbf{Adv}^{mafia}_{\mathsf{Forces}}(d)$ of a mafia fraud attacker. We next bound the attacker's advantage against our Forces protocol. We use *the random oracle model* for this purpose. In our protocol, $[m]_f$ denotes a pair $(m, f(m))$ where $m \in \{0,1\}^*$ and f is a random function $f : \{0,1\}^* \to \{0,1\}^k$ (k being the size of the key and random nonces); f on each different input returns a bit string uniformly distributed on $\{0,1\}^k$ (on the same input f returns the same output). In our model, genuine parties B_j (e.g., different ATMs) are all assumed to be under the control of a single trusted entity (e.g., a bank) with which each honest party A_i shares a common random function f (a unique secret K_{A_iB} in the real world); hence, each party B_j also has an access to the f oracle. Clearly, the adversary is not given access to the f oracle directly, but only through interaction with honest parties A_i and B_j. All legitimate parties are assumed to have unique identities.

The proof proceeds in three steps. We first establish that in Forces protocol the honest party B (the terminal) accepts in a given session only if the service request, i.e. the message $[B,R,m_2]_f$ in the protocol, was generated by the honest party A in the same session. Moreover, we show that A outputs $[B,R,m_2]_f$ only if it previously received $[A,B,N_A,N_B,m_1]_f$ from B in the same session. In this way we essentially establish that A and B mutually authenticate each other as well as messages m_1 and m_2 (except with a negligible probability). Using this initial result, in the second step we establish that honest parties A and B are mutually synchronized at the end of the session (except with a negligible probability). Finally, in the third step, building upon the facts that A and B are mutually authenticated and synchronized, we prove that A and B must have been located within the relaying distance d at some time during the given session (except with a satisfactorily small probability). The bound on $\mathbf{Adv}^{mafia}_{\mathsf{Forces}}(d)$ is obtained by summing up the *failure* probabilities from the three steps.

Step 1: If B accepts, then A and B have mutually authenticated each other, as well as messages m_1 and m_2. Fix A, B and a time instant t. At t, B receives the first protocol message comprising an identifier ID_A of the honest party A. The identifier must belong to the honest party, otherwise B will simply abort the protocol[4]. By the protocol, B accepts if it receives (and successfully verifies) $[B,R,m_2]_f$ at time instant $(t+\Delta t_s)$, with $\Delta t_s \leq \Delta t_{out}$. If no honest party previously output $[B,R,m_2]_f$, no party previously submitted $[B,R,m_2]$ to the f random oracle. Therefore, the probability that the attacker can compute correctly $[B,R,m_2]_f$ in this session is bounded by 2^{-k}. Consider now the case where some honest party did

[4]The identifier is used by B to lookup the corresponding authentication credentials.

output $[B,R,m_2]_f$. The form of the message $[B,R,m_2]_f$ implies that it must be honest party $A^{'}$ (not a terminal) that received R in the corresponding fortune cookie. The probability that this happened before the moment T_{B0} ($T_{B0} \in (t,t+\Delta t_s)$) at which B output (in the present session) the random nonce R (in the fortune cookie) is at most $q_1 \cdot 2^{-k}$, where q_1 is bounded by the number of calls made to the f random oracle by that time.

Next, we establish that $A^{'} = A$ by means of contradiction. Let us assume that $[B,R,m_2]_f$ was output by $A^{'} \neq A$ at some time $(t,t+\Delta t_s)$. By the protocol, $A^{'}$ must have received $[A^{'},B,N_A^{'},N_B^{'},m_1^{'}]_f$ at some earlier time. If no honest party previously output $[A^{'},B,N_A^{'},N_B^{'},m_1^{'}]_f$, no party previously submitted $[A^{'},B,N_A^{'},N_B^{'},m_1^{'}]$ to the f random oracle. Therefore, the probability that the attacker can compute correctly $[A^{'},B,N_A^{'},N_B^{'},m_1^{'}]_f$ is bounded by 2^{-k}. Consider now the case where some honest party did output it. The form of the message $[A^{'},B,N_A^{'},N_B^{'},m_1^{'}]_f$ implies that it must be honest party B (the terminal). The probability that this happened before the moment $t(N_A^{'})$ at which the honest party $A^{'}$ output the random nonce $N_A^{'}$ is at most $q_2 \cdot 2^{-k}$, where q_2 is bounded by the number of calls made to the f random oracle by that time.

Claim 1. *$t(N_A^{'}) \geq t(R)$ where $t(R) \triangleq T_{B0}$ and $t(R) \in (t,t+\Delta t_s)$.*

Proof. Let us assume by contradiction that $t(N_A^{'}) < T_{B0}$, that is $t(N_A^{'}) < t(R)$. As $A^{'}$ is the honest party, the following holds by the protocol: $t(N_B^{'}) < t(N_A^{'}) < t(R)$, where $t(N_B^{'})$ is the moment at which honest B output $N_B^{'}$. Again, by the protocol, B must have output $R^{'}$ at some earlier time $t(R^{'})$. Therefore, $t(R^{'}) < t(N_B^{'}) < t(R)$. We already established above that $A^{'}$ output $[B,R,m_2]_f$ after $t(R)$ (i.e.T_{B0}), except with $(q_1+1) \cdot 2^{-k}$. But the condition $t(R^{'}) < t(R)$ implies that $A^{'}$ could have output $[B,R,m_2]_f$ with only a negligible probability $q_1 \cdot 2^{-k}$, leading to the contradiction. We conclude that $t(N_A^{'}) \geq T_{B0}$, that is, $t(N_A^{'}) \geq t(R)$. □

We established above that the message $[A^{'},B,N_A^{'},N_B^{'},m_1^{'}]_f$ was output by honest B and after the time $t(N_A^{'})$, except with $(q_2+1) \cdot 2^{-k}$. Combining this result and Claim 1, we have $t([A^{'},B,N_A^{'},N_B^{'},m_1^{'}]_f) \geq t(N_A^{'}) \geq t(R)$. But honest B output only one message $[A,B,N_A,N_B,m_1]_f$ during $(t(R),t+\Delta t_s)$; honest B can participate in at most one session at a time, by design. Therefore, $A^{'} = A$, $N_A^{'} = N_A$ (since $[A,B,N_A,N_B,m_1]_f$ accepted by $A^{'}$), $N_B^{'} = N_B$ and $m_1^{'} = m_1$ (since $[A,B,N_A,N_B,m_1]_f$ output by B).

Summing up, the probability that honest B accepts to provide a service in a given/fixed session while no honest A requested that very service in the same session is bounded by $(q_1+q_2+2) \cdot 2^{-k}$, or more generally by $q \cdot 2^{-k}$, where q is a polynomial bound (in k) on the number of oracle calls available to the attacker (we assume $(q_1+q_2+2) \leq q$).

Step 2: At the end of a successful session A and B are mutually synchronized within the system parameter Δt_{sync}^{*}. In the proof, we assume that the local clocks of A and B are of a reasonable quality - meaning that they drift apart by only a negligible time (e.g., up to a few microseconds) during some relative short time period (in the order of tens of milliseconds). We are interested only in an instantaneous (i.e., short term) synchronization and do not care

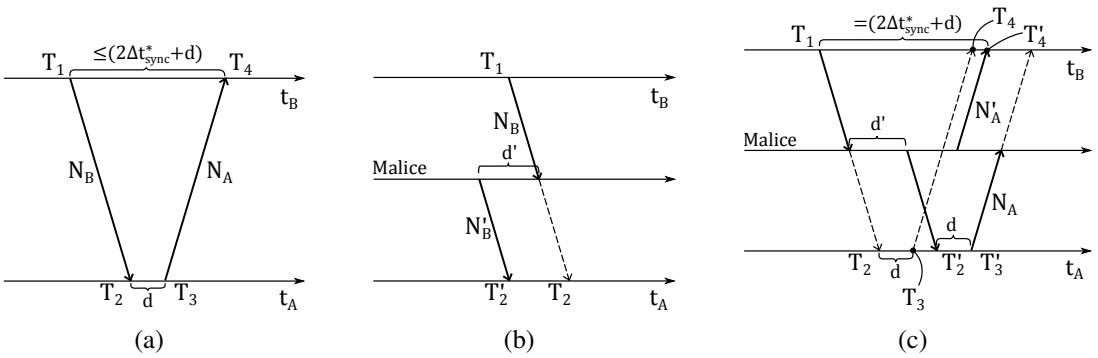


Figure 3.13. (a) The end-to-end delay Δt_{sync} is a bound on the time of flight of the nonces N_A and N_B. The attacker (Malice) has only two possibilities when attacking the synchronization protocol: he can either (b) advance or (c) delay the delivery of the random nonces N_B and N_A.

about possible long term effects of clock drifts. This assumption greatly simplifies our proof, yet it does not severely affect the security of our protocol in the real world. These small errors (measured in μs) simply add to the overall synchronization error margin (measured in *ms*).

In this step of the proof, we analyze the security of the synchronization procedure executed in **Phase II: Event synch** of our Forces protocol (see Figure 3.4), conditioned on the fact that all the messages exchanged between A and B are authentic (as established in **Step 1** above). In particular, referring to Figure 3.4, we assume that the nonces N_A and N_B as well as the timestamps T_1 and T_4 are all successfully authenticated by the honest party A.

Recall from Section 3.3, the honest party A, having received the authentic message $[A,B,N_A,N_B,m_1]_f$, with $m_1 = (T_{B1},T_1,T_4)$, verifies that the *end-to-end delay* Δt_{sync} is within the error margin Δt^*_{sync}, which is a system parameter. More precisely, A has to verify that the following holds: $\Delta t_{sync} = [(T_4 - T_1) - (T_3 - T_2)]/2 \leq \Delta t^*_{sync}$. Referring to Figure 3.13(a), we can see that Δt_{sync} is essentially the bound on the time of flight of the nonces N_A and N_B transmitted over a radio channel. Please note that we do not assume either A or B to be equipped with nano-precision clocks. The value of Δt_{sync} (and hence of Δt^*_{sync}) is essentially determined by the precision with which A and B (their operating systems) can timestamp a message transmission and reception. Please note that in line with the assumption of negligible clock drifts during the short synchronization phase, it is reasonably to assume that $\Delta t_{sync} \geq 0$. We do acknowledge that in the real world it may happen, due to anomalous drifts in the A's and B's local clocks, the end-to-end delay to be negative. In our proof, we can handle this by requiring that the honest party A runs the following verification before proceeding with the protocol: $|\Delta t_{sync}| \leq \Delta t^*_{sync}$.

Using the authentic timestamps T_1, T_2, T_3 and T_4, the party A can calculate the offset

between its clock and the B's local clock as follows[5]:

$$offset_A = T_1 + \Delta t_{sync} - T_2 \quad . \tag{3.12}$$

In turn, A can adjust its local clock t_A as follows: $t_A = t_A + offset_A$. Using the fact that $0 \leq \Delta t_{sync} \leq \Delta t^*_{sync}$ we can bound $offset_A$ as follows:

$$T_1 - T_2 \leq offset_A \leq T_1 - T_2 + \Delta t^*_{sync} \, . \tag{3.13}$$

We are now ready to prove that an attacker cannot manipulate $offset_A$ by a larger amount than Δt^*_{sync}. Observe first that the attacker has no control over the timestamp T_1 and Δt^*_{sync}; the timestamp T_1 is controlled and singed by the honest party B (see Step 1) and Δt^*_{sync} is simply a fixed system parameter. Therefore, the attacker can only manipulate the timestamp T_2 in the bounds given by (3.13). The attacker has only two possibilities as shown in Figures 3.13(b) and 3.13(c): either cause (i) a decrease or (ii) an increase of T_2. The attacker (Malice in Figures 3.13(b) and 3.13(c)) can accomplish this by either advancing or delaying the delivery of the random nonces N_B and N_A. Thus, to cause the party A to timestamp an arrival of N_B with $T_2^{'} < T_2$, the attacker has to transmit N_B before the party B has released it, see Figure 3.13(b). As the random nonces are authenticated by A, this means that the attacker has to essentially guess it, which can happen with the probability 2^{-k}.

Likewise, to cause the party A to timestamp an arrival of N_B with $T_2^{'} > T_2$, the attacker has to delay the transmission of N_B to A (e.g., the attacker jams A, receives N_B and re-transmits it with some delay $d^{'}$ as shown in Figure 3.13(c)). It follows readily from the graphical argument shown in Figure 3.13(c) that the attacker can delay the transmission of N_B by at most $d^{'} = \Delta t^*_{sync} - \Delta t_{sync}$, otherwise, the end-to-end delay Δt_{sync} will exceed the error margin Δt^*_{sync} and the party A will abort the protocol. The only way for the delay $d^{'}$ to go beyond $\Delta t^*_{sync} - \Delta t_{sync}$ without causing the protocol to fail, is to have the attacker transmit N_A before A has released it (see Figure 3.13(c)). As the random nonces are authenticated by A, this means that the attacker has to essentially guess N_A, which can happen with the probability 2^{-k}. Overall, the probability that the attacker succeeds is bounded by 2^{-k}, because a failure to guess correctly either N_B or N_A will result in the aborted protocol.

We conclude this part of the proof by observing that the parties A and B are synchronized within $|offset_A^{\text{actual}} - offset_A| \leq |\Delta t_{e2e} - \Delta t^*_{sync}| \approx \Delta t^*_{sync}$, where Δt_{e2e} is a true end-to-end delay, i.e., the time of flight of random nonces over a radio channel and $\Delta t_{e2e} \ll \Delta t^*_{sync}$.

Step 3: The requesting honest party A was located within the relaying distance d from the honest party B at some time during the session period $[t, t + \Delta t_s)$ (except with a satisfactorily small probability). We proved above that if the honest party B is to accept a request from A, B must receive back the signed challenge R. Thus, the challenge R must

[5]Here we neglect possible clock drift that potentially accumulates during Δt_{sync}; Δt_{sync} is reasonably short and in addition our protocol can tolerate synchronization errors in the order of couple of milliseconds.

be transferred (in our case by the attacker) from B over to the party A that will have it signed before sending it back to B. Before signing and sending R, A verifies that $|T_A - T_{B1}| \leq \Delta t^*_{AB}$ holds, where T_A and T_{B1} represent the moment at which the fortune cookie leaves the terminal B as recorded by the user A (her smartphone) and the terminal B, respectively. We established (in **Step 1**) that A and B are synchronized and that T_{B1} is authentic, so A can securely bound the difference $|T_A - T_{B1}|$. As explained in Section 3.3.1, T_A is extracted from the video frame that the user A has selected in the Phase II of the Forces protocol. Let us denote with T_A^{act} the *actual time* at which the cookie left the terminal. As the user may be tricked into selecting a wrong frame (or do it mistakenly), we distinguish the following two cases: (C_1) $|T_A^{\text{act}} - T_{B1}| \leq \Delta t^*_{AB}$ with $T_A^{\text{act}} \leq T_A$ and (C_2) $|T_A^{\text{act}} - T_{B1}| > \Delta t^*_{AB}$ with $T_A^{\text{act}} > T_A$. Please note that in both cases we have $|T_A - T_{B1}| \leq \Delta t^*_{AB}$ (by assumption B accepts). We assume conservatively that the second case (C_2) is always the consequence of a successful *simulability attack* against the human user (see Section 3.5.3).

Let S be the following event: $\{B \text{ accepts while } dist(A,B) > d \text{ during } [t, t+\Delta t_s)\}$. As the two cases above (C_1 and C_2) describe disjoint and exhaustive events, the following holds independently of the strategy taken by the attacker:

$$\begin{aligned} \mathbf{P}(S) &= \mathbf{P}(S|C_1)\mathbf{P}(C_1) + \mathbf{P}(S|C_2)\mathbf{P}(C_2) && (3.14)\\ &\leq \mathbf{P}(S|C_1) + \mathbf{P}(S|C_2)\mathbf{P}(C_2) && (3.15)\\ &\leq \mathbf{P}(S|C_1) + \mathbf{P}(S|C_2)\mathbf{P}(A_s)\ , && (3.16) \end{aligned}$$

with $\mathbf{P}(A_s)$ being the probability of a successful simulability attack. The last inequality follows from the assumption that the case C_2 is always the consequence of a successful simulability attack, i.e., $\mathbf{P}(A_s|C_2) = 1$. Indeed, for any two events A and B, with $\mathbf{P}(A|B) = 1$, we have $\mathbf{P}(B) = \mathbf{P}(B|A)\mathbf{P}(A)$ from which $\mathbf{P}(B) \leq \mathbf{P}(A)$.

Next we bound the probabilities $\mathbf{P}(S|C_1)$ and $\mathbf{P}(S|C_2)$. We already established that the challenge R must be transferred by the attacker from B over to the party A. We argued in Section 3.4, that the attacker has only two basic possibilities: either (i) to transfer the original fortune cookie (with R in it) between B and A - the *transport attack* (A_t), or (ii) to somehow read the challenge R from the original cookie and clone it (create a fake cookie) at the A's side - the *cloning attack* (A_c). Therefore, $\mathbf{P}(S|C_i) = \mathbf{P}(A_t \vee A_c|C_i) \leq \mathbf{P}(A_t|C_i) + \mathbf{P}(A_c|C_i)$. By plugging in this bound into the above bound on $\mathbf{P}(S)$ we obtain:

$$\begin{aligned} \mathbf{P}(S) &\leq \mathbf{P}(A_t|C_1) + \mathbf{P}(A_c|C_1) + \mathbf{P}(A_s)\left(\mathbf{P}(A_t|C_2) + \mathbf{P}(A_c|C_2)\right) && (3.17)\\ &= \varepsilon_t(d, \Delta t_t) + \varepsilon_c(\Delta t_c) + \varepsilon_s \cdot \left(\varepsilon_t(d, \Delta t_t') + \varepsilon_c(\Delta t_c')\right)\ , && (3.18) \end{aligned}$$

where $\varepsilon_t(\cdot)$, $\varepsilon_c(\cdot)$ and $\varepsilon_s(\cdot)$ are the probability of the successful transport, cloning and simulability attack, respectively, d is the relaying distance, Δt_t and Δt_c are bounded as shown by the expressions (3.8) and (3.11) (Section 3.5), $\Delta t_t' < (t + \Delta t_{out} - T_{B1})$ and $\Delta t_c' < (t + \Delta t_{out} - T_{B0})$.

(a) (b)

Figure 3.14. (a) The experimental setup used in our user performance study, and (b) an example of a sequence of frames presented to the user in the camera-based experiment.

Please note that $\Delta t_t < \Delta t_t^{'}$ and $\Delta t_c < \Delta t_c^{'}$; in other words, a successful simulability attack buys the attacker an extra time to perform either the transport or cloning attack. Moreover, the bounds on $\Delta t_t^{'}$ and $\Delta t_c^{'}$ are not tight, as the user A has to extract the cookie, open it and have it read by her smartphone all before the session times out at $(t + \Delta t_{out})$.

To conclude the proof for this single fixed session, we simply have to sum up the bounds from the three steps above. Finally, the bound on the attackers advantage $\mathbf{Adv}^{mafia}_{\mathsf{Forces}}(d)$ is obtained by observing that the attacker is allowed to make at most q oracle calls, which is also the number of possible protocol sessions. Therefore:

$$\mathbf{Adv}^{mafia}_{\mathsf{Forces}}(d) \leq q \cdot \left[(q+1) \cdot 2^{-k} + \varepsilon_t(d, \Delta t_t) + \varepsilon_c(\Delta t_c) + \varepsilon_s \cdot \left(\varepsilon_t(d, \Delta t_t^{'}) + \varepsilon_c(\Delta t_c^{'}) \right) \right] . \quad (3.19)$$

3.8 User Performance Study

We implemented a simple simulator to evaluate different phases of the Forces protocol (Figure 3.4 and Figure 3.6 - for the camera-based event synchronization). Our goal was to estimate the cost of each protocol phase in terms of time (and effort) required to execute them by the end user. We implemented the Forces protocol (Figure 3.4) with two alternative event-synchronization methods: (i) the *pushbutton* method (Section 3.3.2) and (ii) the *camera-based* method (Section 3.3.1).

Our simulator (Figure 3.14(a)) consists of a part representing an ATM terminal and a smartphone device (Nokia 5800). The terminal was emulated using a 22" monitor and an optical mouse both attached to HP 6730b laptop. The optical mouse was used in the im-

	Age				Use QR codes		Use m-banking		Use ATM	
	18-25	26-29	30-34	>35	Y	N	Y	N	Y	N
Pushbutton solution	9	3	2	0	7	7	2	12	11	3
Camera solution	10	6	2	2	16	4	4	16	16	4
Simulability attack	19	1	0	0	13	7	2	18	15	5

Table 3.1. Summary of the users' demographics.

provised paper/cookie dispenser, for the purpose of precise measurement of the duration of the paper extraction process (i.e., to precisely determine the time instants T_{B0} and T_{B1}, as seen Figure 3.7). This was accomplished by simply turning the mouse upside down, then fixing it in the improvised paper dispenser, and finally by sliding the paper cookie over the mouse. In most of our tests, the testers extracted the same (2 times folded) fortune cookie of the fixed size - 10 cm by 5 cm. As explained later, in a part of conducted experiments we used a slightly modified fortune cookie. The fortune cookie contained 2 cm by 2 cm QR code. In each test (except in special scenarios explained later) the 10 by 5 cm fortune cookie was placed inside the improvised cookie dispenser, 8 cm inside the outer edge of the dispenser and 2 cm outside the edge. We used Bluetooth wireless technology as a primary communication channel between the simulated terminal and the corresponding smartphone. Recall, our solution works at the application layer and as such is independent on the type of the underlaying communication technology. During the tests, we switched off the Bluetooth pairing protocol since in any case we consider the primary channel to be insecure (under the attacker's control). The software services used in our simulator were implemented in Java: Java SE on the terminal's side and Java ME for the smartphone services.

A total of 54 participants took part in the study: 14 (14 males) in the *pushbutton* study, 20 (4 females and 16 males) in the *camera* study. We tested the robustness of the pushbutton method against the simulability attack with 20 users (4 females and 16 males). The tests were voluntary and the testers were recruited among the students and staff of our university. Table 3.1 summarizes the users' age, experience in the use of QR codes, m-banking and ATM terminals. Before proceeding to each test, the users were briefly introduced to the concept of mafia-fraud attacks and explained how our protocol is secure against them. Each step of the protocol was explained to them. This was followed by the practical demonstration from the administrator. For each participant, we logged the elapsed time for each protocol phase, as well as the overall error rate. Although our university does not require the ethical review board to review and approve research work with the human testers, all the testers in our study were informed in advance how the collected data of their study will be processed after the evaluation.

During the evaluation of both the *pushbutton* and *camera*-based event-synch methods, the users were asked to successfully run the Forces protocol for five times. Likewise, in the study related to the simulability attack, the testers first executed 10 regular (no attack)

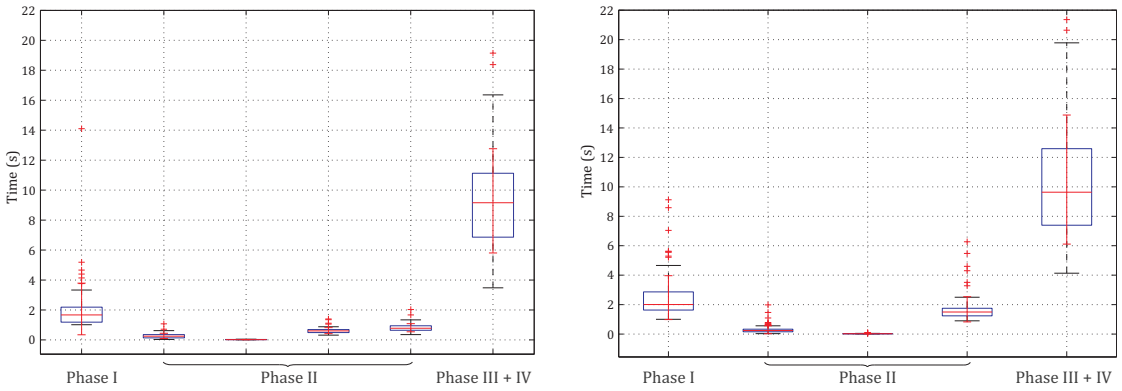


Figure 3.15. Box plots representing the average protocol execution time for each phase: (a) the pushbutton*-based event-synchronization (Phase I - Initialization, Phase II - Cookie extraction, clock synchronization, push on the button after cookie extraction, pushing the button twice (for error prevention), Phase III - Manual opening of the cookie and QR code detection, and Phase IV - transmission of the last protocol message (b) the* camera-based *event-synchronization (differs only in Phase II - Cookie extraction, clock synchronization, selection of the event-synchronization frame).*

protocol runs and then one additional run in which the simulability attack has taken place. There were no economical incentives provided to the testers in order to achieve smaller error rates and faster protocol execution times. Before the beginning of the test, the users were asked to complete a pretest questionnaire that summarized users' demographic and computer/mobile knowledge background. Each study took about 15 minutes per user. In the sequel, we present the obtained results from this performance evaluation study.

3.8.1 Results of the Pushbutton Study

Recall from Section 3.3.1, after the fortune cookie extraction, the smartphone finds the moment T_A by having the user push the button once she observes that the fortune cookie completely left the terminal dispenser. To avoid the situations in which the user would accidently push the synch button on the smartphone, we asked the testers to make 3 consecutive button pushes - serving as the error prevention. Figure 3.15(a) shows the average execution times taken by each phase of the Forces protocol. The average overall/cumulative protocol execution time is 13.08 seconds (with std $=$ 4.09 sec). Please note that here we do not consider the time it takes for the user to log into the smartphone and prepare the desired transaction. As shown in Figure 3.15(a), the major part of the protocol execution time falls on the Phase III: the paper unfolding, QR code detection and secret nonce verification (average time 9.28 sec with std. $=$ 3.47 sec). The average times taken to complete the initialization, pulling out the paper, clock synchronization and pushing of the button were 2.06 sec (std. $=$ 1.72 sec), 0.271 sec (std. $=$ 0.17 sec), 22.65 ms (std. $=$ 10.66 ms) and 0.62 sec (std $=$ 0.201 sec), re-

spectively. It turned out that the slowest part of the protocol, the one that induced the largest delay, was the QR code detection and reading. This is actually the good news, since modern smarphones have better QR code processing capabilities and therefore make this delay less of an issue.

Error rates. Recall, in the pushbutton event-synchronization method, a push of the appropriate button by the given user, on her smartphone, confirms that the fortune cookie is completely extracted/detached from the terminal. In our tests, two testers pressed the synch button (not by chance as we require 3 pushes in a row) while the corresponding cookie was still partly inside the terminal. As discussed earlier, by pushing the synch button too early (while the cookie is still inside the terminal), the attacker can potentially gain more time to successfully mount the relay attack. Although the observed error rate was small (only 2 fatal errors in 70 protocol runs), it motivated us to run another series of tests.

Our goal was to understand to what extent we can rely on users' alertness when it comes to detecting such simulability attacks. Therefore we conducted the test in which we tried to trick the testers into pushing the button while the cookie was still inside the terminal. To accomplish this we asked each tester to run the protocol 10 times in a row in normal conditions (i.e., with regular-sized cookie (10 cm) and no attacks). In the final run we replaced the original 10 cm cookie with a longer one (around 30 cm); of course, the testers were not aware of this. As a result we were able to trick 5 out of 20 testers (25%) in this way, proving that the pushbutton event-synch method is not robust enough. Therefore, we developed a more reliable and robust event-synch method based on the smartphone camera; in the following subsection we report on the related study.

3.8.2 Test Results of the Camera Study

Recall from Section 3.3.1, a camera-based event synchronization is accomplished by showing the user a sequence of recorded frames on the smartphone as shown in Figure 3.5(a). In turn, the user has to simply select the first frame on which she sees the fortune cookie is detached from the terminal. Because each frame is timestamped relatively to the beginning of Phase II, by selecting the appropriate frame the smartphone can learn the moment T_A at which the cookie left the terminal. In our implementation of this event-synch method, the testers were presented with a randomly selected sequence of frames showing the recorded cookie extraction process (these were previously recorded and inserted into the smartphone); see Figure 3.14(b) for one such sequence.

Figure 3.15(b) shows the average execution times taken by each phase of the Forces protocol. The overall time it took the testers to complete the protocol was 13.28 sec on average (std. $= 4.73$ sec). Similarly to the pushbutton study, here too the major part of the induced delay is due to the paper unfolding and QR code detection and decoding (the average time was 8.81 sec with std $= 4.41$ sec). The average time taken to complete the initialization,

pulling out the fortune cookie, clock synchronization and the frame selection were 2.47 sec (std. = 1.48 sec), 0.293 sec (std. = 0.27 sec), 24.48 ms (std. = 14.18 ms) and 1.68 sec (std = 0.85 sec), respectively.

Error rates. The most important result obtained from this study is that all the testers executed the protocol without any errors. More precisely, all the testers correctly selected the first frame in which one could observe a completely extracted cookie/paper. This result is important as it shows that the only part of the protocol that relies on the user's alertness can be made robust against human errors at a reasonably low additional cost (the users have to record the extraction process). We accept that no error might be due to the participants' age and computer/mobile devices background. Therefore, our study presents preliminary results and in future work we plan to extend our evaluation on a larger and elderly population.

3.9 Related work

A classic example of the relay attack was first presented in the work by Conway [101], explaining how a player who does not know the chess rules could play against two grandmasters. By subsequently relaying the moves of one grandmaster to the other, the player could win against one of them or at least draw against both. The first use of relay attack in security protocols was introduced by Desmedt et. al. [102], and referred as "mafia-fraud" attack. Some other similar attacks have been proposed, named as "wormhole attacks" [103], "terrorist attack" [104] and "distance hijacking attack" [105]. Practical implementation of mafia fraud attack on Chip and PIN system was proposed by Drimer and Murdoch [85]. Anderson and Bond [106] suggest a prevention mechanism that uses attorney device (a trusted device) which informs the user about the payment amount. However, these attacks are not secure in scenarios where the attacker relays the expected request (e.g. a withdrawn money) as Francis et. al. [87] demonstrate with a NFC peer-to-peer relay attack on smartphones. In particular, the relay attack allows the adversary to circumvent the higher application layer cryptography simply by forwarding the message from the genuine source to the actual destination, without any knowledge of encryption algorithms and shared key.

One of the first solutions against relay attacks was presented by Brands and Chaum [107], called *distance-bounding protocols*, that involves two parties, a prover and a verifier. A number of distance bounding protocols were proposed since [85, 108, 109, 110, 111], and most of them are based on Round-Trip-Time (RTT) calculation of cryptographic protocols (time of flight). As the result, these protocols place an upper bound on the physical distance from the prover (e.g. a sphere around the prover). The proposed protocols can estimate the physical distance between the prover and the verifier based on the fact that the signal cannot travel faster than light. Indeed, in a recent paper Rasmusen and Čapkun proposed realization of RF distance bounding protocol where the malicious prover can pretend to be at most 15 cm closer to the verifier [90]. However, the protocol requires specialized

hardware that samples the received data at higher rates. To achieve spatial resolution using a simple equipment, some solutions suggest distance bounding protocols based on ultrasound channel [112]. However, ultrasound is not secure against the relay attack, since the attacker can easily forward the message over a faster channel (e.g. a radio channel) and thus extend the distance of the genuine prover to the verifier.

In recent work [1] Stajano et. al. propose another solution to counter relay attacks, which is based on multichannel protocols. They suggest augmenting the channels normally used for the transaction with an additional, specific channel that attackers will not be able to relay (e.g. a banknote). Stajano et. al. conclude their work by stating that more robust and practical implementation of unrelayable channels are needed.

3.10 Summary

Future NFC technologies look promising as they could move payments from the existing credit cards to smartphone devices. Yet, all these (and similar) technologies have a problem in common, they are all vulnerable to a tricky form of a relay attack - *mafia fraud attack*. Inspired by the work of Stajano et. al. [1] on unrelayable channels we showed how a piece of paper (appropriately folded - hence *fortune cookie*) in combination with omnipresent smartphone devices can implement a strong protection against mafia-fraud attacks (a *weakly unrelayable channel*).

Our weakly unrelayable channel has a highly practical application against relay attack in the context of financial transaction that involve (paper) receipts such as ATM money withdrawal and payment terminals. Our solution is independent of the underlying contactless technology (e.g., being NFC, WiFi, Bluetooth, Infrared) or contact-based technology (e.g., intra-body communication). The proposed solution is suitable for authorization of a high-risk financial transactions in an unknown environment (e.g., withdrawal of a larger amount of money from a given terminal in a foreign country). Moreover, we showed that in some realistic attacking scenarios, even an unfolded paper (fortune cookie) may provide a good protection against relay attacks. This makes our solution highly practical and user-friendly.

The results of user performance study indicate that the proposed solution has reasonably low execution time (around 13 seconds on average), minimal error rate and is easy to understand. In our future work we also plan to investigate the usability of an accelerometer based solution for the paper detection with the smartphone.

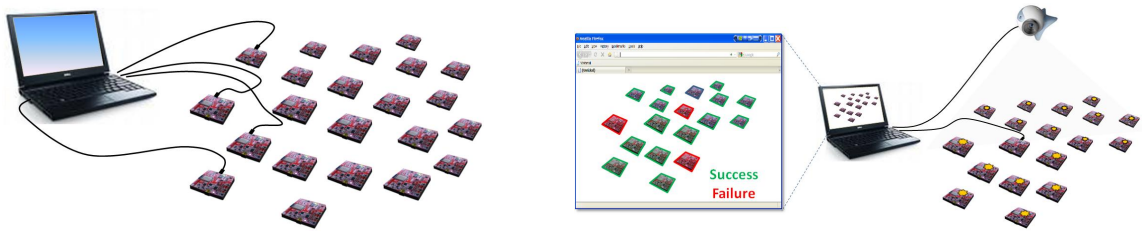
4 BOOTSTRAPPING MULTIPLE CONSTRAINED WIRELESS DEVICES: SCALABLE AND USER-FRIENDLY KEY DEPLOYMENT

4.1 Introduction

Wireless Sensor Networks (WSN) are increasingly gaining momentum in our lives. Tomorrow's e-healthcare systems, smart homes, power management systems will involve a large number of inter-connected smart wireless (sensor) devices that will be operated and controlled by end users (a home user or an administrator). Indeed, Ericsson forecasts that there will be 50 billion internet-connected devices by 2020 - apart from mobile phones and laptops, Ericsson (Chief Executive Hans Vestberg) expects other kinds of machines (e.g. DVD players, fridges, traffic systems, vending machines) to be linked through internet connection.

The aforementioned smart (wireless) devices have the capability to connect and interact, and provide a backbone for the future development of the "Machine-to-machine" systems and services. In a WSN environment, the nodes might need to communicate security sensitive data among themselves and with the base station (also referred to as "sink"). The communication among the nodes might be point-to-point and/or broadcast, depending upon the application. These communication channels, however, are easy to eavesdrop on and to manipulate, raising the very real threat of the so-called man-in-the-middle attacker. A fundamental task, therefore, is to secure these communication channels.

In this chapter we introduce to security solutions that scale well with the number of WSN devices to be handled by an individual. We designed and implemented secure, scalable, fast and user-friendly methods for bootstrapping and securing WSN networks. In this context the term bootstrapping refers to the process of establishing initial security associations between different WSN-enabled devices. This is a critical phase in the process of setting up the WSN network; the failure to securely load the WSN devices with required keying material may render the whole network insecure. According to the 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) RFC4919 specification [113], one of the major security considerations involves application of out-of-band techniques for the initial key establishment among a large number of WSN devices. Experience with existing systems and networks (e.g.,WiFi) have taught us that complicated setup procedures render the security



(a) (b)

Figure 4.1. Bootstrapping large-scale sensor networks: (a) the standard non-scalable approach vs (b) a scalable and easy-to-use solution.

features useless (users easily give up due to complex procedures, and thus leave their networks unprotected); this could be a major deterrent to wider adoption of such a technology.

Motivation for Secure Initialization

A number of so-called "key pre-distribution" techniques to bootstrap secure communication in a WSN have been proposed, e.g., [114, 115, 116, 117, 118]. However, all of these techniques assume that, before deployment, WSN sensor nodes are somehow pre-installed with secret(s) shared with other sensor nodes and/or the sink. The TinySec architecture [119] also assumes that the nodes are loaded with shared keys prior to deployment. This might be a reasonable assumption in some, but certainly not all, cases. Let us consider, for example, a user-centric WSN application. An individual user (Bob) wants to install a sensor network to monitor the perimeter of his property; he purchases a set of commodity noise and vibration sensor nodes at certain retailers, and wants to deploy the sensor nodes with his home computer acting as the sink. Being off-the-shelf, these sensor nodes are not sold with any built-in secrets. Some types of sensor nodes might have a USB (or similar) connector that allows Bob to plug each sensor node into his computer to perform secure initialization. This would be immune to both eavesdropping and man-in-the-middle attacks. However, most sensor nodes might not have any wired interfaces, since having a special "initialization" interface influences the complexity and the cost of the sensor node. Also, note that Bob would have to perform security initialization manually and separately for each sensor node (Figure 4.1(a)). This undermines the scalability of the approach since potentially a reasonably large number of sensor nodes might be involved.

Furthermore, keys can not always be pre-loaded during the manufacturing phase because eventual customers might not trust the manufacturer, for example in wireless nodes deployed for military applications. Moreover, a WSN application might involve nodes produced by multiple manufacturers. Due to this reason, establishing pre-shared secrets or a PKI-based solution might be infeasible as it would require a global infrastructure involving many diverse manufacturers. We note that the problem of secure WSN initialization that we consider in

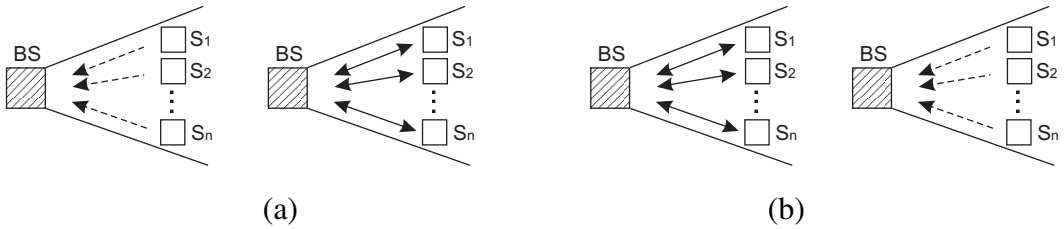


Figure 4.2. Two phases of node initialization for (a) secret key and (b) public key deployment protocol. In (a) nodes transmit the key to the base station via VLC (dashed arrows) and perform authentication via a radio channel (full line arrows), while in (b) they exchange public keys over a radio channel and perform authentication via VLC.

this thesis is very similar to the well-studied problem of "wireless (two-device) pairing", the premise of which is also based on the fact that the devices wanting to communicate with each other do not share any pre-shared secrets or a common PKI with each other [120, 121].

In this chapter, we propose two novel *multichannel* protocols for initialization of large scale wireless sensor networks (Figure 4.1(b)). Similar to [8], our protocols involve communication over a radio channel and an out-of-band visible light channel (VLC). The first protocol uses only secret key cryptography and is suitable for CPU-constrained sensor nodes. The "secret key"-based initialization of sensor nodes is depicted in Figure 4.2(a). In this protocol, each sensor node establishes a unique secret key with a *base station (BS)*. The base station comprises of a simple web camera and one sensor node attached to an ordinary PC. In the first phase of the protocol, the sensor nodes transmit secret keys to the base station over a *protected* visible light channel (Figure 4.2(a)). In the second phase, each sensor node runs a key verification protocol with the base station over a bidirectional radio channel. Once the keys are verified, the base station can serve as a *trusted third party* and mediate establishment of security associations between any pair or any group of sensor nodes.

Our second protocol uses public key cryptography. The "public key"-based sensor node initialization process is summarized in Figure 4.2(b). As with the previous protocol, the ultimate goal is to establish security association between each sensor node and the base station. This protocol is based on the multichannel pairing protocol from [6, 5]. Thus, each sensor node first exchanges its public key (through specially formed commitment/openning pairs) with the base station over the radio channel (Figure 4.2(b) - left). In turn, each sensor node transmits a *short authentication string* (`SAS`) using the visible light channel (Figure 4.2(b) - right). The proposed "public key"-based protocol is similar to [8], with the difference that our protocol is designed to be secure in a very strong attacker model, where an attacker can eavesdrop, jam and modify transmitted messages by adding his own message to both a radio and a visible light channel; the attacker however cannot disable the visible light communication channel[1].

[1]It was brought to our attention recently that a similar approach has been suggested by [122]. This model

The chapter is organized as follows: in Section 4.2 we state the problem and assumptions. In Sections 4.3 and 4.4 we describe the "secret key"- and "public key"-based protocols (including security analysis of both protocols). We present the implementation of the protocols and of a random number generator in Section 4.5. Related work is given in Section 4.6. Finally, we summarize the chapter in Section 4.7.

4.2 Problem Statement and System Model

We consider the following problem: *How to securely initialize a large number of sensor nodes in an user-friendly fashion?* As the initialization will be performed by potentially non-expert personnel, a solution has to be easy both to use and learn (*user-friendliness*). In addition, the hardware cost per node has to be minimized (*cost-efficiency*).

4.2.1 System Model

We assume that a user is equipped with a base station used for verification and monitoring as shown in Figure 4.3.

Base Station. The base station comprises a monitor, a simple web camera and one sensor node (a verification node) attached to an ordinary PC. The verification node serves as a radio modem to the base station.

Uninitialized Sensor Nodes. Nodes may be equipped with a single LED (we used two LEDs in our implementation) used for key transmission via out-of-band VLC and with radio transceivers. In addition, each node has a "pushbutton" used to either restart or finalize the initialization process.

Cardboard box. A simple cardboard box is used to block the escape of light during the key transmission via VLC. The cardboard box is required only for the "secret key" - based key deployment protocol.

4.2.2 Attacker Model

An adversary has full control over the radio channel. He/she can eavesdrop, drop, delay, replay and modify messages sent via radio. Thus, he/she is able to initiate the communication with any device (node or base station) and at any given time during the key transmission. Furthermore, the adversary can install his/her own web camera in the same place where the initialization is taking place. We assume that devices involved in key deployment (PC and nodes themselves) are not compromised. Taking into account these constraints, we define: (a) *a passive adversary* who only observes the visual channel and can eventually record a secret key if the key transmission takes place in insecure conditions (outside the cardboard

is developed for a weaker attacker model (VLC is considered as an authentic).

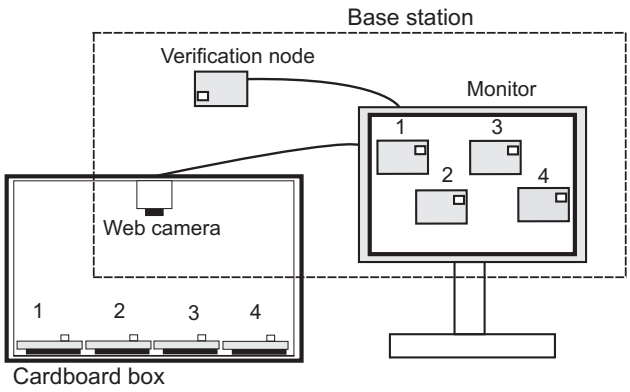


Figure 4.3. Secret key deployment setup comprises a base station and a simple cardboard box.

box), and (b) *an active adversary* who in addition can initiate communication with any device during the initialization phase.

In the case of “public key" - based initialization, we consider a stronger adversary model where an attacker can eavesdrop and modify messages sent over a light channel at all times (we elaborate this in Section 4.4).

4.3 Secret Key Deployment

In this section we propose secret key based key deployment protocol and we also provide initial security assessment for the proposed key deployment method.

4.3.1 Key Transmission And Verification

Prior to the start of node initialization, the user connects both the web camera and the base station to the PC. Next, the user places the web camera on top of the box from the inside, as shown in Figure 4.3. At this stage, the user powers the nodes ON and places them inside the box. Next, the user closes the box, runs the program on the PC and initiates the node initialization procedure. The box remains closed until the key transmission and verification is performed on all nodes which is subsequently indicated on the monitor.

Key Transmission. Our secret key based (Figure 4.4) deployment is based on ISO/IEV 9798-2 [123] three pass key authentication protocol. We modify this protocol to include the communication over VLC (dashed arrow in Figure 4.4). The modified protocol evolves as follows.

The node S_i generates n-bit random key K_{S_iB} and k-bit random string N_{S_i}. The base station generates k-bit random string N_{B_i}. The node, equipped with minimally one LED, sends the key via VLC (step 1) to the base station (web camera), as shown in Figure 4.4. At the same time, the base station performs three tasks: (i) collects keys K_{S_iB} generated by the nodes (step

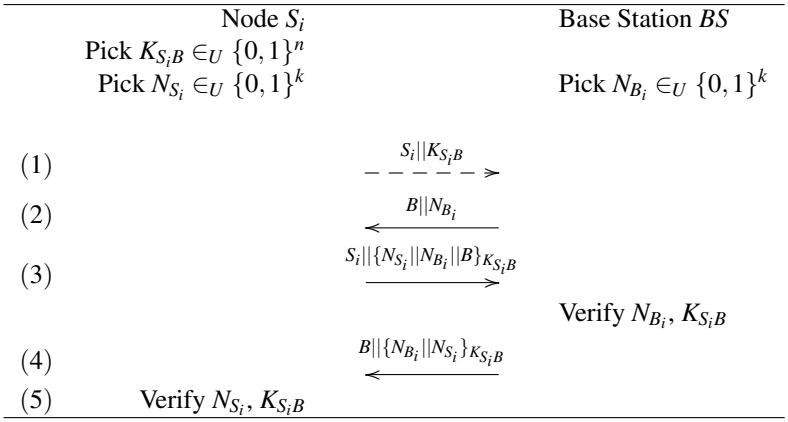


Figure 4.4. Modification of ISO/IEV 9798-2 three pass key authentication protocol. The dashed arrow represents key transmission over secure VLC.

1), (ii) initiates key verification over radio channel (steps 2-4), and finally, (iii) notifies the user which node has been successfully initialized via monitor (step 5). Section 4.5 provides details of the key transfer over VLC.

Key verification. After the key is transmitted over VLC, the base station initiates the key verification protocol. All messages in the key verification are exchanged over the radio channel. The base station (using a verification node) sends random nonce N_{B_i} over the radio channel to node S_i (step 2). Next, S_i forms a packet by encrypting concatenations $N_{S_i}||N_{B_i}||B$ with the key K_{S_iB}. The node S_i sends this message (and its identity) to base station (step 3). The base station extracts the random nonce N_{S_i}, verifies the key K_{S_iB} and the random nonce N_{B_i}. If the verification is successful, the BS encrypts concatenation $N_{B_i}||N_{S_i}$ using K_{S_iB} and sends it back to node S_i. The node S_i receives and verifies both the key K_{S_iB} and the random nonce N_{S_i}. The whole procedure is considered as completed if all the nodes are successfully initialized, which is finally indicated by the GUI on the monitor. At the end, the user opens the box and completes the initialization with the short push on the node's button. This feature is used to ensure the "proof of presence" property to prevent an active attack (as described in Section 4.3.3).

4.3.2 Sensor Node State Diagram

Both user and base station need to know the status of the initialization process at any given time. For that reason, the current state of the node will be indicated with a LED according to the state diagram shown in Figure 4.5. During the initialization process, the node can take one of the four following states: Uninitialized, Ready, Initialized and Confirmed. Next, we describe each of these states separately as well as the transitions between them.

Uninitialized state. Initially, when the user powers the node ON it is in the Uninitialized state. Prior to the start of key transmission the LED blinks with a predefined frequency.

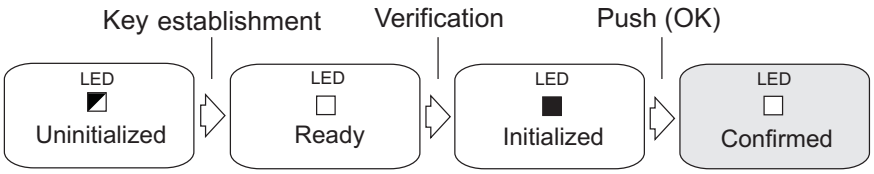


Figure 4.5. Node's state diagram. A colored square indicates that the LED is ON, while a half colored that the LED is blinking.

During this phase the node generates the key and, upon completion, sends it via VLC to the camera. After the key transmission is complete, the node advances to the Ready state (step 1 in Figure 4.4).

Ready state. The node remains in this state for a predefined period of time (e.g., a few seconds). In this state the node has sent the key and awaits the base station to initiate the key verification protocol over a radio channel. During this phase the node's LED is OFF (Figure 4.5). If the node does not receive any messages from the base station within the predefined period of time, it automatically restarts and returns back to the Uninitialized state. The node repeats the whole procedure which involves new key generation and transmission over VLC. Alternatively, the node receives a message from the base station and starts the key verification (steps 2-4 in Figure 4.4). If the key verification is successful, the node advances to the Initialized state.

Initialized state. In this state the node's LED is powered ON (Figure 4.5). At the same time, the base station notifies the user via monitor about the node's position within the box as well as its current state. If the key verification succeeded on both sides (the node's and the base station's), the user is instructed to remove the nodes from the box and to shortly push the button on the node to finalize the initialization. The push of the button serves as "proof of presence", an aspect we describe in Section 4.3.3. However, if the node or the base station failed to verify the key, the user is instructed over the monitor to restart the initialization on selected nodes with a longer push on the button in order to repeat the node initialization. After the short push of the button, the node advances to the Confirmed state.

Confirmed state. In this state the node and the base station established a secret and authenticated key and the initialization process is finalized.

4.3.3 Security Analysis

In this section, we assess the security of the secret key deployment protocol. Our protocol is based on ISO/IEV 9798-2 [123] three pass key authentication protocol which was proven to be secure when used over a radio channel. Therefore, we focus on possible attacks over the VLC, as we extended the ISO/IEV 9798-2 [123] protocol by including a transmission of a secret key via the VLC.

Camera Recording (Passive) Attacker. A camera-recording attacker will attempt to learn the secret key simply by recording the key sent from the node via VLC (step 1 in Figure 4.4). In this model the attacker is passive and does not interact in any way with the node initialization procedure.

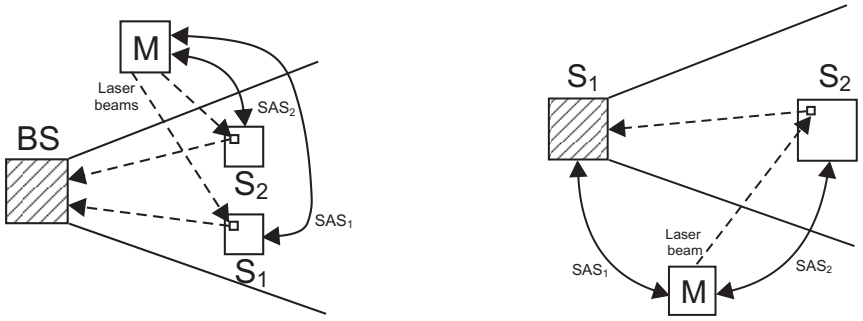
Let us consider the case in which the node starts sending the key under insecure conditions (e.g., outside of the box, while the camera setup is still not ready, etc.). Thus, the attacker records the key, and the node advances from Uninitialized to Ready state (node's LED powers ON). In this state, the node waits a predefined period of time for the base station to initiate the key authentication (Figure 4.5). After the predefined time period has passed during which the base station didn't initiate the key authentication protocol, the node returns back to the Uninitialized state and repeats the whole procedure again (generates a new key and, again, sends it via VLC). The base station receives notification from the user that the system is ready (operates in secure conditions). Only then will the base station begin to process keys received over VLC and initiate the key verification protocol.

Active attacker. In this attacker model, the attacker controls both the radio channel and communication over VLC when sensor node(s) are out of the cardboard box. Let us assume that the attacker captures the key sent by a node via VLC under insecure conditions (e.g., the node outside of the box). At this stage, the node is in Ready state and awaits the base station to initiate key verification (Figure 4.5). Next, the attacker initiates the key verification phase over the radio channel using the captured key. If the verification is successful on the node's side, the node advances to the Initialized state (the LED powers ON as shown in Figure 4.5). In this state the node waits for the user to confirm the initialization (push on a button). The user doesn't know that the attacker placed the node in the Initialized state so she picks the node up, and places it inside the box. Once the compromised node is placed inside the box, the base station recognizes a constantly powered ON LED on it and warns the user (via the monitor) to restart the initialization of that node. This is done by a longer push on the node's button. This form of active attacks does not work as the attacker does not have physical access to the node, therefore he cannot force the node to advance to Confirmed state. The user basically "proves his presence" by a push on the button.

4.4 Public Key Deployment

In this section we extend the attacker model to a more powerful adversary who can observe the electromagnetic radiation emanating from the LEDs. We assume the LEDs emanate radio signals which cannot be blocked by a simple cardboard box and we also assume that the attacker is able to easily eavesdrop on the leaked signals. This is a variant of an attack previously introduced in [124].

To establish keys between nodes and the base station by using a bidirectional radio channel and an unidirectional out-of-band VLC, we use SAS protocols [5, 6]. The protocols make



(a) (b)

Figure 4.6. An attack in which the attacker M, with the aid of a laser, tries to modify the SAS *established between the devices S_1 and S_2 sent via VLC to the (a) base station BS or (b) device S_1; dashed arrows and full-line arcs represent communication over a semi-authentic VLC and a radio channel, respectively.*

the key exchange process more usable, but at the cost of having to introduce public key cryptography. Recent work on elliptic curve cryptography has shown promising results regarding key distribution on resource constrained devices like our sensor nodes. In NanoECC [125] and TinyPBC [126] times less than 2 and 1 seconds for point multiplication in binary fields were achieved, respectively.

Many prominent solutions that use LEDs and cameras [12, 8] assume that the Visible Light Channel is authentic, which is not the case in our attacker model. To convey information via VLC they use on-off keying (switch the LED ON or OFF). An attacker equipped with a directional light source (e.g. a laser) has the capability to modify a message sent via VLC. In our model the attacker can modify messages by flipping $0 \rightarrow 1$, but not vice versa ($1 \rightarrow 0$) as the attacker cannot force a switched ON LED to turn OFF. In this case we speak of a *semi-authentic* visible light channel.

In the following sections we describe how to perpetrate such attacks and we also propose solutions on how to protect against them.

4.4.1 Attacks on Visible Light Channel

We consider prominent device pairing methods proposed in [12] and [8]. Both of the methods were developed for an authentic VLC (an attacker cannot modify messages sent via VLC). The proposed methods are secure within the authentic VLC model but, as we will show, are insecure in our semi-authentic VLC model (an attacker can flip $0 \rightarrow 1$).

Protocol [12] in the semi-authentic model. In [12] two devices (S_1 and S_2 as shown in Figure 4.6(a)) exchange public key values via a radio channel using the SAS protocol [5, 6]. To authenticate these messages, each device simultaneously transmits short authentication strings (SAS) using visible light. The camera (*BS* in Figure 4.6(a)) captures both of these

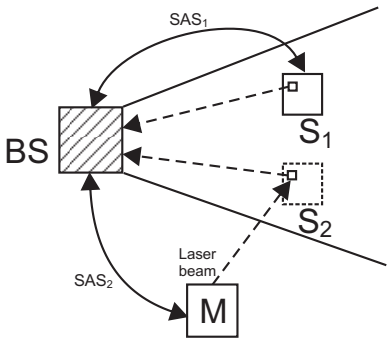


Figure 4.7. An example of the virtual node attack; dashed arrows and full-line arcs represent communication over a semi-authentic VLC and a radio channel, respectively.

authentication strings and compares them. As *BS* does not know the SAS beforehand, the attacker can mount a MITM attack and modify these strings with a laser. Attacker M exchanges public keys with two devices S_1 and S_2 via a radio channel. When transmission via VLC occurs, the attacker points the lasers into the nodes' LEDs and appropriately modifies the bits (flips $0 \rightarrow 1$). The simplest attack is the one in which the attacker flips all bits $0 \rightarrow 1$. In this case, the base station will see all 1s and inform the user about the correct authentication. Please note that all 1s is a legitimate SAS.

Protocol [8] in the semi-authentic model. In an approach similar to [12], two devices (S_1 and S_2 in Figure 4.6(b)) exchange public key values over a radio channel. In the presented scheme, at least one device has an integrated web camera. In order to verify the exchanged public key values, device S_2 sends the SAS via VLC (using LEDs) to the device S_1. Here, an attacker tries to mount a MITM attack by exchanging different public keys with devices S_1 and S_2, (Figure 4.6(b)). To succeed, the attacker has to ensure that $\mathtt{SAS}_1 = \mathtt{SAS}_2$. Due to the property of the protocols [5] in which the probability for $\mathtt{SAS}_1$ and $\mathtt{SAS}_2$ to be equal is 2^{-k} (k is the length of the SAS) the attacker will establish two different authentication strings $\mathtt{SAS}_1$ and $\mathtt{SAS}_2$ with a high probability. However, in semi-authentic model where the adversary can modify the bits (flip $0 \rightarrow 1$) this probability is significantly reduced.

If the ith bits of $\mathtt{SAS}_1$ and $\mathtt{SAS}_2$ are equal, an attacker will not need to modify them in any way. On the other hand, if the ith bits of $\mathtt{SAS}_1$ and $\mathtt{SAS}_2$ equal 1 and 0, respectively, an attacker could flip $0 \rightarrow 1$ by using the laser. If the ith bits of $\mathtt{SAS}_1$ and $\mathtt{SAS}_2$ are 0 and 1, the attacker will be unable to flip $1 \rightarrow 0$ for he cannot switch OFF an already powered ON LED. This is summarized below:

$\mathtt{SAS}_{1i}$	$\mathtt{SAS}_{2i}$	Attack
0	0	yes
0	1	no
1	0	yes
1	1	yes

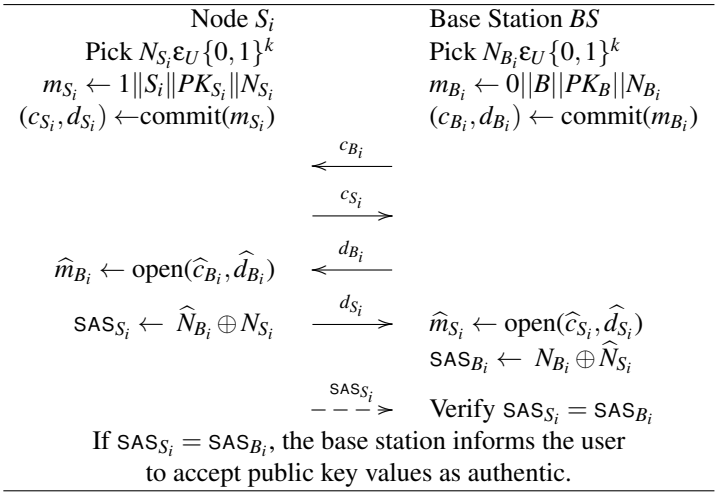


Figure 4.8. SAS *protocol by [5, 6]. The dashed arrow represents communication over a semi-authentic VLC.*

Thus, we conclude that that 3 combinations of ith bits of SAS_1 and SAS_2 are beneficial for the attacker (all combinations but the second one). It follows that the probability for an attacker to modify the bits is 3/4, therefore, the probability of a successful attack increases to $(3/4)^k$ as opposed to 2^{-k} (in the case of authentic VLC). If $k = 15$, the probability in a single attack increases from 2^{-15} to approximately 2^{-6}.

Virtual node attack. Let us assume the user wants to initialize one node (S_1) and the attacker (M) wants to inject his own virtual node (S_2) as shown in Figure 4.7. Attacker M simply exchanges public key values over a radio channel with BS and points his laser within the visible area of the base station's camera. The pointed laser is used to create a virtual node (device S_2 in Figure 4.7), and as such, to "blink" the correct short authentication string in such a way that the base station's camera detects it. The BS compares the SAS_2 it received from the attacker's laser with the one established over radio, sees that they match, and accepts the public key values from M as authentic.

4.4.2 "Public Key"-Based Deployment Protocol

In this part we assume that each node S_i and the base station BS previously generated public key values PK_{S_i} and PK_B. In order to exchange authenticated public key values over radio channel, we propose using the protocol introduced in [5, 6], and shown in Figure 4.8. Please note, the base station performs this protocol individually with each node. The protocol evolves as follows:

(i) The user counts the number of nodes he/she wants to initialize and enters the number into the base station control software via a keyboard. We will show later that by entering the number of nodes we can prevent the virtual node attack and make the size of the SAS invariant of the number of nodes to be initialized.

(ii) The user switches the nodes ON and places them in front of the camera, with the LEDs facing the camera.

(iii) The node's LED starts flashing with the delimiter 111000 to indicate to the *BS* they are ready to be initialized and to enable the *BS* to count them.

(iv) Next, the user instructs the base station to begin with the protocol shown in Figure 4.8. Having exchanged commit/open pairs with the *BS*, each node S_i first calculates the respective $\mathtt{SAS}_{S_i}$ (Figure 4.8), Manchester encodes $\mathtt{SAS}_{S_i}$ and begins transmitting it repetitively via a VLC (using on-off keying, switching LED OFF and LED ON). The Manchester encoded *short authentication string* $M(\mathtt{SAS}_{S_i})$ is separated with delimiter 111000. The usage of the delimiter and Manchester encoding was inspired by *I*-codes [127] and was used to prevent the flipping attacks (Section 4.4.3). Finally, the node transmits the following repetitive sequence:

$$\cdots \overbrace{111000}^{\text{delimiter}} \overbrace{1001\cdots10}^{M(\mathtt{SAS}_{S_i})} \overbrace{111000}^{\text{delimiter}} \overbrace{1001\cdots10}^{M(\mathtt{SAS}_{S_i})} \overbrace{111000}^{\text{delimiter}} \cdots$$

(v) If the SAS verification is successful for **all** the nodes, the user is instructed to finalize the initialization procedure by pushing a button on each of the nodes. If one or more nodes fail to initialize properly (e.g. due to errors in transmission, attacks etc.) the initialization procedure is aborted for all the participating nodes.

4.4.3 Short Security Analysis

In this section we provide only a short security analysis of the public key deployment protocol.

Flipping attacks. In order to prevent flipping attacks we used Manchester encoded $\mathtt{SAS}_{S_i}$ for the transmission via VLC. Note that such a message contains an equal number of 0s and 1s. Due to the on-off keying modulation and the fact that an attacker is unable to switch OFF the LED (flip $1 \rightarrow 0$), any attempt of flipping will be detected by the *BS* as an excess of 1s. This construction is proved secure in [127].

Virtual node attack. According to the protocol, the base station knows exactly how many nodes it has to initialize (step (i) of the protocol). In addition, the *BS* counts itself the nodes by detecting respective delimiters (111000) transmitted over VLC. In order to successfully inject his own virtual nodes, the attacker has to block transmission of the delimiter 111000 over VLC for at least one of the nodes. However, the attacker cannot do this, for he is unable to turn OFF an already switched ON LED. In addition, any attempt of flipping $0 \rightarrow 1$ in the delimiter will be detected by the *BS*.

All or none. The design choice to abort the initialization procedure if at least one node fails to initialize properly makes the SAS invariant of the number of nodes. Indeed, from the above analysis we know that an attacker can neither add new (virtual) nodes, remove existing (legal) ones, nor perform bit flipping attacks. It follows that the attacker can only

Figure 4.9. Meshnetics ZigBee sensor node used in our implementation.

try to perform a man-in-the-middle attack against one or more legal nodes. Now, if an attacker attempts to mount a man-in-the-middle attack against m nodes (out of n nodes) and the respective short authentication strings are mutually independent, the probability of a successful attack against at least one sensor node, in a single attempt, will be at least $\min\{m \cdot 2^{-k}, 1\}$ [5]. For example, if the attacker attacks $m = 100$ nodes and $k = 15$, the probability for the attacker to succeed against at least one node is around 2^{-8}. However, by restricting the attacker to be successful against all the nodes, the probability for the attacker to succeed is reduced to $(2^{-15})^m = 2^{-15 \cdot m}$. Therefore, the best strategy for the attacker is to mount an attack against exactly one node (i.e., $m = 1$), which implies the probability of success (in a single attempt) to be bounded by $2^{-k} + \varepsilon$ (k being the size of SAS and ε a negligible probability) [5].

4.5 Implementation

We next describe the implementation of our secret-key deployment protocol. More specifically, we describe the implementation of a simple random number generator (RNG) and the key recognition software that enables communication over the light channel. We used Meshnetics ZigBee sensor nodes (Figure 4.9) equipped with Green and Red LEDs, Atmel AT-mega1281V Microcontrollers and AT86RF230 RF Transceivers. Each sensor module features 128KB of flash memory and 8KB of RAM with data rate of 250 kbps in frequency band from $2.400 - 2.483$ GHz. For software developing and testing of the initialization procedure, a PC with the following configuration was used: Intel dual core processor clocked at 2.66GHz, 2GB of RAM, a Logitech notebook deluxe webcam with VGA resolution at 30fps interfaced via USB to the computer and Windows XP SP3 operating system.

4.5.1 Random Number Generator

The key feature for secure communication lies in a good random number generator. However, low processing devices like ZigBee sensor nodes are unable to perform a good processor-based random number generators. In this section, we describe our Random Number Generator (RNG). We first describe some related work on random number generators

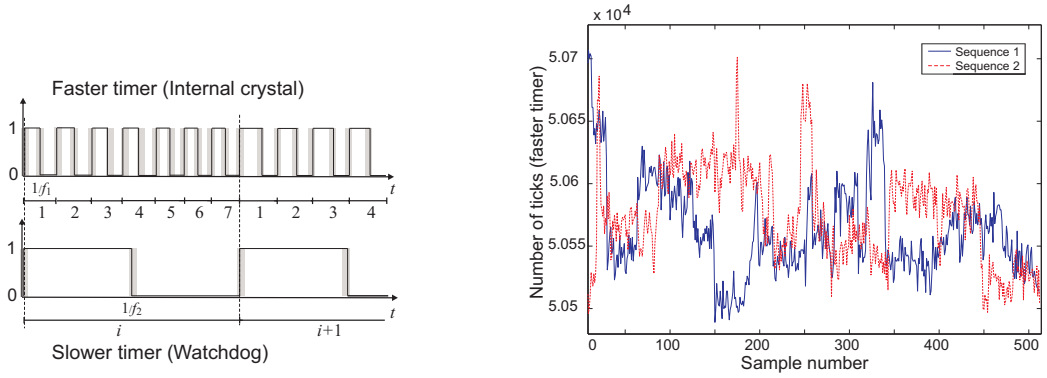


Figure 4.10. (a)An example of oscillator frequency instability (jitter). (b) Two traces showing the number of ticks of a faster timer relating to one tick of the slower one.

suitable for devices with limited processing capabilities.

TinyRNG [128] uses transmission bit errors as a source of randomness. These bits are randomly distributed as well as uncorrelated and may not be manipulated by an adversary. In [129] two oscillators are used, one oscillating much faster than the other. Generated bit stream's randomness is based on the frequency instability of a free running oscillator. The slow oscillator samples the higher frequency oscillator. They have shown that if the jitter in the slow oscillator signal is sufficient, the output of the RNG will have very little bit-to-bit correlation. Tkacik [130] also uses two free-running oscillators whose frequency vary with voltage and temperature. Random numbers are generated as exclusive-or of previously selected and permuted 32 bits of the LFSR (linear feedback shift register) and CASR (cellular automata shift register). Each shift register is clocked by these oscillators. However, an initial seed is required for each register.

Design of a Random Number Generator

In our implementation we used the approach from [129]. The generation of random numbers goes as follows: Meshnetics ZigBee nodes are equipped with two usable oscillators, an Internal Calibrated RC Oscillator (4 MHz) and a Watchdog Oscillator (128 kHz) [131]. The software running on the sensor nodes creates two timers; one timer is associated with the slower oscillator and the other timer with the faster one. The timers are configured with clock dividers in such a way that the slower timer fires once per second, while the faster one fires roughly 50000 times per second. On every tick of the slower timer, the number of ticks from the faster timer is logged. Figure 4.10(b) shows two traces of the number of ticks from the faster timer during the period of 512 ticks from the slower timer (roughly 512 seconds). As shown, the source of randomness comes from the instability (jitter) of the two used oscillators (Figure 4.10(a)).

Table 4.1. An example of digital postprocessing performed on the generated raw bitstream as well as the generation of random numbers. The generated bit stream is 110100.

Number of ticks (Watchdog)	Number of ticks (Internal RC)	Partial binary representation	Last two digits
1	50607	10101111	11
2	50605	10101101	01
3	50640	11010000	00

Table 4.2. ENT test results

Entropy = 0.999999 bits per bit.
Optimum compression would reduce the size of this 3×10^6 bit file by 0 percent.
Chi square distribution for 3267632 samples is 2.40 and randomly would exceed this value 12.14 percent of the times.
Arithmetic mean value of data bits is 0.4996 (0.5 = random).
Monte Carlo value for Pi is 3.155460889 (error 0.44 percent).
Serial correlation coefficient is 0.000264 (totally uncorrelated = 0.0).

Digital postprocessing. Table 4.1 shows the digital postprocessing and the random number generation process. As shown, on each successful low frequency timer tick the number of high frequency ticks is counted. Next, this value is converted into a binary representation (last eight binary digits are presented in Table 4.1) from which the last two bits are taken. We could extract more than two bits at the expense of a more complex extractor. Since for our purpose the entropy is sufficient, we choose to use this simple extractor. The results of statistical tests are presented in the next section.

Statistical Tests

ENT [132] and NIST [133] statistical test suites were used to test the randomness of our generated bitstreams. Statistical tests were conducted on a 3×10^6 long bitstream which we obtained from 7 ZigBee nodes over the period of approximately 3 days.

ENT [132] is a pseudorandom number frequency test that performs a variety of tests such as Entropy, Arithmetic mean, Monte Carlo value for Pi, Serial correlation coefficient and Chi square distribution. Table 4.2 contains the results of the ENT test performed on a 3×10^6 long bitstream.

Only the last two bits were taken from the binary representation of the faster timer tick count. If more than two bits are taken, the bitstream fails the "Chi square" part of the ENT test suite. But, as we already mentioned, our RNG directly samples the number of faster timer ticks, without the requirement for other complex extractors.

Table 4.3. NIST test results

TEST	P-VALUE	PROPORTION	TEST	P-VALUE	PROPORTION
frequency	0.148094	0.9922	fft	0.468595	1.0000
block-frequency	0.500934	0.9922	aperiodic	all passed	all passed
cumulative-sums	0.311542	0.9922	apen	0.275709	0.9844
cumulative-sums	0.031497	0.9922	serial	0.671779	0.9844
runs	0.437274	0.9922	serial	0.637119	0.9922

NIST STS [133] contains 15 tests out of which only 8 were performed due to the minimum bitstream requirement (3×10^6 bits were produced) for each test. Each test is used to calculate the p-value which shows the strength of the null hypothesis. The hypothesis passes the test if the p-value is higher than 0.01 in which case the sequence is considered to be random. As shown in Table 4.3, the generated sequence passed the tests (p-value is higher than 0.01).

These tests were performed over the raw bits. Since the output bitstream passes both NIST and ENT test suites, no additional randomness extractors (universal hash functions [134, 135], von Neumann extractor [136], or simply applying a cryptographic hash function over the bitstream) are necesarry.

These results are preliminary; future work will include a more detailed study of factors which impact the work of RC oscillators (e.g. voltage and temperature), which directly reflects on the quality of the generated random numbers.

4.5.2 Communication via a Visible Light Channel

After the key generation follows the key transmission via an out-of-band Visible Light Channel (VLC). The sensor nodes are programmed in such a way that generated key bits are Manchester encoded prior to transmission which ensures lower bit error rates during the transmission over VLC. The bits are transmitted in such a way that logical 0 and 1 of our bitstream are represented with LED ON and OFF states, respectively. The duration of each state (single LED's blink) is approximately 200 ms. In Figure 4.11 we give an example of a bitstream's "life-cycle"; from the bit generation to the bit transmission phase. As shown in Figure 4.11, the generated bits are separated in such a way that the first and the second LED (Green and Red LED in our case) transmit the code of the uneven and even bits via VLC, respectively. This form of transmission was used to achieve easier key recognition on the side of the base station using the mathematical operation of convolution, as described in the sequel.

Computer Vision. Once the user places sensor nodes inside of the box, we use our computer vision (CV) system to derive the secret key from the nodes' LED blinking sequence.

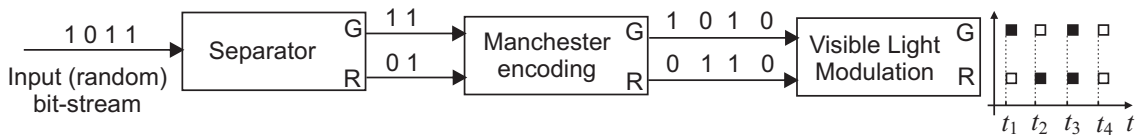


Figure 4.11. An example of the bit stream sent via VLC using Manchester encoding. G and R stand for Green and Red LED, respectively.

We developed our CV system in MATLAB 2007 GUI [47], and achieved transmission speeds of 10 bits per second (5 b/s per each LED).

The image processing part of our CV system is CPU demanding. In order to achieve real-time performance, we process only certain parts of an webcam-obtained image - so called “Areas of Interest" (small rectangles encompassing LEDs of each node in Figure 4.12). The algorithm was designed to work with two LEDs on each node (Green and Red LED). To determine the Area of Interest (AoI) for each node, which is the first step, a few seconds of buffered frames is required. Once the areas are determined, the rest of the algorithm is performed in real-time. All of the following steps are performed only over Areas of Interest. The rest of the image does not contain any relevant information, and thus is excluded from future processing.

Image transformation. In the second step, the selected image parts (AoIs) are converted from RGB to HSV color space, known to be more reliable for detecting colors in low and changing light conditions [137]. Obtained images are tested for their levels of Hue, Saturation and Value, which enables us to detect the state (ON/OFF) of each LED. Color detector relies mainly on the level of Hue, while levels of Saturation and Value are just used in order to avoid false detection due to noise at low illumination conditions in the dark box.

Recognition of VLC signal. Due to a high frame loss and transmission error rates, during the transmission each bit is repeated in 6 consecutive frames (6 samples per bit). Decoding starts by detecting first 18 frames of the packet delimiter (3 binary ones in a sequence

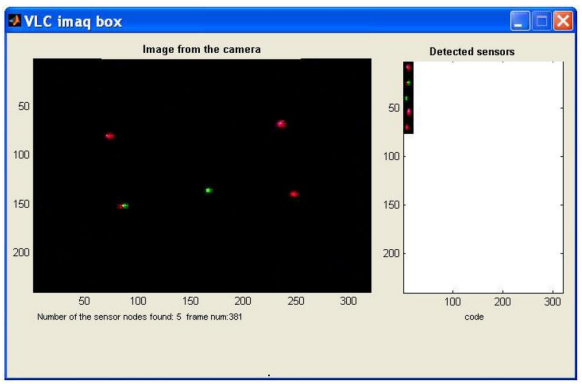


Figure 4.12. An example showing detection of 5 nodes based on their flashing LEDs.

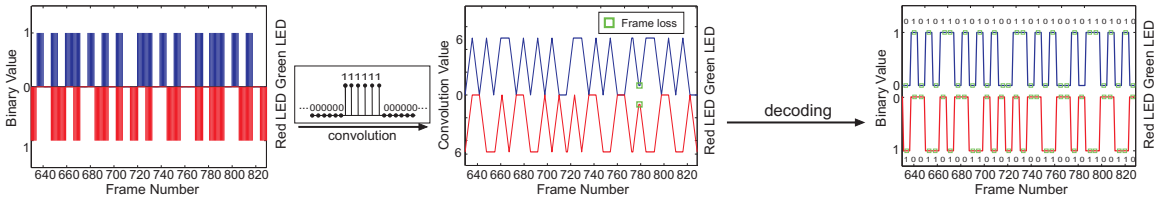


Figure 4.13. A key recognition process consisting of: (a) detecting the status of LED indicators, (b) applying the convolution over the sampled area and (c) the bit identification process after the convolution.

on both LEDs). Next, the key recognition algorithm performs the mathematical operation of convolution over the frames following the delimiter with a mask of six consecutive 1s (Figure 4.13(a)). As a result, data arrays containing values ranging from 0 to 6 are obtained (Figure 4.13(b)), where elements with extremes 0 and 6 are decoded as bits 0 and 1, respectively. Plateaus (areas with multiple, identical and consecutive elements) are decoded as double 0s or 1s, depending on their values (0 or 6). As Manchester encoding was used, only the convoluted signal's slope is analyzed, and not their values. This results in a method highly robust to de-synchronization effects. As shown by an example in Figure 4.13(b), frame loss during transmission via VLC does not affect correct bit recognition in any way.

4.6 Related Work

Recently, many key deployment schemes such as Zig Bee [138], SPINS [114], LEAP [139] and Transitory Master Key [140] have been proposed. Others [118, 117, 115, 141, 142] propose random key pre-distribution schemes. All of these schemes rely on an unspecified secure key deployment mechanism between devices.

In On-off Keying, the presence of an RF signal represents a binary 1, while its absence represents a binary 0 [5, 127]. By using an unidirectional encoding scheme, On-off Keying ensures that an attacker is unable to modify a packet during transmission.

In Shake Them Up [143], user establishes a secret key between two nodes by holding and shaking the devices together while they send identical packets over the radio. This way, they assume that an adversary is unable to distinguish the source of the packets. This may be violated by using radio fingerprinting. Also, this does not scale well. The three related schemes are "Are You With Me" [144], "Smart-Its Friends" [145] and "Shake Well Before Use" [146].

In Key Infection [147], two nodes establish a secret key by sending it in the clear over radio. They assume an attacker is unable to eavesdrop all the keys from all the nodes (e.g., 10.000 nodes) during key deployment. Based on simplicity and cost effectiveness, this scheme is insecure against a determined adversary. Moreover, an adversary is capable of

injecting his own key, also violating key authentication.

In Resurrecting Duckling [148], a physical contact is required to securely establish a secret key. Based on the assumption that physical contact is secure, key authenticity and secrecy are ensured. But, since it requires specialized additional hardware, this scheme is not cost effective.

In Message In a Bottle [149], keys are sent in the clear to the nodes located inside a Faraday cage that ensures key secrecy and authenticity. However, the number of simultaneously initialized nodes determines the size of the Faraday cage. Moreover, a scale is used to determine the number of nodes within the Faraday cage based on total Faraday cage weight. In order to ensure key secrecy and authenticity for a large number of nodes, this scheme requires specialized setup hardware.

In HAPADEP [150] both data and verification information is sent over an audio channel. The pairing devices are both required to have speakers and microphones. In a related paper, Saxena and Uddin [8] present a device pairing method with an unidirectional channel based on devices equipped with LEDs and a video camera as the receiver. Their method is used for asymmetric pairing scenarios. Again, Saxena et. al. [12] use an auxiliary device (a laptop equipped with a web camera) to compare a short authentication string sent from the nodes to the laptop via unidirectional visible light channel. Both protocols are prone to laser attacks where an adversary may inject his/her malicious key by modifying the messages sent via the light channel with a directional light source (e.g. laser emitter).

Talking to strangers [120] requires specialized setup hardware (e.g. audio or infrared) in order to setup a public key. Seeing Is Believing uses an installation device with a camera or a bar code reader to create an out-of-band secure channel [151]. Key authenticity is achieved through public keys.

Mayrhofer and Welch [152] use an out-of-band laser channel constructed with off the shelf components for transmitting short authentication strings. According to [152], the proposed solution does not ensure complete authenticity of the the laser channel. Roman and Lopez [153] discuss general aspects of communication over a visible light channel.

4.7 Summary

In this chapter we proposed two novel *multichannel* protocols for initialization of large scale wireless WSN networks. The first protocol uses only secret key cryptography and is suitable for CPU-constrained sensor nodes. The second protocol is based on public key cryptography. Both protocols involve communication over a bidirectional radio channel and an unidirectional out-of-band *visible light channel*.

We demonstrated the importance of considering a very strong and realistic attacker model, where an attacker can eavesdrop, jam and modify transmitted messages in both a radio and a visible light channel; many existing protocols that rely on a visible light channel

were shown to be insecure in this strong adversary model. Our "public key" - based protocol is designed to be secure in this very strong attacker model. Moreover, we showed that procedure "all or none" keeps invariant the size of *short authentication strings* to changing the number of sensor nodes to be initialized.

The proposed protocols are implemented on the Meshnetics ZigBee sensor nodes. We showed that the proposed protocols are cheap to implement (sensor node has to be equipped with one LED and a "pushbutton") and scalable. We also designed and tested a simple random number generator suitable for CPU-constrained sensor nodes.

5 BOOTSTRAPPING MULTIPLE CONSTRAINED WIRELESS DEVICES: AN APPROACH WITHOUT AUXILIARY DEVICES

5.1 Introduction

In the previous chapter, we proposed two scalable approaches for loading cryptographic keys into WSN devices. While being suitable for a large number of WSN devices, the proposed solutions rely on the presence of an auxiliary device (such as a programmable camera). In this chapter, we present first secure and usable initialization mechanism that works with multiple (sensor) devices having constrained resources (a LED, a button and limited power supply) and does not require any auxiliary devices.

Our initialization mechanism is based on a novel (multichannel) protocol, called the *Group message Authentication Protocol* - GAP. GAP involves communication over a radio channel and an out-of-band visible light channel (VLC). GAP is inspired by the two-party SAS protocol [6, 5]; we show that straightforward generalizations of SAS to a multiparty protocol may easily fall short of being secure. A notable feature of GAP is that the information to be authenticated is independent of the *short authentication string* (an *indirect* binding protocol [9]) to be verified by the user over a visible light channel (i.e. the GAS and authenticated information are completely independent in the sense of probability). This, as we show, results in a lower communication cost compared to existing *direct* binding protocols. The advantage in the communication cost of our GAP protocol is especially important for battery-powered devices, such as wireless sensor nodes. We also show how to secure GAP against malicious insider attacks (compromised sensor nodes); in [10], the devices are assumed to be benign during the initialization phase.

As we show later, the visible light channel (VLC) is prone to certain bit-manipulation attacks, that (contrary to the common belief) renders VLC *semi-authenticated*; however many existing protocols [11, 12, 8, 10] that use VLC consider it to be authenticated. In order to prevent these attacks, we use a simple combination of well known unidirectional codes (Berger and Manchester), which is easy to interpret by an end user.

Finally, we demonstrate the feasibility of the proposed mechanism via a preliminary usability study with 28 users. The study indicates that the method has reasonably low execution

time, minimal error rate and is user-friendly. We further discuss how the usability and scalability of our mechanism can be improved by utilizing a zero-configuration auxiliary device (e.g., a standard camera phone with no additional computational logic), when available.

We note that although we target our scheme as a means of secure initialization of a WSN, our proposal is also equally applicable to other wireless devices scenarios. This includes, for example, the initialization of a number of commodity wireless access points that need to be installed as part of an enterprise's wireless network.

Chapter Outline. In Section 5.2, we state the assumptions and a give an overview of our solution. In Section 5.3, we present the GAP protocol, and in Section 5.4, we give a solution against insider attacks. Section 5.5 deals with securing of transmissions over a visible light channel. We discuss several implementation aspects of GAP in Section 5.6. Usability evaluation is presented in Section 5.7. Related work is provided in Section 5.9, and we summarize the chapter in Section 5.10.

5.2 Assumptions and Solution Overview

We consider the following problem: *How to securely initialize a large number of sensor devices in a user-friendly way without relying on auxiliary devices?* An important research challenge, is to design secure WSN initialization mechanisms that satisfy the following properties:

1. *User-friendliness*: The mechanism should be easily administered by a non-specialist and unaided end user. By unaided, we mean that the user is not in possession of any auxiliary devices that can facilitate or automate the initialization process. It is important to note that auxiliary devices may not always be available. They will also add to the overall cost of the system. Furthermore, requiring a specialized auxiliary device only for the sake of a security operation presents deployment hurdles.

2. *Scalability:* The mechanism should be able to initialize a reasonably large number of nodes. Due to the manual nature of initialization (because of the lack of an auxiliary device as stated above), however, one can only hope to initialize a maximum of, for example, 20-30 devices per batch of initialization (the mechanism can be repeated in multiple batches whenever needed).

3. *Compatibility with Constrained Resources:* Being mass produced, sensor devices are usually constrained, i.e., they do not usually have wired or other traditional interfaces, such as displays and keypads. Thus, the initialization mechanism should be able to work within these resource constraints. In other words, secure initialization should still be possible even with a few on-board LEDs and buttons[1]. Because sensor-motes

[1]Most commercially available sensor motes and devices possess multiple LEDs and an on/off button (Mica2 [154]).

are typically on a limited power supply, an additional goal is to minimize the communication overhead associated with the secure initialization mechanism.

In addition to the necessary requirement of providing security against the man-in-the-middle attacker, it is desirable that the initialization provides protection against compromised nodes. This is needed to address scenarios such as those whereby a manufacturer sneaks in malicious sensor node(s) along with normal sensor nodes shipped to a customer, as pointed out in [149].

5.2.1 Prior Work

The problem of secure initialization of sensor devices has received considerable attention by the research community and a number of solutions have been proposed. The prior solutions do not satisfy one or more of the requirements outlined above, however. Many existing solutions work only with a small number of (i.e., two) wireless devices and are not scalable. These include the "Shake-them-up" [143] scheme that suggests a simple manual technique for pairing two sensors that involves shaking and twirling them in very close proximity to each other, in order to prevent eavesdropping. Another scheme "Are You with Me" [144] uses human-controlled movement to establish a secret key between two devices.

The other notable recent result "Message-in-a-Bottle" [149] explores the use of a *Faraday Cage* to shield communication from eavesdropping and outside interference and allow a set of sensors to be simultaneously paired with the sink. This is a scalable technique, although as illustrated in [149], building a truly secure Faraday Cage is a challenge. The primary issue with this approach is the need to obtain and carry around a specialized piece of equipment – a Faraday Cage (an auxiliary device). The cost and the physical bulk of the cage can be problematic in practice.

Another well-established approach to securing initial key deployment involves two communication channels – an insecure high bandwidth radio channel and a low-bandwidth Out-of-Band (OoB) channel, such as visible light. The security of this approach relies on the assumption that the underlying OoB channel, being human-perceptible, is authenticated and preserves the integrity of transmitted messages. This approach has also been discussed in the 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) RFC4919 specification [113]. According to this specification, one of the major security considerations involves application of "out-of-band techniques for the initial key establishment" among a large number of sensor devices.

Many prior solutions based on the above multi-channel approach, however, rely on the presence of auxiliary devices. For example, the solutions presented in [11, 12, 8, 155, 10, 156] all require a programmable video camera. Yet other solutions (e.g., GAnGs [157] and Groupthink [158]) are geared for multi-user group settings whereby each user is in possession of a personal device, such as a smartphone. These solutions require interfaces beyond

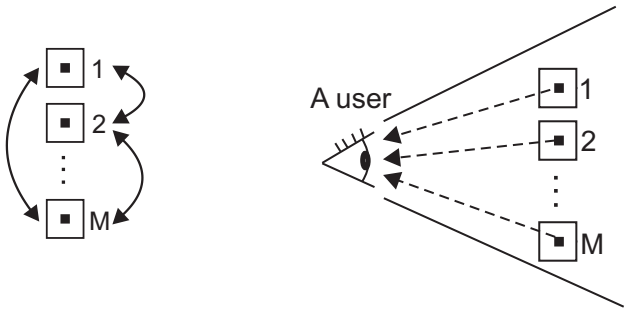


Figure 5.1. Two phases of GAP: in (a) devices exchange messages to be authenticated over a radio channel and (b) a user performs authentication via a visible light channel (dashed arrow).

the reach of current sensor devices, such as full displays or cameras.

5.2.2 Attacker Model

In our initialization protocol the information is sent over two channels: a radio and a visible light channel. We assume that an attacker has a full control over the radio channel; he can eavesdrop, drop, delay, replay and modify messages sent on this channel. He can initiate communication with any device and at any given time. The attacker can also eavesdrop and modify messages sent over a visible light channel (VLC) at all times (Section 5.5); the attacker however cannot disable the visible light communication channel (erase the messages). To convey information via VLC we use on-off keying (i.e., bit "0": LED OFF, bit "1" LED ON). Note that an attacker equipped with a directional light source (e.g. a laser) can potentially modify bits sent via VLC. With such keying the attacker can modify messages by flipping $0 \rightarrow 1$, but not vice versa ($1 \rightarrow 0$) as the attacker cannot force a switched ON LED to power OFF. In this case, in our model, we speak of a *semi-authenticated* visible light channel. For this reason we apply error detection codes to the group authentication string before its transmission over VLC; in Section 5.5 we show that such coding prevents the bit flipping attacks. Note that the existing related approaches [11, 12, 8, 10, 156] consider the visible channel to be authenticated (i.e., the attacker does not control messages sent over VLC). These protocols are therefore insecure in our model and we work in a much stronger attacker model. To start with, we assume that devices involved in key deployment are not compromised. Later in Section 5.4, we extend this attacker model to include stronger insider attacks from compromised wireless devices.

5.2.3 Solution Overview

A user wishes to initialize a larger set of wireless sensor devices. She makes sure that the sensor devices are all placed in her visual field, so that she can simultaneously observe their

LEDs (Figure 5.1(b)). She powers on the devices, picks an arbitrary one and designates it as a *coordinator* node. Once this has been done, the coordinator initiates the execution of our *Group message Authentication Protocol* (GAP) that enables mutual authentication of messages (e.g. public keys) exchanged by the devices over an insecure radio channel (Figure 5.1(a)). Our protocol accomplishes this by first generating a common short *group authentication string* (GAS) on all the devices. In turn, the GAS is communicated over a semi-authenticated visible light channel. Finally, the user visually verifies that the GAS transmitted over the VLC is the same on all the devices (Figure 5.1(b)). If the verification is successful the user completes the protocol by pushing the button on each device.

A notable feature of our *Group message Authentication Protocol* is that it binds authenticated information (e.g. public keys) *indirectly* to a group authentication string [9]. In indirect binding protocols, the GAS is functionally independent of the information to be authenticated (i.e. the GAS and authenticated information are completely independent in the sense of probability [9]). While there is no relation between the compared GAS and the authentic information, the security of the protocols comes from some mechanism that binds short random nonces (used for calculation of the GAS) and the authentic information together in a secure way. In our case, this secure binding is achieved through the commitment scheme. For this reason in GAP the devices exchange messages to be authenticated (e.g., public keys) through specially formed commit/decommit pairs over a radio channel (Section 5.3 and Figure 5.2).

On the other hand, the direct binding approach requires the GAS to be dependent on the information devices want to authenticate. The basic principle behind the direct binding approach is to make all the parties, who are intended to be part of a protocol run, agree on a short-output hash or digest of a complete description (the collection of all the information that any member of the group wishes to have authenticated to) of the protocol run [9].

We show in Section 5.3.3 that indirect binding, as used in our GAP protocol, can reduce the communication cost compared to the representative directly binding protocol proposed by Laur and Pasini in [2, 159].

5.3 Efficient and Secure Group Message Authentication Protocol

In this section we provide details of our *Group message Authentication Protocol* - GAP. The GAP protocol shares some similarities with the existing group authentication protocols (please see an excellent survey on such protocols by Nguyen and Roscoe [9]). Still, it is different in two important ways:

1. The GAP binds authenticated information (e.g. public keys) *indirectly* to a *group authentication string* [9]. As a result, GAP incurs *lower communication cost* compared to some (provably secure) directly binding protocols (Table 5.1). This is especially

important for battery-powered devices such as wireless sensor devices. It appears that this disadvantage is common to all (provably secure) directly binding schemes that use universal hash functions to generate short group authentication string (as discussed later in the section).

2. The GAP is a *secure* generalization of the two party SAS protocol [5] into a multiparty version. This is in contrast with recent Nguyen and Roscoe's attempt in [9] to generalize the SAS protocol. Indeed, we show in Section 5.3.2 that their straightforward generalization results in an insecure protocol. We state the security result for GAP in Theorem 2.

Two aspects of the GAP protocol are essential for its security: (i) GAP imposes strict ordering among the messages exchanged by the devices and (ii) at least one device from the group (e.g. a coordinator) must know the correct group size. Please note that the group size information is entered by the user. This process can be very challenging with devices that have constrained interfaces (e.g. a single pushbutton and one LED). We describe a possible user-friendly approach for accomplishing this task in Section 5.6 and study its usability aspects in Section 5.7. We next give details of our GAP protocol.

5.3.1 Description of the GAP

Let us introduce some notation. A user wishes to initialize a set of M sensor devices. We assume that sensor devices involved in key deployment are trusted. We denote this group with $\mathcal{G}$, i.e., $\mathcal{G} = \{ID_1, ID_2, \ldots, ID_M\}$, ID_j being the identity of the jth device (IDs are unique). The set $\mathcal{G}$ is ordered with respect to the increasing identities ($ID_1 < ID_2 < \ldots ID_M$). Let PK_j, $j \in \mathcal{G}$, denote a public key of wireless device j (i.e., the device with the identifier ID_j). For clarity, variables $\widehat{c}_j$, $\widehat{d}_j$, $\widehat{N}_j$ and $\widehat{h}_{\mathcal{G}_j}$ denote the variables from device with identifier ID_j (or shortly device j) received by the device i (i.e., the device with the identifier ID_i). The hats in the notation indicate a possible modification (or influence) by an adversary. We denote with k the *coordinator* sensor device. The GAP protocol evolves as shown in Figure 5.2. The goal of the protocol is to assist the devices in $\mathcal{G}$ to mutually authenticate their respective public keys.

Phase I (radio). The user first designates one arbitrary sensor device as a coordinator; this is achieved by a push on a button (Section 5.5). We denote this device with $k \in \mathcal{G}$. Upon selecting the coordinator $k \in \mathcal{G}$, it starts to broadcast its ID_k to all other nodes. Upon receiving the ID_k from the coordinator, the other nodes from $\mathcal{G}$ begin to broadcast their own IDs until the predefined timeout (a couple of seconds). Each device $i \in \mathcal{G}$ orders all received IDs in the order of increased identities ($ID_1 < ID_2 < \ldots < ID_M$). We use $\mathcal{G}_i$ to denote the ordered set of IDs as seen by device $i \in \mathcal{G}$.

Phase II (radio). A commitment scheme is an important cryptographic building block that is used in our GAP. A commitment function transforms a value m into a commit-

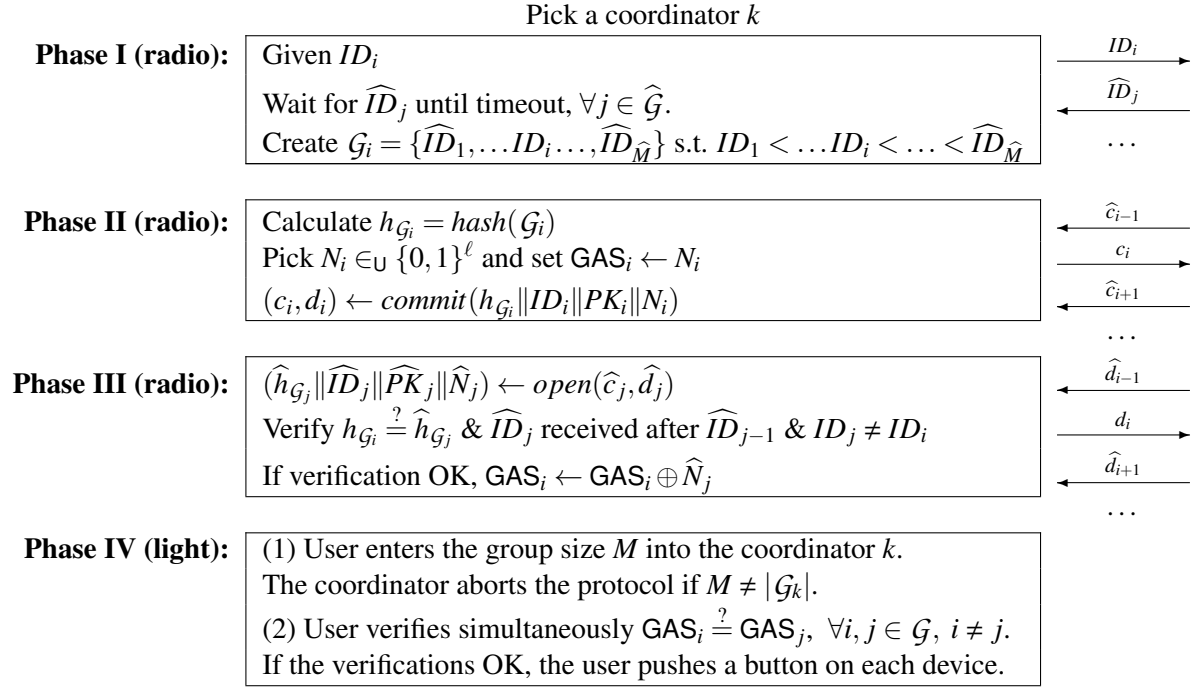


Figure 5.2. Group message Authentication Protocol (GAP): Authenticating public keys PK_i ($i \in \mathcal{G}$) using a Group Authentication String (GAS).

ment/opening pair (c, d), where c reveals no information about m, but (c, d) together reveal m, and it is infeasible to find $\widehat{d}$ such that reveals $\widehat{m} \neq m$. Every device $i \in \mathcal{G}$ calculates a hash value $h_{\mathcal{G}_i}$ of the set $\mathcal{G}_i$, generates random nonce N_i (ℓ bits). Next, the device i sends its commitment c_i to all other participants $j \in \mathcal{G}_i$ but only after having received commitments from all the devices $j \in \mathcal{G}_i$, such that $ID_j < ID_i$. Note the each device $i \in \mathcal{G}$ adds the $h_{\mathcal{G}_i}$ and its ID_i to the commitment. This is used to prevent reflection and node injection attacks.

Phase III (radio). Decommitments are sent in the same order as the corresponding commitments. Upon receiving decommitment $\widehat{d}_j$ from $j \in \mathcal{G}$ the device i ($i \neq j$) opens the commitment $\widehat{c}_j$ and verifies the sender's ID and that $\mathcal{G}_i$ matches $\mathcal{G}_j$ (by comparing $h_{\mathcal{G}_i}$ and $\widehat{h}_{\mathcal{G}_j}$). If the verifications are OK, i updates GAS_i as follows: $\mathsf{GAS}_i \leftarrow \mathsf{GAS}_i \oplus \widehat{N}_j$. Otherwise, device i aborts the protocol. This process repeats for all received decommitments.

Phase IV (light). In the last phase of the protocol the user first enters the size of the group (M) into the coordinator that in turn verifies it to be equal to $|\mathcal{G}_k|$ (see Section 5.6 for details). Next, the user simultaneously verifies that the established GAS values on all the devices satisfy $\mathsf{GAS}_i = \mathsf{GAS}_j$ $\forall i, j \in \mathcal{G}$. In Section 5.5) we describe a secure method to accomplish this task, which is based on LED blinking. If the verification is OK, the user pushes a button on each device to complete the initialization process. At this stage, each sensor device from $\mathcal{G}$ holds authenticated public keys (or other messages) of all the other devices.

Next we state the security result for the GAP protocol. We assume that sensor devices involved in key deployment are trusted. Next, we assume the used hash function $hash(\cdot)$ to

be collision resistant and the commitment scheme $commit(\cdot)/open(\cdot)$ to be non-malleable.

Theorem 2. *The probability that a computationally bounded adversary breaks (in a single attempt) the GAP is bounded by* $2^{-\ell} + \varepsilon$*, where* ℓ *is the size (in bits) of the group authentication string (GAS) and* ε *is a negligible probability.*

We provide the sketch of the proof of Theorem 2 in Section 5.8. The important implication of this result is that the authentication string GAS can be reasonably short (e.g., 15-20 bits). This is especially important given that the user verifies these bits without any assistance from auxiliary devices. In Section 5.8 we proved the security of the GAP protocol in the attacker model in which the sensor nodes are trusted. However, GAP protocol is not secure against insider attacks (compromised nodes). Note that this is a realistic attack because an attacker, for example, can sneak in malicious sensor node(s) along with normal sensor nodes during the transportation phase. Later in Section 5.4.1 we propose a simple extension of the GAP protocol, which is secure against compromised insider devices, all this at a small additional communication cost.

As stated at the beginning of this section, the GAP protocol generalizes the two party SAS protocol into a multiparty version. Although it may appear at first that this is a straightforward task, we exemplify next that such belief may be unfounded.

5.3.2 Insecure SAS Protocol Generalization

In [9] Nguyen and Roscoe propose a straightforward generalization of the two party SAS protocol [5]. Similar to GAP, in their proposal group members in the first phase exchange commitment messages and subsequently, in the second phase, exchange the corresponding decommitments. However, Nguyen and Roscoe allow for arbitrary interleaving of the commitment and decommitment messages in their respective phases (which is not the case in our GAP). Such a construction (arbitrary interleaving) results in an insecure protocol, as we exemplify next. We show that an attacker can replace for example a public key (or any other message to be authenticated) of a legitimate device with the one of his own choosing. We will describe the attack in a two party scenario as a special case of a multiparty scenario. Let us assume that two devices ID_1 and ID_2 want to mutually authenticate their public keys PK_1 and PK_2, respectively. Following Nguyen and Roscoe's protocol [9] the devices send their respective commitments c_1 and c_2. Attacker A blocks commitment c_2 sent by ID_2 and waits to receive the corresponding decommitment d_2. Having received c_1 and transmitted c_2, the device ID_2 considers the first phase to be completed. ID_2 enters the second phase of the protocol and sends d_2 before receiving d_1; according to Nguyen and Roscoe this is a legitimate behavior [9]. Now the attacker A blocks d_2, opens the commitment c_2 and retrieves the random number N_2 from it. This allows A to create a commitment $\widehat{c_2}$ (in which he replaces PK_2 with his own $\widehat{PK_2}$) which he finally sends to ID_1. This in the end results in the same short group authentication string (i.e. $N_1 \oplus N_2$) at the devices ID_1 and ID_2, but A has succeeded in

Table 5.1. Comparison of GAP protocol and SAS-GMA [2] in terms of communication and computation.

		GMA [bits]	GAP [bits]	Difference: GMA−GAP [bits]
Tx/Rx cost	ID	$M \cdot \lvert ID\rvert$	$M \cdot \lvert ID\rvert$	0
	c	$M \cdot \lvert K_L\rvert/2$	$M \cdot \lvert K_L\rvert/2$	0
	d	$M \cdot \big(\lvert ID\rvert + \lvert K_L\rvert\big)$	$M \cdot \big(\lvert ID\rvert + \lvert K_L\rvert/2 + \lvert PK\rvert + \lvert N\rvert\big)$	$M \cdot \big(\lvert K_L\rvert/2 - \lvert PK\rvert - \lvert N\rvert\big)$
	PK	$M \cdot \lvert PK\rvert$	0 (part of the commit/open pair)	$M \cdot \lvert PK\rvert$
				$\mathbf{M \cdot \big(\lvert K_L\rvert/2 - \lvert N\rvert\big)}$
Comput. cost	hash $h(\cdot)$	$M \cdot \lvert K_L\rvert/2 \cdot \big(\lvert ID\rvert + \lvert PK\rvert\big)$	$M \cdot \lvert K_L\rvert/2 \cdot \lvert ID\rvert$	$M \cdot \lvert K_L\rvert/2 \cdot \lvert PK\rvert$
	$commit(\cdot)$	$M \cdot \lvert K_L\rvert/2 \cdot \big(\lvert ID\rvert + \lvert K_L\rvert\big)$	$M \cdot \lvert K_L\rvert/2 \cdot \big(\lvert ID\rvert + \lvert K_L\rvert/2 + \lvert PK\rvert + \lvert N\rvert\big)$	$M \cdot \lvert K_L\rvert/2 \cdot \big(\lvert K_L\rvert/2 - \lvert PK\rvert - \lvert N\rvert\big)$
	GAS	0	0	0
		$\lvert K_L\rvert = 2 \cdot length\big(h(\cdot)\big)$	M - group size	$\mathbf{\lvert K_L\rvert \cdot M \cdot \big(\lvert K_L\rvert/2 - \lvert N\rvert\big)}$

replacing ID_2's public key (PK_2) with the one of his own choosing ($\widehat{PK}_2$). While it is relatively easy to detect this flaw in the two party scenario, the problem arises in the multiparty setting where designers usually fail to realize that the ordering between exchanged messages has to be maintained between all the possible pairs of the group members. This is exactly what GAP does.

5.3.3 Communication cost: Indirect Binding vs. Direct Binding Protocols

Due to the potentially large number of sensor nodes and the requirement for a power source such as a battery, even small energy savings per device imply a significant "green potential" [160]. In this section we study potential advantages of directly over indirectly binding protocols in terms of communication cost. Nguyen and Roscoe in [9] give a simple model which compares the computation cost between various group pairing protocols. While Nguyen and Roscoe focused on the computational aspects of various group pairing protocols, we believe that in our setting it is more important to consider related communication cost, especially in low-power wireless sensor networks where the communication cost dominates the computation.

Recall from Section 5.2 in indirect binding protocols (our GAP) the short group authentication string is functionally independent of the information to be authenticated, whereas direct binding protocols require group authentication string to be dependent on the information devices want to authenticate. In direct binding protocols (e.g. SAS-GMA from [2]) this is achieved by making all group members agree on a short-output hash or digest of a complete description of the protocol run; this short-output digest is a short group authentication string. A common approach to generating a short-output digest in a provably secure way (in direct binding protocols) is to use universal hash functions. Note that the random keys required by the universal hash functions might be significantly longer than the hash output in several constructions of universal hash functions invented to date and these long keys have to be exchanged between the devices [9, 2]. At the same time in our GAP protocol we use

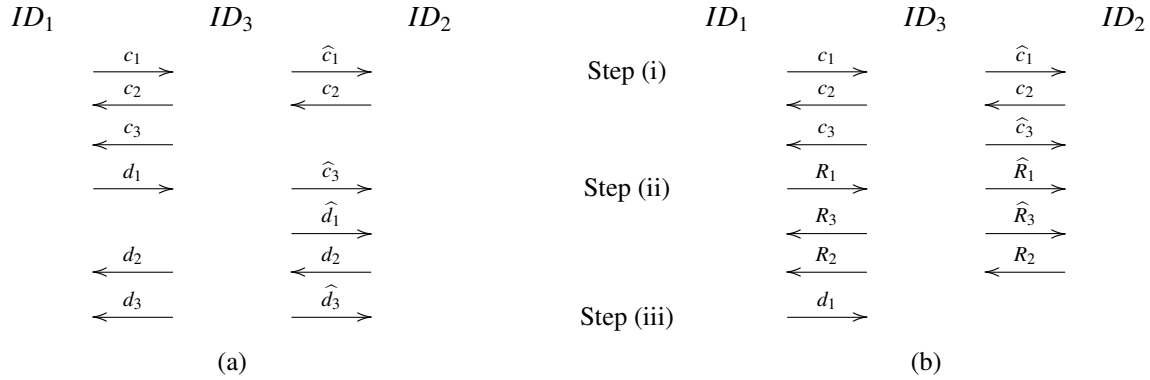


Figure 5.3. (a) An example of the insider attack on GAP. Here the attacker ID_3 wants to impersonate itself as ID_1 to the sensor device ID_2. (b) Strengthening GAP.

and communicate only short random keys/nonces (in addition to information to be authenticated). It is this difference in the key lengths that makes indirect binding schemes more efficient in terms of a communication cost.

For example, we compared the communication cost of the representative provably secure directly binding scheme proposed by Laur and Pasini [2] (the GMA protocol) and our protocol in Table 5.1. As shown in the table the advantage of our GAP protocol over GMA (expressed as the difference in the number of exchanged bits) is $M \cdot (|K_L|/2 - |N|)$ where $|K_L|/2$ represents a short-output hash or digest (as used in GMA), M being the group size and N is a short random key/nonce (as used in our GAP protocol). For example, with SHA-256 we have $|K_L| = 512$ bits [2] and $|N| = 15$ bits the advantage of GAP over GMA amounts to $241 \times M$ bits per device, i.e. for the whole group to $241 \times M^2$ (e.g. for $M = 30$ devices the difference is 26.5 KB in ideal conditions - no retransmissions).

For completeness, we also provide the comparison between the two protocols in terms of the computation cost (Table 5.1). For this we used the same computation cost model[2] as in [9]. Referring back to Table 5.1 note that $|K_L| = 2 \cdot length\big(h(\cdot)\big)$ [2]. It turns out that GAP has advantage over GMA also in terms of the computation cost, which is similar to the advantage in the communication cost.

5.4 Securing GAP Against Compromised Devices

Theorem 2 holds under the assumption that all nodes in $\mathcal{G}$ are trusted. In addition to the necessary requirement of providing security against the man-in-the-middle attacker, it is also desirable that the initialization provides protection against compromised nodes. As pointed out in [149], a manufacturer may sneak in malicious sensor node(s) along with normal sensor nodes shipped to a customer. Also, an adversary could insert his malicious code inside already deployed sensor network that requires keying with new devices brought to the field.

[2]The computation cost is as follows: (1) $cost\big(h(m)\big) \approx length\big(h(\cdot)\big) \times length(m)$ and (2) $cost\big(commit(m)\big) \approx cost\big(h(m)\big)$.

Here we show how to strengthen the GAP to withstand insider attacks (compromised sensors).

5.4.1 Insider Attack on GAP

For simplicity, let us consider the following scenario. A user wants to initialize a total of 3 sensor devices (having ID_1, ID_2 ID_3). Let us assume that the device with ID_3 is compromised and controlled by the attacker. This device will try to impersonate itself to device ID_2 as ID_1. To accomplish this, ID_3 does the following (as shown in Figure 5.3(a)). It first blocks the commitment c_1 from ID_1 to ID_2 and replaces it with its own $\widehat{c}_1$ (and thus replacing N_1 with $\widehat{N}_1$ and PK_1 with its own $\widehat{PK}_1$). Later, ID_3 sends c_3 only to ID_1 in order to trigger it to open N_1. Now, ID_3 creates $\widehat{c}_3$ (in which it commits to $N_3 \oplus N_1 \oplus \widehat{N}_1$) and sends it to ID_2. It is easy to verify that at the end all the devices will share equal $GAS = N_1 \oplus N_2 \oplus N_3$. However, ID_3 successfully replaced ID_1's public key PK_1 with the one of its own choosing $\widehat{PK}_1$.

5.4.2 Strengthening GAP Against Insider Attacks

To strengthen the basic GAP against insider attacks, we introduce an additional phase in the original protocol. More specifically, each device $i \in \mathcal{G}$ will generate a short (ℓ bits long, e.g. 15 bits) random number R_i and transmit it at the end of Phase II of the GAP over a radio channel. The purpose of this random number is to explicitly signify the completion of the Phase II on the side of the given device. The strengthened protocol is shown in Figure 5.4. Note that the Phase IV is similar to Phase III in the original GAP with the difference that instead of verifying the order in which messages are received, each device verifies random number $\widehat{R}_j$ (received in clear over a radio channel) against $\widehat{R}'_j$ extracted from the commit message $\widehat{c}_j$. *In this way, we not only mitigate the insider attack, but we also alow the devices to exchange the messages in an arbitrary order*. The cost of this solution is only ℓ bits (e.g. 15) per device. Although the messages are now exchanged in an arbitrary order the strengthened GAP remains secure (even against the attack introduced in Section 5.3.2), as we discuss next.

Let us consider again the scenario introduced in Section 5.4.1. Three nodes want to exchange some authentic information (e.g. public keys) using the strengthened GAP where the device ID_3 is compromised. As a part of the insider attack, this device will try to impersonate itself as ID_1 to ID_2 and replace public key PK_1 with one of its own choosing $\widehat{PK}_1$. In order to accomplish this the attacker blocks the commitment c_1 to the device ID_2 and replaces it with $\widehat{c}_1$ (therefore replacing N_1, R_1 and PK_1 by $\widehat{N}_1$, $\widehat{R}_1$ and $\widehat{PK}_1$) as shown in Figure 5.3(b) (step (i)). Next, the attacker sends its commitments c_3 and $\widehat{c}_3$ to the devices ID_1 and ID_2, respectively. The attacker has to make sure that $GAS_1 = GAS_2$ on devices ID_1 and ID_2, respectively. In the strengthened GAP, the devices ID_1 and ID_3 generate nonces N_1 and $\widehat{N}_1$ independently of each other, respectively. To generate $\widehat{N}_3$ (and send $\widehat{c}_3$) ID_3 has to trigger

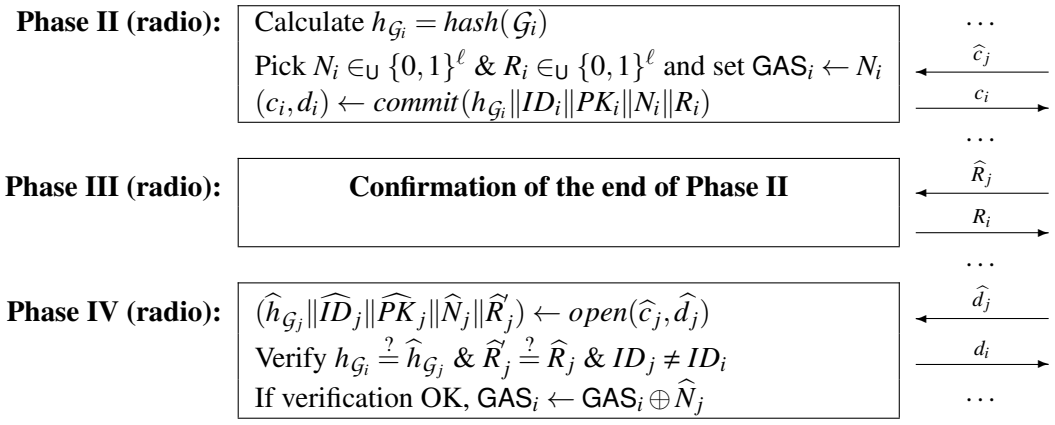


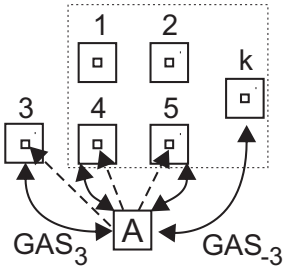
Figure 5.4. Strengthening Group message Authentication Protocol (GAP) against insider attack. Phases I and V are not shown as they are identical to the first and the last phases in the original GAP.

ID_1 to see N_1 (step (iii)). An adversary does not benefit from seeing decommitment d_1 when trying to construct $\widehat{c}_3$. Indeed, it follows that at the moment at which ID_1 revealed d_1, the device ID_1 must already have received the nonces R_2 and R_3 (step (ii)), as well as all the commitments. By sending the nonce R_i in the Phase III of the strengthened GAP, each device $i \in \mathcal{G}$ acknowledges that it has successfully received the commitments $\widehat{c}_j$ in Phase II of the strengthened GAP from the devices $j \in \mathcal{G}$. In this way, the device i will send its decommitment d_i only after all the devices j previously acknowledged (with R_j) their successful reception of the commitments.

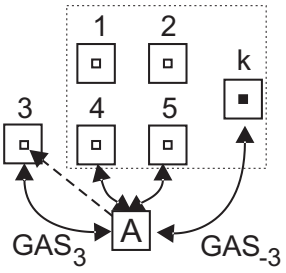
Manipulating the group size. The important security consideration of GAP is that the user is required to enter the group size into a single arbitrary sensor (e.g. a coordinator). However, a compromised coordinator could manipulate the entered group size. This is true for any protocol that falls in this category of protocols [10, 161, 157]. The simplest solution for this problem would be to assume that at least one device is not compromised and that user enters the group size into each device [161, 157].

5.5 Securing a Visible Light Channel

In this section we first describe possible attacks on messages transmitted over a visible light channel when on-off keying modulation is used. We then show how to secure the transmission of a GAS over a semi-authenticated VLC.



(a)



(b)

GAS_{ki}	GAS_{-3i}	GAS_{3i}	Attack
1	0	0	yes
1	0	1	no
0	1	0	yes
0	1	1	yes

(c)

Figure 5.5. The attacker A, with the aid of a laser, tries to (a) modify both GAS_3 and GAS_{-3} to match each other, and (b) modify only GAS_3 to match the fixed GAS_{-3}. The dashed arrows represent transmissions by the attacker using directed light source.

5.5.1 Attacks on VLC and Preventive Mechanisms

No encoding of the GAS

In this scenario, the coordinator and the other sensor devices would simply transmit the GAS in its original form via VLC. Let us consider the scenario shown in Figure 5.5(a). Here, an attacker runs the Phases I-III of the GAP protocol and establishes two different group authentication strings, GAS_3 with device 3 and GAS_{-3} with the remaining devices. From Theorem 2, it follows $\mathbf{P}[GAS_3 = GAS_{-3}] \leq 2^{-\ell} + \varepsilon$. If $\ell = 15$ or 20 bits, most likely GAS_3 and GAS_{-3} will differ with a high probability. Normally, this will be detected by the user in Phase IV of the GAP. However, in the semi-authenticated model of VLC, the adversary can flip bits 0 to 1 using a directed light source (e.g., a laser pointer). By flipping all the bits 0 to 1 in both GAS_3 and GAS_{-3} the user will see all 1s on all the sensor devices and wrongly conclude that the verification is successful. Please note that all 1s is a legitimate GAS.

Manchester and Berger Coding

We can mitigate the above bit flipping attack by using **Manchester** coding ($0 \rightarrow 01$ and $1 \rightarrow 10$). Manchester encoded GAS contains an equal number of 0s and 1s. To verify the GAS, a user would have to count the number of 0s and 1s in the transmitted sequence and also verify that the sequence has at most 2 subsequent bits set to 0 or 1. This solution doubles the size of GAS and requires the user to perform additional verifications (count the number of 0s and 1s, verify that the sequence has at most 2 subsequent 1s and 0s). Clearly, this is not an optimal option given that usability is severely deteriorated. In Figure 5.6(a) we compare different encodings wrt their impact on both usability and security. Clearly, the Manchester coding appears in the upper left corner (marked with the square). In an attempt to increase the usability of the GAS verification procedure we could use Berger codes [162].

The **Berger** code is a well know unidirectional error detecting code. Berger codes can detect any unidirectional error in a given codeword. Unidirectional errors are errors that only

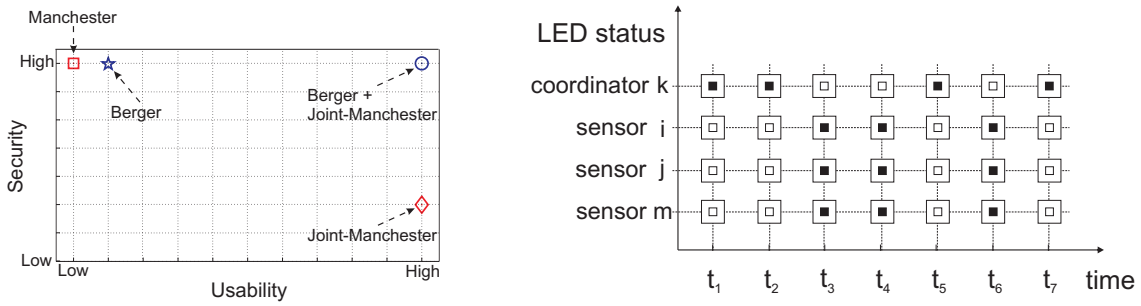


Figure 5.6. (a) Security vs. Usability tradeoff of the proposed solutions for the GAS verification via VLC, (b) Joint Manchester coding for GAS=0011010. The LEDs on sensor devices i, j and m always occupy the opposite state of the coordinator k (a colored box indicates the LED is ON).

flip zeros into ones or only ones into zeros, but not both at the same time. Let us consider a binary string (vector) $\mathbf{s}$ of size ℓ bits. Then a Berger coded string $\mathbf{s}$ (denoted $\mathsf{Ber}(\mathbf{s})$) is defined as follows: $\mathsf{Ber}(\mathbf{s}) \equiv \mathbf{s}||\mathbf{s_B}$, where $\mathbf{s_B}$ represents (in binary) the number of zeros in $\mathbf{s}$. The Berger code appends to $\mathbf{s}$ the *check value* $\mathbf{s_B}$ of size $\lceil \log_2(\ell+1) \rceil$ bits, giving the Berger code of length $\ell + \lceil \log_2(\ell+1) \rceil$.

Example 1. *For* $\mathbf{s} = 1001101$, *we have* $\mathbf{s_B} = 011$ *and* $\mathsf{Ber}(\mathbf{s}) = 1001101\mathbf{011}$.

This coding is secure given that the user counted correctly the number of 0s in GAS, converted it to the binary representation and compared it successfully with the Berger check value. Clearly this is too heavy for an end user and therefore highly unusable. This places Berger coding next to Manchester coding in Figure 5.6(a).

"Joint-Manchester" Coding

We have seen that neither Manchester nor Berger coding result in a usable solution (Figure 5.6(a)). To improve the usability while trying to preserve the security, we introduce another coding scheme, termed "Joint-Manchester" coding. In this solution, the GAS is initially Manchester coded. However, each sensor device transmits only the half of the Manchester encoded GAS, according to the following rule: the coordinator and other sensor devices transmit even and odd bits of the Manchester encoded GAS, respectively. In Figure 5.6(b) we show an example of "Joint-Manchester" coding with coordinator k. Note that the devices other than k always share the same state.

The important difference from the usability point of view, with respect to regular Manchester coding, is that the user now has to only make sure that all the sensor devices other than the coordinator share the same LED state and is opposite of the LED state on the coordinator (please refer to Figure 5.6(b)). In Section 5.7, we show that "Joint-Manchester" coding can be easily verified by the user. This solution significantly decreases the time required to

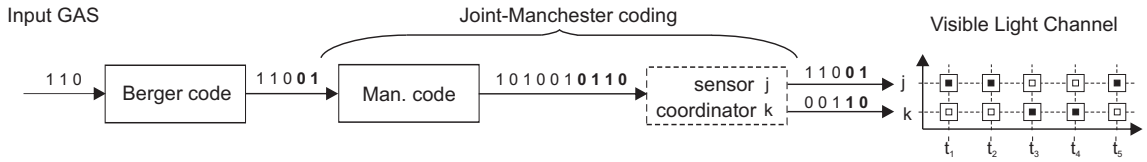


Figure 5.7. An example of the GAS verification via VLC using Berger-Manchester encoding. Labels j and k stand for jth sensor device and the coordinator, respectively.

transmit the Manchester encoded GAS (to only ℓ bits, ℓ being the size of GAS). It does not put any additional effort on the user.

How secure is the "Joint-Manchester" coding? To answer this question, let us consider the scenario shown in Figure 5.5(b). Here, the attacker runs Phases I to III of the GAP protocol and establishes two likely different GAS values, namely, GAS_3 with device 3 and GAS_{-3} with the remaining devices. The goal of the attacker is to flip bits of GAS_3 (using a directed light source) such that the modified GAS_3 (denoted $\widehat{\mathsf{GAS}_3}$) satisfies $\widehat{\mathsf{GAS}_3} = \mathsf{GAS}_{-3}$. Note that the attacker has no advantage in flipping bits in GAS_{-3} and/or GAS_k as the devices other than the coordinator should occupy the state that is opposite to the one of the coordinator k. The table given in Figure 5.5(c) shows possible combinations of GAS_3 and GAS_{-3} (their ith bits) that are beneficial for the attacker. Thus, if the ith bits of GAS_{-3} and GAS_3 are equal, an attacker will not need to modify them in any way. On the other hand, if the ith bits of GAS_{-3} and GAS_3 equal 1 and 0, respectively, an attacker could flip $0 \rightarrow 1$ by using the laser. If the ith bits of GAS_{-3} and GAS_3 are 0 and 1, the attacker will be unable to flip $1 \rightarrow 0$ for he cannot switch OFF an already powered ON LED.

We conclude that 3 combinations out of 4 are beneficial to the attacker (all combinations but the second one). It follows that the probability for an attacker to modify the bits is 3/4, therefore, the probability of a successful attack increases to $(3/4)^{\ell}$ as opposed to $2^{-\ell}$ (the probability of a successful attack where the attacker is unable to modify the GAS). If $\ell = 15$, the probability of a successful attack in a single attempt increases from 2^{-15} to approximately 2^{-6}. From this security analysis we conclude that the "Joint Manchester" coding is user-friendly but less secure than the basic Manchester coding. Therefore, in Figure 5.6(a) "Joint Manchester" coding appears in the bottom right corner.

Berger-"Joint Manchester" Coding

Finally, we show that by combining "Joint Manchester" coding and Berger codes (Berger-Manchester coding from here on) we obtain a highly usable and yet a secure solution (Figure 5.6(a)). As shown in Figure 5.7 the GAS is first Berger encoded and then "Joint-Manchester" encoded. Recall, with "Joint-Manchester" coding each sensor device transmits only a half of the Berger-Manchester encoded GAS, according to the following rule: the coordinator and the other sensor devices transmit even and odd bits of the Manchester en-

coded GAS, respectively. This solution significantly decreases the time required to transmit the Manchester encoded GAS (to only $\ell + \lceil \log_2(\ell+1) \rceil$ bits, ℓ being the size of GAS). As in "Joint-Manchester" coding, Berger-Manchester coding does not require the user to perform any additional task. Therefore, in Figure 5.6(a) Berger-Manchester coding appears in the upper right corner. The price that we have to pay for the increased security is the increased number of bits that the user has to verify by $\lceil \log_2(\ell+1) \rceil$ bits (due to the Berger check value). In Section 5.7 we show that Berger-Manchester coding can be easily verified by the user. We next prove the security of the Berger-"Joint-Manchester" coding.

Fact 1. *Let $i, j \in \mathcal{G}$ be any two sensor devices such that $GAS_i \neq GAS_j$. Then, the group authentication string GAS_k as generated by the coordinator, satisfies: $(GAS_k \neq GAS_i) \vee (GAS_k \neq GAS_j)$.*

In other words, GAS_k cannot be equal to the respective GAS values of both the sensor device i and the device j. Therefore, to detect an error (i.e., a potential attack) in the initialization process, it is sufficient to compare the GAS of the coordinator with the GAS value of each remaining sensor device.

Fact 2. *Let $a, b \in \mathbb{N}_0$ be two natural numbers (including zero) such that $a > b$, and let $\mathbf{a}, \mathbf{b} \in \{0,1\}^\ell$ be their binary representations (vectors). Then, $\exists i \in \{0, 1, \ldots, \ell-1\}$ such that $a_i > b_i$.*

We denote with $\mathsf{Man}(\mathbf{a})$ the Manchester encoded binary vector $\mathbf{a} \in \{0,1\}^\ell$. We also use notation $\mathbf{a}_{\mathbf{Odd}}$ and $\mathbf{a}_{\mathbf{Even}}$ to denote the odd and even bits of a Manchester-Berger encoded vector $\mathbf{a}$, respectively. For example, $\mathbf{a} = 1001101$, $\mathbf{a}_{\mathbf{Odd}} \leftarrow [\mathsf{Man}(\mathsf{Ber}(\mathbf{a}))]_{\mathbf{Odd}}$ and $\mathbf{a}_{\mathbf{Even}} \leftarrow [\mathsf{Man}(\mathsf{Ber}(\mathbf{a}))]_{\mathbf{Even}}$ we obtain: $\mathbf{a}_{\mathbf{Odd}} = 1001101011$ and $\mathbf{a}_{\mathbf{Even}} = 0110010100$. Finally, we use $\mathbf{1}$ as a shorthand notation for the vector comprising all ones (the vector length should be clear from the context) and $\oplus$ to denote bitwise addition modulo 2. Next, we state our main result in this section.

Theorem 3. *Let $\mathbf{a}, \mathbf{b} \in \{0,1\}^\ell$ be two arbitrary but different ℓ-bit binary vectors ($\mathbf{a} \neq \mathbf{b}$). It is not possible to modify $\mathbf{a}_{\mathbf{Odd}} \leftarrow [Man(Ber(\mathbf{a}))]_{\mathbf{Odd}}$ and $\mathbf{b}_{\mathbf{Even}} \leftarrow [Man(Ber(\mathbf{b}))]_{\mathbf{Even}}$ using only unidirectional changes of the type $0 \rightarrow 1$ such that the resulting binary vectors, denoted $\widehat{\mathbf{a}}_{\mathbf{Odd}}$ and $\widehat{\mathbf{b}}_{\mathbf{Even}}$, respectively, satisfy $\widehat{\mathbf{a}}_{\mathbf{Odd}} \oplus \widehat{\mathbf{b}}_{\mathbf{Even}} = \mathbf{1}$.*

Proof. Let us denote with $\mathsf{Hw}(\cdot)$ the Hamming weight of a given binary vector. Considering the binary vectors $\mathbf{a}, \mathbf{b} \in \{0,1\}^\ell$ for which $\mathbf{a} \neq \mathbf{b}$, we can distinguish the following three cases:

$$\begin{array}{lllll} 1. & \mathbf{a} \neq \mathbf{b} & \text{and} & \mathsf{Hw}(\mathbf{a}) = \mathsf{Hw}(\mathbf{b}) \\ 2. & \mathbf{a} \neq \mathbf{b} & \text{and} & \mathsf{Hw}(\mathbf{a}) < \mathsf{Hw}(\mathbf{b}) \\ 3. & \mathbf{a} \neq \mathbf{b} & \text{and} & \mathsf{Hw}(\mathbf{a}) > \mathsf{Hw}(\mathbf{b}) \ . \end{array}$$

Case 1. Let $\mathbf{a} \neq \mathbf{b}$ and $\mathsf{Hw}(\mathbf{a}) = \mathsf{Hw}(\mathbf{b})$. If two binary vectors are not equal but have the same Hamming weight (i.e., $\mathbf{a_B} = \mathbf{b_B}$), then clearly $\exists i \in \{0,1,\ldots,\ell-1\}$ such that $a_i = 1$ and $b_i = 0$. From the definitions of $\mathbf{a_{Odd}}$ and $\mathbf{b_{Even}}$ we have $a_{\mathbf{Odd},i} \leftarrow [\mathsf{Man}(a_i)]_{\mathbf{Odd}} = [(1,0)]_{\mathbf{Odd}} = 1$ and $b_{\mathbf{Even},i} \leftarrow [\mathsf{Man}(b_i)]_{\mathbf{Even}} = [(0,1)]_{\mathbf{Even}} = 1$. Since only unidirectional changes ($0 \rightarrow 1$) are allowed, it is not possible to modify neither $a_{\mathbf{Odd},i}$ nor $b_{\mathbf{Even},i}$. Therefore, $\widehat{a}_{\mathbf{Odd},i} = \widehat{b}_{\mathbf{Even},i}$ and consequently $\widehat{\mathbf{a}}_{\mathbf{Odd}} \oplus \widehat{\mathbf{b}}_{\mathbf{Even}} \neq \mathbf{1}$.

Case 2. Let $\mathbf{a} \neq \mathbf{b}$ and $\mathsf{Hw}(\mathbf{a}) < \mathsf{Hw}(\mathbf{b})$. From $\mathsf{Hw}(\mathbf{a}) < \mathsf{Hw}(\mathbf{b})$ it follows that $\mathbf{a}$ has a larger number of zeros than $\mathbf{b}$ and hence the larger Berger check value, i.e., $a_B > b_B$ (in base-10 notation). Since both a_B and b_B are from $\mathbb{N}_0$, it follows from Fact 2 that $\exists i \in \{0,1,\ldots,\lceil \log_2(\ell+1) \rceil - 1\}$ such that $a_{Bi} = 1$ and $b_{Bi} = 0$. Now, using the same reasoning as in the first case (**Case 1**), it follows that in this case, too, we have $\widehat{\mathbf{a}}_{\mathbf{Odd}} \oplus \widehat{\mathbf{b}}_{\mathbf{Even}} \neq \mathbf{1}$.

Case 3. Let $\mathbf{a} \neq \mathbf{b}$ and $\mathsf{Hw}(\mathbf{a}) > \mathsf{Hw}(\mathbf{b})$. From $\mathsf{Hw}(\mathbf{a}) > \mathsf{Hw}(\mathbf{b})$ it follows directly that $\exists i \in \{0,1,\ldots,\ell-1\}$ such that $a_i = 1$ and $b_i = 0$. Therefore, following the same steps as in the first case (**Case 1**), we can conclude that in this case, too, we have $\widehat{\mathbf{a}}_{\mathbf{Odd}} \oplus \widehat{\mathbf{b}}_{\mathbf{Even}} \neq \mathbf{1}$.

Thus, in all the possible cases we have $\widehat{\mathbf{a}}_{\mathbf{Odd}} \oplus \widehat{\mathbf{b}}_{\mathbf{Even}} \neq \mathbf{1}$ (given $\mathbf{a} \neq \mathbf{b}$ and the unidirectional changes $0 \rightarrow 1$). □

In other words, the Berger-Manchester coding is secure in the model where the attacker can only flip bits 0 into 1 on a visible light channel (semi-authenticated channel).

5.6 Implementation Details

In this section we discuss some implementation aspects of the proposed GAP based secure initialization mechanism, which are relevant for the usability study that we conduced. More precisely, we describe a simple method for entering the group size into the coordinator (Phase IV of GAP). Note also that the user needs to know the status of each device through the initialization process. In our implementation, this is accomplished using the sensor's LED.

State Diagram

The user needs to know the status of each device through the initialization process. For this reason and given minimal hardware requirements on sensor devices, each state will be represented using the sensor's LED. The state diagrams for a sensor and coordinator device are shown in Figures 5.8 and 5.9, respectively.

Sensor Devices State Diagram

The sensor can take one of four states in the initialization process: Uninitialized, Ready, Initialized and Confirmed.

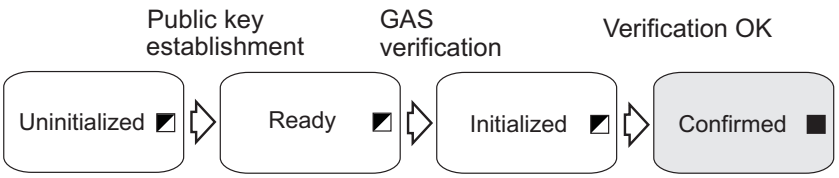


Figure 5.8. Sensors's state diagram. A colored box indicates that the LED is ON, while half colored that the LED is blinking.

Uninitialized state. Initially, after the sensors are powered ON, they occupy the Uninitialized state. A sensor remains in this state until the end of Phase III. This state is indicated with continuous blinking of the corresponding LED. At the end of Phase III (assuming all verifications are successful) the sensor advances to the Ready state, which is also indicated with the continuous blinking of the sensor's LED. The user is notified about the transition between Uninitialized and Ready states through the coordinator, as described later.

Ready state. In this state a sensor waits for the coordinator to initiate the GAS verification. This state is indicated by the blinking LED (Phase IV). If the verification does not start within a predefined time period (e.g., a few seconds), the sensor returns back to the Uninitialized state. If the GAS verification is successful, the sensor advances to the Initialized state.

Initialized state. A sensor enters the Initialized state after completing a transmission of the GAS over a visible light channel. The completion of the GAS transmission is also indicated by the blinking LED (but with a faster blinking rate compared to the Uninitialized and Ready state).

Confirmed state. In this state all the sensors (including the coordinator) hold mutually authenticated messages (e.g., public keys).

Coordinator's State Diagram

The coordinator shares Uninitialized, Initialized and Confirmed (Figure 5.9) states with the other sensors. However, its state diagram differs in several states that reflect the coordinator's main role, namely, verifying the group size.

Ready state. After the Uninitialized state the coordinator advances to the Ready state, which is indicated by a LED powered OFF. In this state the coordinator waits for the user to enter the group size. If the verification does not start within the predefined time period (e.g., a few seconds), the coordinator returns back to the Uninitialized state. The push on the coordinator's button initiates the procedure for the Verification of the Group Size.

Verification of the Group Size. If the verification is successful, the coordinator advances to the Number OK state, that is indicated by the blinking LED. The process of the group size verification is described in details in the following section. Otherwise, if the number of devices entered by the user does not match the group size $|\mathcal{G}_k|$ (seen by the coordinator),

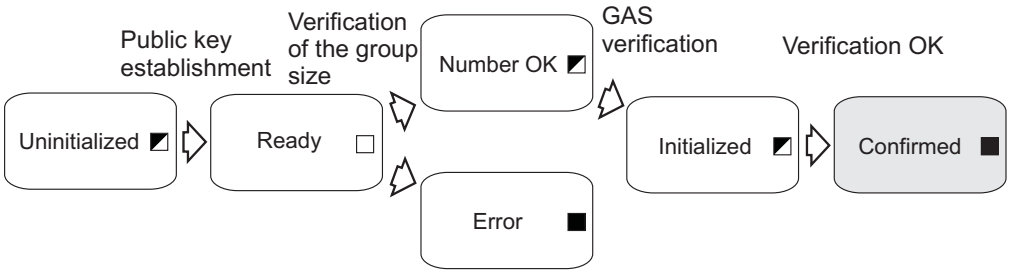


Figure 5.9. State diagram of the coordinator.

the coordinator will advance to the Error state indicated by the LED powered ON. After the successful verification, a short push on the button initiates the GAS transmission via VLC.

Entering the Group Size and Synchronization. It is essential for the security of GAP that at least one device, from the group of devices being initialized, knows the correct group size. The procedure goes as follows. Let us assume that the user wishes to initialize $M < 100$ devices. We can represent M using decimal notation as follows: $M = M_1||M_2$, where $M_i \in \{0,1,...,9\}$ (for example, if $M = 32$, then $M_1 = 3$ and $M_2 = 2$). Next, the user takes the coordinator (indicated with the LED powered OFF; other devices blink) and initiates the procedure for entering the group size with a short push on the button ($push_0$ in Figure 5.10(a)). In turn, the coordinator's LED powers ON which indicates to the user that the coordinator is ready to accept the first digit of M ($M_1 = 3$ in our example). To enter the first digit the user pushes the button M_1 times ($push_1$, $push_2$ and $push_3$ in Figure 5.10(a)). After that, the user waits the predefined time period Δt_1 for the LED to blink once (the LED powers subsequently OFF and ON as shown in Figure 5.10(a)). This blink indicates to the user that the coordinator is ready to accept the second digit of M (i.e., M_2). Again, the user enters the second digit by pushing the button M_2 times ($push_4$ and $push_5$) and waits the predefined time period Δt_1 (Figure 5.10(a)). After the coordinator "concludes" that the user has entered the second digit, the coordinator assembles the group size by concatenating two digits and compares the result with the group size that the coordinator has learned from the Phase I of the GAP. If the two match, the coordinator advances to the Number OK state (continuous LED blinking). Otherwise, the coordinator enters the Error state (constantly powered LED ON).

Synchronization. The coordinator initiates simultaneous and synchronized transmission of Berger-Manchester encoded GAS over VLC on all devices. The synchronization can be achieved by having the coordinator send SYNC messages over a radio channel to the other devices. Any attempt of jamming or injecting synchronization messages will result in desynchronization among sensor devices. This is eventually detected by the user because the sensor's LED blinks fast in all states but during the GAS transmission. Indeed, the fast blinking of a LED will overlap (in time) with much slower GAS transmission; in our implementation a single LED pulse during the GAS transmission is 4 seconds long. In future,

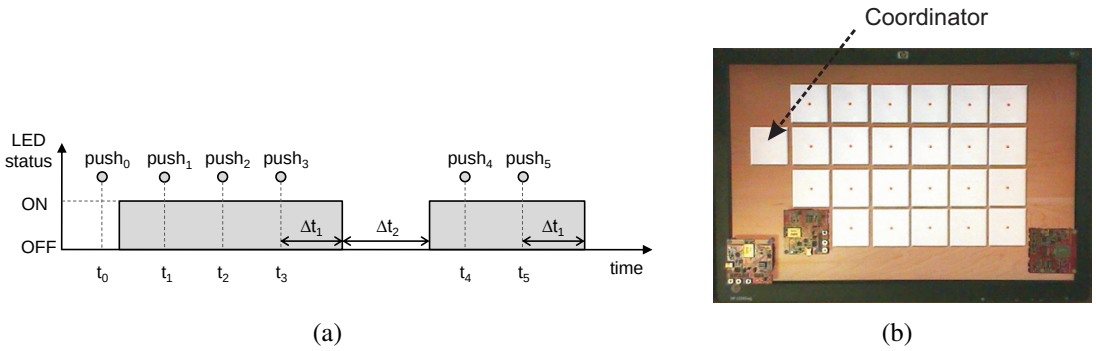


Figure 5.10. (a) An example of the group size verification for $M = 32$ devices. Experimental setup: (b) a 22-inch monitor (placed horizontally) featuring 25 sensor devices.

we plan to study these aspects in greater detail. A similar approach to synchronizing GAS transmissions appears in Prasad and Saxena [163].

5.7 Usability Evaluation

Experimental setup. Our focus in this preliminary study was to verify the thesis that Berger-Manchester coding (the GAS verification) is easy to interpret (perform) for an end user. In addition we evaluated the procedure for entering the group size into the coordinator (Figure 5.10(b)). For this purpose we implemented a simple simulator called BlinkTest (Figure 5.10(b)). The BlinkTest allows us to simulate different scenarios in which sensor devices are placed on arbitrary virtual surfaces (e.g., a desk office as shown in Figure 5.10(b)). In our study we arranged sensor devices in four rows and six columns; this is logical decision when working with a large number of devices. BlinkTest also allows us to choose different casings for sensor devices. In this study a simplistic white casing with one red LED (Figure 5.10(b)) was used. As can be seen from Figure 5.10(b), the size of a virtual sensor device matches the one of a real ZigBee sensor device (6x6 cm^2). In BlinkTest we can configure virtual sensor to blink arbitrary GAS values in synchrony. A user in our study interacts with the simulator via the 22-inch monitor that is placed horizontally on a office desk (Figure 5.10(b)). The user uses a mouse to select the coordinator (Figure 5.10(b)); the left mouse click simulates the pressing of a sensor's button.

Test cases. We first tested the users' capability to correctly enter the group size into the coordinator. We used the following group sizes: 7, 10, 16 and 25 sensor devices. In the second round of usability tests we studied the ability of users to perform GAS verification and their ability to detect intentionally introduced errors. In these tests the blinking period was set to 4 seconds. The length of the GAS as transmitted over visible light channel was 19 bits.

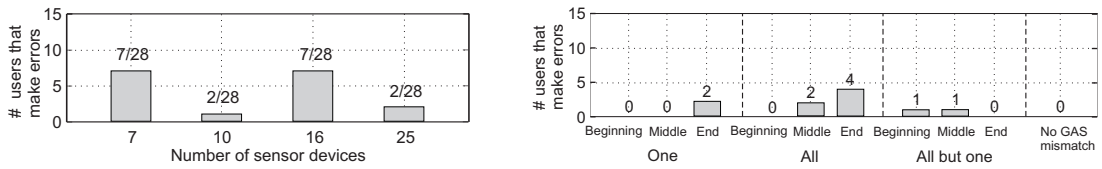
We created eleven test cases for the GAS verification, which we divide into four cate-

Table 5.2. The testers' demographic info as well as computer and mobile devices usage.

Age		**Sex**		**Eyesight**		**Using Computer (hours/week)**			
18-25	> 25	Male	Female	No glasses/contacts	Glasses/contacts	> 5	6-15	15-30	< 30
28	0	22	6	18	10	1	8	11	8
Using Internet (hours/week)				**Using mobile device pairing**		**Feel secure while using wireless**			
> 5	6-15	15-30	< 30	Y	N	Agree	Not agree	Neutral	Don't know
5	11	11	1	24	4	15	3	7	3

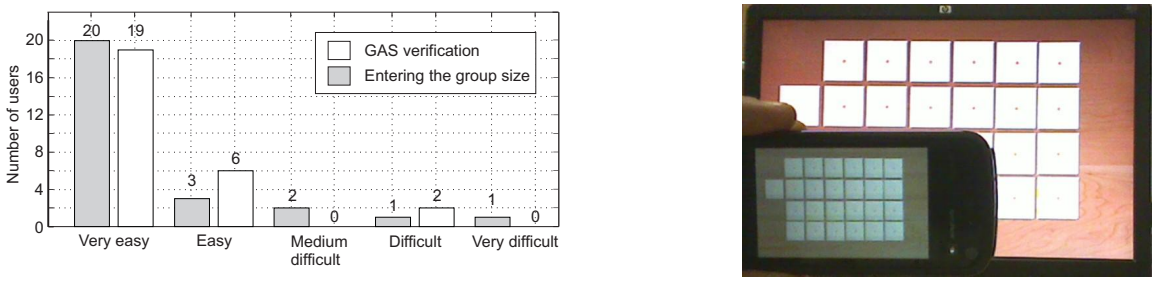
gories: (1) GAS mismatch between the coordinator and a single sensor node in: (1.1) the first bit, (1.2) a middle bit and (1.3) the last bit, (2) GAS mismatch between the coordinator and all sensor devices in: the (2.1) first bit, (2.2) a middle bit and (2.3) the last bit and (3) GAS mismatch between the coordinator and all the remaining sensor devices but one in: (3.1) the first bit, (3.2) a middle bit and (3.3) the last bit. At the end, we tested the scenario with (4) no GAS mismatches. The tests involved a 25 sensor devices (Figure 5.10(b)).

Procedure. A total of 28 participants took part in the usability study. The testers were given a short introduction to the initialization procedure, which involved the description of the node's state diagram, possible applications of such a pairing scenario (access points, ad-hoc networks, smart homes etc.). None of the participants have taken part in any of our tests before. All the participants were in their early twenties. Table 5.2 summarizes the participants' demographic information as well as information about their everyday usage of computers and mobile devices. The usability test is divided into two phases: a training phase and a testing phase. The training phase served the purpose of teaching the participants (i) how to enter the group size on the coordinator and (ii) how to perform the GAS verification on sensor devices. The training phase lasted for about 5 minutes. In the testing phase the participants performed the actual test. At the end of every usability test, the participants completed a post-test questionnaire, which involved the System Usability Scale [49] to numerically express their subjective opinion about the usability of the tested procedures.



(a) (b)

Figure 5.11. Testers that made mistake: (a) while entering the group size, (b) for particular GAS verification test case.



(a) (b)

Figure 5.12. (a) User feedback on the usability of the initialization protocol. (b) Zero-configuration auxiliary device: Using a smart phone equipped with a camera to assist the initialization of a larger number sensor devices.

Results of the Study

Each of the 28 participants performed 4 test cases for testing the purpose of the procedure for entering the group size and 10 test cases for testing the GAS verification procedure, leading to a total of 392 test cases.

Entering the group size. Each user was asked to enter once each of the following numbers 7, 10, 16 and 25, while using the procedure presented in Section 5.6 (Figure 5.10(a)). As the results in Figure 5.11(a) show, some users experienced problems while entering 7 and 16. In the first case the users had to enter two digits: 0 and 7. It turned out that the users would miss completely to enter the first digit 0. The high error rate with the group size of 16 is due to the fact the users confused the $push_0$ event in Figure 5.10(a) with the $push_1$. The average time for entering the group size was around 16, 14, 17 and 18 seconds for 7, 10, 16 and 25 devices, respectively. Please note that these included times Δt_1 and Δt_2 in Figure 5.10(a). Evaluating a usable-security application with young and educated participants is a natural first step (an application that does not fare well with them is unlikely to be acceptable by other samples of population), and therefore, our study only represents a preliminary evaluation. Future work is needed to evaluate the method with a sample representative of a larger population.

GAS verification via VLC. The testers were asked to observe the sensor devices on the display as shown in Figure 5.10(b), and to indicate (through a keyboard) if and when the status of the coordinator's LED (the isolated sensor device in Figure 5.10(b)) is incompatible with the LED status on the other sensor devices. Recall that by the Berger-"Joint Manchester" coding the status of the LED on the coordinator must always be opposite of the status of the LED on all the other sensor devices (Section 5.5.1). Each test case included the initialization of 25 sensor devices (Figure 5.10(b)). In Figure 5.11(b) we plot the number of testers that make mistakes for different GAS verification scenarios. Overall, we see a reasonably low error rates. As expected, we observe the highest error rate when GAS mismatch

occurs at the end of the GAS transmission (2 errors in the scenario when only one device has incompatible GAS with the coordinator and 4 errors in the scenario when all the devices have incompatible GAS values with the coordinator). The reason for this is that the users become less focused towards the end of the GAS verification phase. Somewhat higher error rate in the scenario where all the sensors have incompatible GAS with the coordinator is also understandable, as in this case all the sensor devices share the same LED state; it is much easier to detect the incompatible GAS on only one device. Finally, from the last test case (No GAS mismatch) we conclude that there were no false positives in our study. Note that the duration of the GAS verification phase can be calculated by multiplying the duration of the LED pulse (4 seconds in our implementation) with the size of the encoded GAS (19 bits in our case). In our study this amounts to approximately 80 seconds. Given that the user initializes 25 sensor devices this amounts to around 3.2 seconds per device.

Questionnaire. At the end of the usability tests the users were asked to fill in the post-test questionnaire from which the System Usability Score (SUS) [49] was calculated. The average SU-score for 28 users was 80,8 (out of 100). Finally, Figure 5.12(a) summarizes the users' answers on the questions related to difficulty of the procedure for entering the group size and detection of mismatches in the GAS verification phase. As shown, most of the users found these two procedures relatively easy to use.

Improving Usability and Scalability with a Zero-Configuration Auxiliary Device. In some situations the user may want to initialize even larger number of nodes than we considered in this chapter (e.g. more than 100 in several batches). Note that there is a certain limit to the number of devices that can be initialized in one batch because of the constrained nature of the devices as well as that of the human operator. Therefore, to significantly improve scalability, usability and reduce likelihood of errors we can use a camera on a smartphone if available, to record a group authentication string (GAS) transmitted using LEDs, as shown in Figure 5.12(b). Most existing solutions that involve cameras, such as [157, 151, 11, 12, 8, 155, 10, 156] require video processing or pattern recognition services installed and preconfigured with the camera devices. On the contrary, in the case of our GAP protocol, the GAS verification aided by a smartphone requires no special services or configuration on the side of the smartphone. All that is required from the user is to record a the GAS procedure (Figure 5.12(b)) and review it as many times as necessary to make sure that LED of the group members at all times occupy the opposite state than the coordinator device. We have shown before that this is an easy task for the user thanks to the Berger-Manchester coding.

5.8 Proof of Theorem 2

A user wishes to initialize a set of M sensor devices. We denote this group with $\mathcal{G}$, i.e., $\mathcal{G} = \{ID_1, ID_2, \ldots, ID_M\}$, ID_i being the identity of the ith device (all IDs are unique). In

the proof, we assume that the adversary does not belong to the set $\mathcal{G}$ (i.e., no sensor device from $\mathcal{G}$ is compromised). We further assume that each device has an access to a perfect random number generator. We denote with k the coordinator sensor device. Our security proof is based on the notion of *matching conversations* introduced by [164]. Informally, we say that devices $i, j \in \mathcal{G}$ $(i \neq j)$ have matching conversations, if for each message m_i (m_j) sent out by i (j) at the time instant t_i (t_j), the device j (i) received the same (unaltered) message $m_{ij} = m_i$ $(m_{ji} = m_j)$ at the time instant t_{i+1} (t_{j+1}), where $t_i < t_{i+1}$ $(t_j < t_{j+1})$. Clearly if all pairs of devices from $\mathcal{G}$ have matching conversations, then all messages transmitted must have arrived at intended destinations unaltered. In other words, the messages are authentic.

We say that all sensor devices from $\mathcal{G}$ "Accept" (shortly "All accept") if all the verifications in the GAP protocol (Figure 5.2) are successful. Let us further define an event S as follows:

$$S \triangleq \{\underbrace{\text{Exists non-matching}}_{S_0}, \underbrace{\text{All accept}}_{S_1}\} = \{S_0, S_1\}. \tag{5.1}$$

The event S says that there exists a pair of devices from $\mathcal{G}$ that does not have matching conversations and at *the same time* all the verifications in the GAP protocol (Figure 5.2) are successful. In other words, there exists a sensor device $i \in \mathcal{G}$ that has accepted a potentially altered message as an authentic one. Therefore, we can define the probability of a successful attack on the GAP protocol as $\mathbf{P}[S]$.

Let us introduce some additional notation. We denote with $view_i$, $i \in \mathcal{G}$ the ordered set comprising all commitments received by the device i, including its own c_i. More precisely, $view_i = \{\widehat{c}_{ID_1 i}, \widehat{c}_{ID_2 i}, \ldots, \widehat{c}_{ID_{i-1} i}, c_i, \widehat{c}_{ID_{i+1} i}, \ldots, \widehat{c}_{ID_{M_i} i}\}$, where $(M_i - 1)$ is the number of commitments received by the device i. To avoid somewhat cumbersome notation, we drop the ID from each index so that finally we have $view_i = \{\widehat{c}_{1i}, \widehat{c}_{2i}, \ldots, \widehat{c}_{i-1 i}, c_i, \widehat{c}_{i+1 i}, \ldots, \widehat{c}_{M_i i}\}$. Please note that the set $view_i$ is ordered with respect to the sender identities. The following fact follows directly from the Phase IV of the GAP protocol.

Fact 3. *In a successful attack, the number of commitments received by the coordinator k must be $M-1$, implying, $|view_k| = |\mathcal{G}_k| = M$.*

We next state the following useful result (we omit a straightforward proof for the lack of space).

Lemma 1. *If we have non-matching conversation(s) and all the devices from $\mathcal{G}$ "Accept" then either $\exists i, j \in \mathcal{G}$ such that $view_i \neq view_j$* or *otherwise all the devices "Accept" with the negligible probability ε_0.*

We continue our proof by introducing another event denoted A:

$$A \triangleq \{\exists (i, j) \in \mathcal{G} \text{ s.t. } view_i \neq view_j\}\ . \tag{5.2}$$

Then we can bound the probability of a successful attack ($\mathbf{P}[S]$) as follows:

$$\begin{aligned}\mathbf{P}[S] &= \mathbf{P}[S|A]\cdot\mathbf{P}[A] + \mathbf{P}\left[S|\overline{A}\right]\cdot\mathbf{P}\left[\overline{A}\right] \\ &\leq \mathbf{P}[S|A] + \mathbf{P}\left[S|\overline{A}\right] \\ &\overset{(1)}{\leq} \mathbf{P}[S|A] + \varepsilon_0 \\ &\overset{(2)}{=} \mathbf{P}[S_1|A] + \varepsilon_0\end{aligned} \tag{5.3}$$

where (1) follows from Lemma 1 and (2) from the fact that the event A implies that we will have for sure non-matching conversation(s). From the definition of event S_1 and by applying the probability product rule we obtain the following bounds on $\mathbf{P}[S_1|A]$, $\forall i \in \mathcal{G}\backslash\{ID_k\}$, k being the coordinator device:

$$\mathbf{P}[S_1|A] \leq \mathbf{P}\left[\mathsf{GAS}_k = \mathsf{GAS}_i | A, S_{1-(\mathsf{GAS}_k=\mathsf{GAS}_i)}\right] , \tag{5.4}$$

where $S_{1-(\mathsf{GAS}_k=\mathsf{GAS}_i)}$ denotes that all verifications in the GAP protocol, other than $\mathsf{GAS}_k = \mathsf{GAS}_i$, are successful.

Fact 4. *If $\exists i, j \in \mathcal{G}$ s.t. $view_i \neq view_j$, then the following holds for the coordinator $k \in \mathcal{G}$: $view_k \neq view_i$ or/and $view_k \neq view_j$.*

Then from Fact 4 we know that if the event A has occurred, then there exists $i \in \mathcal{G}$ such that $view_i \neq view_k$. Let i denote such a device. We have two possibilities for devices $(i,k) \in \mathcal{G}$, either $M_i = M_k$ or $M_i \neq M_k$, that is, $M_i = M$ or $M_i \neq M$ (from Fact 3 $M_k = M$). From this and the bounds in equations (5.3) and (5.4) one can easily establish (using the law of total probability) the following bound on the probability of a successful attack $\mathbf{P}[S]$:

$$\mathbf{P}[S] \leq \max\begin{cases}\mathbf{P}\left[\mathsf{GAS}_k = \mathsf{GAS}_i | A, S_{1-ki}, (M_i = M)\right] \\ \mathbf{P}\left[\mathsf{GAS}_k = \mathsf{GAS}_i | A, S_{1-ki}, (M_i \neq M)\right]\end{cases} , \tag{5.5}$$

where S_{1-ki} is a shorthand notation for $S_{1-(\mathsf{GAS}_k=\mathsf{GAS}_i)}$.

What remains to be shown is that both probabilities on the right in inequality (5.5), are bounded above by $2^{-\ell}$ plus some negligible probability. For simplicity and due to page limitations, we focus only on the case $(M_i \neq M)$.

Condition $M_i \neq M$**.** By definition (Section 5.3), $\mathsf{GAS}_i = N_i \oplus \widehat{N}_{-i}$, where $\widehat{N}_{-i} \triangleq \oplus_{j\in G_i\backslash\{i\}}\widehat{N}_{ji}$. Similarly, $\mathsf{GAS}_k = N_k \oplus \widehat{N}_{-k}$, with $\widehat{N}_{-k} \triangleq \oplus_{j\in G_k\backslash\{k\}}\widehat{N}_{jk}$. Note that we must have $|G_i| = M_i$ and $|G_k| = M$, because by assumption all verifications other than $\mathsf{GAS}_k = \mathsf{GAS}_i$ are successful. In other words, the number of commitments received in Phase II of GAS has to match the number of *ID*s received in Phase I, otherwise the protocol is aborted by i and/or k before reaching the final phase of GAS. Now we can write the

following:

$$\mathbf{P}\left[\mathsf{GAS}_k = \mathsf{GAS}_i | A, S_{1-ki}, (M_i \neq M)\right] \tag{5.6}$$

$$= \mathbf{P}\left[N_k = \widehat{N}_{-k} \oplus N_i \oplus \widehat{N}_{-i} | A, S_{1-ki}, (M_i \neq M)\right] \tag{5.7}$$

$$= \mathbf{P}\left[N_i = \widehat{N}_{-i} \oplus N_k \oplus \widehat{N}_{-k} | A, S_{1-ki}, (M_i \neq M)\right] . \tag{5.8}$$

In order to show that the probability (5.6) is bounded by $2^{-\ell}$ (plus a negligible probability), we will next show that N_k is essentially independent of $\widehat{N}_{-k}$ and N_i, and $\widehat{N}_{-i}$, or otherwise N_i is independent of $\widehat{N}_{-i}$ and N_k, and $\widehat{N}_{-k}$.

Independence (N_i, N_k). By the GAP protocol, devices i and k generate N_i and N_k independently of each other.

Independence (N_k, N_{-k}) and (N_i, N_{-i}). Note that the GAP protocol induces a total order on the set of exchanged messages (see Section 5.3). The fact that we condition probability (5.6) on the event S_{1-ki} implies that all verifications other than $(\mathsf{GAS}_k = \mathsf{GAS}_i)$ are successful. From the device i's perspective, a properly ordered exchange of messages with another device $j \in \mathcal{G} \backslash \{i\}$ looks as follows: $\widehat{d}_{ji} \succ d_i \succ \widehat{c}_{ji} \succ c_i$ for $(i < j)$, that is, $d_i \succ \widehat{d}_{ji} \succ c_i \succ \widehat{c}_{ji}$ for $(i > j)$, where the binary operator $\succ$ implies that a variable on the left side succeeds in time the variable on the right side.

We claim that an adversary does not benefit from seeing decommitment d_i when trying to construct $\widehat{c}_{ji}$. Indeed, it follows from S_{1-ki} that at the moment at which i revealed d_i, the device i must already have received all commitments (including a candidate for $\widehat{c}_{ji}$) in the proper order. Any commitment that succeeds d_i contradicts to S_{1-ki}.

Therefore, we ask ourselves: Can the adversary generate $\widehat{N}_{ji}$ (as a part of $\widehat{c}_{ji}$) such that it is related to N_i (from c_i) when only the commitments are available? We consider the scenario where $i < j$ (similar analysis applies to $j < i$). Having seen c_i, the adversary has only four options: **(i)** set $\widehat{c}_{ji} = c_i$, **(ii)** try generate a related commitment such that the covariance $Cov(\widehat{N}_{ji}, N_i) > 0$, **(iii)** break the hiding property of $commit(\cdot)$, and of course **(iv)** try to guess N_i.

In case **(i)**, the adversary has to make sure that the ID_i that appears in c_i (see Section 5.3) is changed into some different value, otherwise i aborts the protocol (i.e., S_{1-ki} has not occurred). Because $\widehat{c}_{ji} = c_i$, this can be done only by altering the corresponding d_i to obtain $\widehat{d}_{ji} \neq d_i$. From S_{1-ki} we know that $\widehat{d}_{ji} \neq d_i$ opens c_i, implying that the commitment scheme is broken. This can happen only with a negligible probability ε_c. Case **(ii)**) implies a successful attack on the non-malleable commitment scheme, which can happen with probability at most ε_c. In case **(iii)** the adversary learns the value of N_i from c_i only with the negligible probability ε_c, thanks to the (computationally) hiding property of the used commitment scheme. Finally, in case **(iv)**, the probability of success is clearly $2^{-\ell}$, ℓ being the length of N_i.

By summing up all the probabilities we conclude that the adversary can relate an arbitrary

$\widehat{N}_{ji}$ (from $\widehat{N}_{-i}$) to N_i with the probability that is at most $2^{-\ell}+3\cdot\varepsilon_c$.

Independence (N_k,N_{-i}) and (N_i,N_{-k}). As before, S_{1-ki} implies that all verifications other than $(\mathsf{GAS}_k=\mathsf{GAS}_i)$ are successful. So both devices k and i see well ordered messages. For device k, a proper exchange of messages with any two devices $j,m\in\mathcal{G}_k$ such that $m<k<j$ is as follows: $\widehat{d}_{jk}\succ d_k\succ\widehat{d}_{mk}\succ\widehat{c}_{jk}\succ c_k\succ\widehat{c}_{mk}$. Similarly, for device i and any two devices $j,m\in\mathcal{G}_i$ such that $m<i<j$ we have $\widehat{d}_{ji}\succ d_i\succ\widehat{d}_{mi}\succ\widehat{c}_{ji}\succ c_i\succ\widehat{c}_{mi}$. We ask ourselves if an adversary can relate $\widehat{N}_{ji}$ and/or $\widehat{N}_{mi}$ ($\widehat{N}_{jk}$ and/or $\widehat{N}_{mk}$) to N_k from c_k (N_i from c_i).

As before, we claim that the adversary does not benefit from seeing any decommitments. Let us consider the moment when the first decommitment is sent (revealed). Let this be d_k (similar analysis applies to any decommitment). At this moment, the adversary learns N_k and can adjust accordingly $\widehat{N}_{ji}$ such that $Cov(\widehat{N}_{ji},N_k)>0$. Given this, can the adversary generate $\widehat{N}_{mk}$ such that $Cov(\widehat{N}_{mk},N_i)>0$ by waiting to receive d_i? We can show that this is not possible. Indeed, from the proper order of messages as seen by the devices k and i above, this attack creates the following two temporal dependencies: $\widehat{c}_{ji}\succ d_k\succ c_k\succ\widehat{c}_{mk}$ and $\widehat{c}_{mk}\succ d_i\succ\widehat{c}_{ji}\succ c_i$. By combining these two temporal chains, we arrive at the following contradiction: $\widehat{c}_{ji}\succ\widehat{c}_{ji}$. Thus, it is not possible to simultaneously relate both $\widehat{N}_{ji}$ to N_k and $\widehat{N}_{mk}$ to N_i. This is valid for any possible combinations of decommitments.

Using similar analysis as in the previous case ("Independence of (N_k,N_{-k})"), we can show that before seeing any decommitment the adversary cannot relate $\widehat{N}_{ji}$ N_k with the probability higher than $2^{-\ell}+3\cdot\varepsilon_c+\varepsilon_h$. The only difference wrt the approach taken in "Independence of (N_k,N_{-k})" is that attacks where the adversary tries to set $\widehat{c}_{ji}=c_k$ are prevented by including $hash(\mathcal{G}_k)$ in c_k and $hash(\mathcal{G}_i)$ in $\widehat{c}_{ji}$ (where $|G_k|\neq|\mathcal{G}_i|$ as $M_i\neq M$); this is the reason for the appearance of the probability (ε_h) of finding a collision for $hash(\cdot)$.

From the analysis of independence between N_i, N_k, $\widehat{N}_{-i}$ and $\widehat{N}_{-k}$ and expressions (5.7) and (5.8) it readily follows that:

$$\mathbf{P}\left[\mathsf{GAS}_k=\mathsf{GAS}_i|A,S_{1-ki},(M_i\neq M)\right]\leq 2^{-\ell}+3\cdot\varepsilon_c+\varepsilon_h\ . \tag{5.9}$$

Finally, from the fact that similar analysis can be carried out for the case $M_i=M$ and expression (5.5), it easily follows that $\mathbf{P}[S]\leq 2^{-\ell}+\varepsilon$, where $\varepsilon\triangleq 3\cdot\varepsilon_c+\varepsilon_h$.

5.9 Related Work

Many existing key (pre-)distribution schemes for wireless networks rely on unspecified secure key initialization mechanisms. Here, we overview existing initialization mechanisms.

In Resurrecting Duckling [148], a physical contact is required to securely establish a secret key. It requires specialized hardware and may not scale well. Similarly, Talking to Strangers [120] requires specialized setup hardware (e.g. audio or infrared) in order to setup

a public key. Seeing Is Believing uses an installation device with a camera or a bar code reader to create an out-of-band secure channel [151]. Key authenticity is achieved through certified public keys.

In Shake Them Up [143], user establishes a secret key between two nodes by holding and shaking the devices together while they send identical packets over the radio. This scheme may be violated by using radio fingerprinting. The three related schemes are Are You With Me [144], Smart-Its Friends [145] and [146]. Mayrhofer and Welch [152] also use an out-of-band laser channel constructed with off the shelf components for transmitting short authentication strings. According to [152], the proposed solution does not ensure complete authenticity of the the laser channel. Roman and Lopez [153] discuss general aspects of communication over a visible light channel.

In Key Infection [147], two nodes establish a secret key by sending it in the clear over radio. They assume an attacker is unable to eavesdrop all the keys from all the nodes (e.g., 10.000 nodes) during key deployment. In Message In a Bottle [149] and KALwEN [165], keys are sent in the clear to the nodes located inside a Faraday cage (a specialized hardware) that ensures key secrecy and authenticity. However, the number of simultaneously initialized nodes determines the size of the Faraday cage. In On-off Keying, the presence of an RF signal represents a binary 1, while its absence represents a binary 0 [5]. By using an unidirectional encoding scheme, On-off Keying ensures that an attacker is unable to modify a packet during transmission.

In the paper, Wong and Stajano [156] present device pairing and group key agreement multichannel protocols that use communication over a radio and an out-of-band channel (e.g. visual). However, their protocol requires each device to be capable of demodulating signals received over an OoB channel (i.e., they have to be equipped with a camera). In HAPADEP [150] both data and verification information is sent over an audio channel. The pairing devices are both required to have speakers and microphones. In a related set of papers, Saxena and Uddin [8, 11], Saxena et. al. [12] and Perković et. al. [155] present device pairing methods based on devices equipped with LEDs and a video camera as the receiver. Li et. al. [10] also propose a protocol for the initialization of the large number of sensor devices that can be operated by a human. However, their protocol is insecure in the attacker model where an adversary performs flipping attacks in semi-authentic VLC.

In GAnGS [157] and SPATE [161] protocols for the secure exchange of authenticated messages among a group of N users are proposed. While GAnGS requires $O(N)$ interactions to authenticate the exchanged data, in SPATE each group member carries out N comparisons in parallel to authenticate other members' data.

5.10 Summary

We made several contributions in this chapter. We proposed a novel multichannel protocol, called *Group message Authentication Protocol* (GAP), for user-friendly initialization of multiple resource-constrained wireless devices. The proposed protocol has minimal hardware requirements on the wireless devices: one LED and one button. Moreover, as an indirect binding scheme [9] GAP has a lower communication cost compared to existing *direct* binding protocols. GAP involves communication over a bidirectional radio channel and an unidirectional out-of-band visible light channel. The proposed protocol is shown to be secure in the very strong attacker model, where an attacker can eavesdrop, jam and modify transmitted messages on both the radio and the visible light channel. We also introduced a novel coding scheme (Berger-Manchester combination) for the secure communication over semi-authentic Visible Light Channel. Finally, we demonstrated the feasibility of the proposed initialization method via the usability study that indicates that the method has reasonably low execution time, minimal error rate and is user-friendly.

6 DISSERTATION CONTRIBUTION

In this section we outline contributions to this thesis and discuss some important directions on the future work. In the area of cognitive user-authentication protocols the contributions are the following:

- To the best of our knowledge, this thesis presents the first public report about a successful (timing) attack on a secure authentication method based on non-uniform human behavior (e.g. response time, error rates, mental computation). Any form of non-uniform human behavior has to be carefully investigated to ensure an acceptable balance between security and usability of secure authentication methods.
- A paradigm that can be followed in the design and performance evaluation of any future secure authentication methods.
- Design of completely new cognitive authentication methods that are secure against mentioned timing attacks.

We have made several contributions to finding a solution against a relay attack:

- We designed a protocol against relay attacks developed for financial transactions and proved its security in a formal model (random oracle model).
- The proposed solution presents the first practical and user-friendly protocol that operates on the application layer and does not require any significant hardware changes to the existing equipment. As such, the proposed solution is oblivious to the underlying communication technologies (being wireless such as near-field communication - NFC technologies ultrasound, intra-body communication, WiFi, Bluetooth).

Most existing approaches work either with a small number of wireless devices (i.e., two) or otherwise rely on the presence of an auxiliary device (such as Faraday cage). In this thesis we designed a solution that allows user-unaided initialization (free from auxiliary devices) of a relatively large number of wireless devices. In the context of secure mutual authentication of wireless devices, the contributions to our thesis are the following:

- We designed new protocols for the initialization of multiple resource constrained wireless devices and proved their security in a formal model. We used a paradigm based

on multichannel protocols in which information is transmitted over both a radio and a visible light channel (VLC).

- We proposed a much stronger attacker model (possibile unidirectional changes in optical channel), where we speak of a semi-authenticated visible light channel. We proposed a new coding scheme that renders messages transmitted over optical channel to be secure within the proposed attacker model. We also proved the security of the new coding scheme in the formal model.

Directions for Future Work

Considering the fact that human behavior can be nonuniform and highly nonlinear in many aspects, the exploitation space of attack based on human behavior may be much larger than what we think of. Note that timing attack may not be the only form of human behavior based attacks on human authentication systems. One direction would be to investigate response times, error rates, personal preference (e.g., selecting a particular set of figures as a password), mental computation (e.g., addition, substraction, multiplication), facial expression and hand/body movement that may lead to human behavior attacks on human authentication systems. In our future work we will investigate if similar human behavior based attacks exist in other human authentication systems.

Related to mafia fraud attacks, our multichannel-based solution presented in Chapter 3 is suitable for scenarios that involve (paper) receipts such as ATM money withdrawals and payment terminals. It would be interesting to see more practical solutions that can be applied on contactless systems (e.g., WiFi, NFC, Bluetooth, IBC) without any significant modifications of the existing equipment.

Our proposed mechanisms for secure authentication of multiple wireless devices in Chapters 4 and 5 require significant user involvement prior to (setting up the camera) or during the initialization procedure (e.g., comparing the LED status on every device during the transmission over a VLC). It would be interesting to find and implement solutions based on the concept of multichannel protocols that require minimal user involvement (e.g., just powering ON the devices). Also, within the context of wireless sensor networks we would like to find user-friendly protocols that, in addition to the initial key bootstrapping, solve the problem of node addition to the existing network and node removal.

Bibliography

[1] F. Stajano, F.-L. Wong and B. Christianson, Multichannel Protocols to Prevent Relay Attacks, *Proceedings of the 14th international conference on Financial Cryptography and Data Security*, FC'10, 4–19, 2010.

[2] S. Laur and S. Pasini, SAS-Based Group Authentication and Key Agreement Protocols, *PKC, 11th International Workshop on Practice and Theory in Public-Key Cryptography*, 2008.

[3] H. Sasamoto, N. Christin and E. Hayashi, Undercover: Authentication Usable in Front of Prying Eyes, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, 183–192, 2008.

[4] M. Hasegawa, N. Christin and E. Hayashi, New Directions in Multisensory Authentication, *Proceedings of the 5th Symposium on Usable Privacy and Security*, SOUPS, 2009.

[5] M. Čagalj, S. Čapkun and J. Hubaux, Key Agreement in Peer-to-Peer Wireless Networks, *Proceedings of the IEEE Special Issue on Cryptography and Security*, 94, 2, 467–478, 2006.

[6] S. Vaudenay, Secure Communications Over Insecure Channels Based on Short Authenticated Strings, *Proceedings of the 25th Annual International Conference on Advances in Cryptology*, CRYPTO'05, 309–326, 2005.

[7] G. T. Wilfong, Method and Appartus for Secure PIN Entry, *Lucent Technologies, Inc., Murray Hill, NJ, U. S. Patent, Ed. United States*, 1999.

[8] N. Saxena and M. B. Uddin, Automated Device Pairing for Asymmetric Pairing Scenarios, *Proceedings of the 10th International Conference on Information and Communications Security*, ICICS '08, 311–327, 2008.

[9] L. H. Nguyen and A. W. Roscoe, Authentication Protocols Based on Low-bandwidth Unspoofable Channels: A Comparative Survey, *J. Comput. Secur.*, 19, 1, 139–201, Jan. 2011.

[10] M. Li, S. Yu, W. Lou and K. Ren, Group Device Pairing Based Secure Sensor Association and Key Management for Body Area Networks, *Proceedings of the 29th Conference on Information Communications*, INFOCOM'10, 2651–2659, 2010.

[11] N. Saxena and M. B. Uddin, Blink 'Em All: Scalable, User-Friendly and Secure Initialization of Wireless Sensor Nodes, *Proceedings of the 8th International Conference on Cryptology and Network Security*, CANS '09, 154–173, 2009.

[12] N. Saxena, M. B. Uddin and J. Voris, Universal Device Pairing Using an Auxiliary Device, *Proceedings of the 4th Symposium on Usable Privacy and Security*, SOUPS, 56–67, 2008.

[13] A. J. Menezes, S. A. Vanstone and P. C. V. Oorschot, Handbook of Applied Cryptography, 1996.

[14] F. Tari, A. A. Ozok and S. H. Holden, A Comparison of Perceived and Real Shoulder-surfing Risks Between Alphanumeric and Graphical Passwords, *Proceedings of the second Symposium on Usable Privacy and Security*, 56–66, 2006.

[15] R. J. Anderson, Why Cryptosystems Fail, *Commun. ACM*, 37, 11, Nov. 1994.

[16] M. Jakobsson and S. Myers, *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*, Wiley-Interscience, 2006.

[17] M. Sanford, Computer Viruses and Malware by John Aycock, *SIGACT News*, 41, 1, 44–47, Mar. 2010.

[18] K. Dunham, *Mobile Malware Attacks and Defense*, Syngress Publishing, 2009.

[19] D. Davis, F. Monrose and M. K. Reiter, On User Choice in Graphical Password Schemes, *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, 11–11, 2004.

[20] R. Kuber and W. Yu, Authentication Using Tactile Feedback, *Interactive Experiences, HCI, London, UK*, 2006.

[21] B. Malek, M. Orozco and A. E. Saddik, Novel Shoulder-Surfing Resistant Haptic-based Graphical Password, *Proceedings of the EuroHaptics Conference (2006)*, 6, 115Ű–122, Canadian Information Processing Society, Jan. 2006.

[22] X. Bai, W. Gu, S. Chellappan, X. Wang, D. Xuan and B. Ma, PAS: Predicate-Based Authentication Services Against Powerful Passive Adversaries, *Proceedings of the 2008 Annual Computer Security Applications Conference*, ACSAC '08, IEEE Computer Society, 2008.

[23] A. Bianchi, I. Oakley, J. K. Lee and D. S. Kwon, The Haptic Wheel: Design & Evaluation of a Tactile Password System, *Proceedings of the 28th of the International Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, 3625–3630, ACM, 2010.

[24] A. De Luca and B. Frauendienst, A Privacy-Respectful Input Method for Public Terminals, *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*, NordiCHI '08, 455–458, ACM, 2008.

[25] F. A. Alsulaiman, J. Cha and A. Saddik, User Identification Based on Handwritten Signatures with Haptic Information, *Proceedings of the 6th International Conference on Haptics: Perception, Devices and Scenarios*, EuroHaptics '08, 114–121, Springer-Verlag, 2008.

[26] A. Bianchi, I. Oakley, V. Kostakos and D. S. Kwon, The Phone Lock: Audio and Haptic Shoulder-Surfing Resistant PIN Entry Methods for Mobile Devices, *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '11, 197–200, ACM, 2011.

[27] A. Bianchi, I. Oakley and D. S. Kwon, The Secure Haptic Keypad: A Tactile Password System, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, 1089–1092, ACM, 2010.

[28] A. De Luca, E. von Zezschwitz and H. Hussmann, VibraPass: Secure Authentication Based on Shared Lies, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, 913–916, ACM, 2009.

[29] T. Deyle and V. Roth, Accessible Authentication via Tactile PIN Entry, *Computer Graphics Topics*, 24–26, 2006.

[30] R. Kuber and W. Yu, Feasibility Study of Tactile-Based Authentication, *Int. J. Hum.-Comput. Stud.*, 68, 3, 158–181, Mar. 2010.

[31] P. Corporation, Passfaces: Two Factor Authentication for the Enterprise, http://www.realuser.com/, [Online; last access 4-December-2012].

[32] N. Zbrodoff, Why is 9+ 7 Harder Than 2+ 3? Strength and Interference as Explanations of the Problem-size Effect, *Memory & Cognition*, 23, November 1994, 689–700, 1995.

[33] R. S. Siegler, The Perils of Averaging Data Over Strategies: An Example From Children's Addition, *Journal of Experimental Psychology-general*, 116, 250–264, 1987.

[34] M. H. Ashcraft and J. Battaglia, Cognitive Arithmetic: Evidence for Retrieval and Decision Processes in Mental Addition, *Journal of Experimental Psychology: Human Learning & Memory*, 4, 527–538, 1978.

[35] D. C. Geary, The Problem-size Effect in Mental Addition: Developmental and Cross-national Trends, *Mathematical Cognition*, 2, 63–94, 1996.

[36] D. C. Geary, K. F. Widaman and T. D. Little, Cognitive Addition and Multiplication: Evidence for a Single Memory Network., *Memory & Cognition*, 14, 6, 478–87, Nov. 1986.

[37] Y. Freund and R. E. Schapire, A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, *Journal of Computer and System Sciences*, 55, 1, 119–139, Aug. 1997.

[38] R. E. Schapire, The Strength of Weak Learnability, *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, 28–33, IEEE Computer Society, 1989.

[39] L. Breiman, Bagging Predictors, *Machine Learning*, 24, 2, 123–140, Aug. 1996.

[40] J. Kittler, M. Hatef, R. P. W. Duin and J. Matas, On Combining Classifiers, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20, 3, 226–239, 1998.

[41] G. Tsoumakas and I. Katakis, Multi Label Classification: An Overview, *International Journal of Data Warehousing and Mining*, 3, 3, 1–13, 2007.

[42] G. H. John and P. Langley, Estimating Continuous Distributions in Bayesian Classifiers, *Proceedings of the Eleventh conference on Uncertainty in Artificial Intelligence*, UAI'95, 338–345, Morgan Kaufmann Publishers Inc., 1995.

[43] Z. I. Botev, J. F. Grotowski and D. P. Kroese, Kernel Density Estimation via Diffusion, *Annals of Statistics*, 38, 5, 2916–2957, 2010.

[44] R. Jin and Z. Ghahramani, Learning with Multiple Labels, *Advances in Neural Information Processing Systems*, 897–904, 2002.

[45] Q. Yan, J. Han, Y. LI and R. DENG, H., On Limitations of Designing Usable Leakage-Resilient Password Systems: Attacks, Principles and Usability, *19th Network & Distributed System Security Symposium (NDSS), Distinguisged Paper Award,*, 2012.

[46] G. O. Einstein, M. A. McDaniel, M. Manzi, B. Cochran and M. Baker, Prospective Memory and Aging: Forgetting Intentions Over Short Delays, *Psychology and Aging*, 15, 4, 1–13, 2000.

[47] M. O. U. Guide, http://www.mathworks.com/, [Online; last access 4-December-2012].

[48] L. Faulkner, Beyond the Five-user Assumption: Benefits of Increased Sample Sizes in Usability Testing, *Behavior Research Methods, Instruments, & Computers*, 35, 3, 379–383, 2003.

[49] J. Brooke, SUS: A Quick and Dirty Usability Scale, *Usability Evaluation in Industry*, 189, 189–194, Taylor and Francis, 1996.

[50] R. Dhamija and A. Perrig, Deja Vu: A User Study Using Images for Authentication, *Proceedings of the 9th conference on USENIX Security Symposium - Volume 9*, SSYM'00, 4–4, USENIX Association, 2000.

[51] T. Matsumoto and H. Imai, Human Identification Through Insecure Channel, *Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'91, 409–421, 1991.

[52] C.-H. Wang, T. Hwang and J.-J. Tsai, On the matsumoto and imai's human identification scheme, *Proceedings of the 14th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'95, 382–392, Springer-Verlag, 1995.

[53] S. Li and H.-Y. Shum, Secure Human-Computer Identification against Peeping Attacks (SecHCI): A Survey. Technical report, http://www.hooklee.com/Papers/SecHCI-Survey.pdf/, 2003, [Online; last access 4-December-2012].

[54] T. Matsumoto, Human-computer Cryptography: An Attempt, *J. Comput. Secur.*, 6, 3, 1998.

[55] E. Hayashi, R. Dhamija, N. Christin and A. Perrig, Use Your Illusion: Secure Authentication Usable Anywhere, *Proceedings of the 4th Symposium on Usable Privacy and Security*, SOUPS '08, 35–45, ACM, 2008.

[56] X.-Y. Li and S.-H. Teng, Practical Human-Machine Identification over Insecure Channels, *Journal of Combinatorial Optimization*, 3, 4, 347–361, 1999.

[57] N. Hopper and M. Blum, Secure Human Identification Protocols, *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, 52–66, Springer-Verlag, 2001.

[58] L. Sobrado and J. C. Birget, Graphical passwords, *The Rutgers Scholar*, 4, 2002.

[59] S. Wiedenbeck, J. Waters, L. Sobrado and J.-C. Birget, Design and Evaluation of a Shoulder-surfing Resistant Graphical Password Scheme, *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '06, 177–184, ACM, 2006.

[60] H. Zhao and X. Li, S3PAS: A Scalable Shoulder-Surfing Resistant Textual-Graphical Password Authentication Scheme, *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops - Volume 02*, AINAW '07, 467–472, IEEE Computer Society, 2007.

[61] H. J. Asghar, S. Li, J. Pieprzyk and H. Wang, Cryptanalysis of the Convex Hull Click Human Identification Protocol, *Proceedings of the 13th International Conference on Information Security*, ISC'10, 24–30, Springer-Verlag, 2011.

[62] S. Li and H.-Y. Shum, Secure Human-Computer Identification (Interface) Systems Against Peeping Attacks: SecHCI. Cryptology ePrint Archive, Report 2005/268, 2005.

[63] H. Jameel, R. A. Shaikh, H. Lee and S. Lee, Human Identification Through Image Evaluation Using Secret Predicates, *Proceedings of the 7th Cryptographers' track at the RSA conference on Topics in Cryptology*, CT-RSA'07, 67–84, Springer-Verlag, 2006.

[64] H. Jameel, R. A. Shaikh, L. X. Hung, Y. W. Wei, S. M. Raazi, N. T. Canh, S. Lee, H. Lee, Y. Son and M. Fernandes, Image-Feature Based Human Identification Protocols on Limited Display Devices, *Information Security Applications*, 211–224, Springer-Verlag, 2009.

[65] D. Weinshall, Cognitive Authentication Schemes Safe Against Spyware (Short Paper), *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, SP '06, 295–300, 2006.

[66] P. Golle and D. Wagner, Cryptanalysis of a Cognitive Authentication Scheme (Extended Abstract), *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, S&P '07, 66–70, 2007.

[67] S. Li, H. J. Asghar, J. Pieprzyk, A.-R. Sadeghi, R. Schmitz and H. Wang, On the Security of PAS (Predicate-Based Authentication Service), *Proceedings of the 2009 Annual Computer Security Applications Conference*, ACSAC '09, 209–218, IEEE Computer Society, 2009.

[68] M. Lei, Y. Xiao, S. V. Vrbsky, C.-C. Li and L. Liu, A Virtual Password Scheme to Protect Passwords, *2008 IEEE International Conference on Communications*, 1536–1540, IEEE, 2008.

[69] S. Li, S. Khayam, A.-R. Sadeghi and R. Schmitz, Breaking Randomized Linear Generation Functions Based Virtual Password System, *Communications ICC 2010 IEEE International Conference on*, 1–6, IEEE, May 2010.

[70] V. Roth, K. Richter and R. Freidinger, A PIN-entry Method Resilient Against Shoulder Surfing, *Proceedings of the 11th ACM conference on Conference on Computer and Communications Security*, CCS '04, 236–245, ACM, 2004.

[71] T. S. B. Passfaces, http://www.realuser.com/, [Online; last access 4-December-2012].

[72] A. Forget, S. Chiasson and R. Biddle, Shoulder-Surfing Resistance with Eye-Gaze Entry in Cued-Recall Graphical Passwords, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, 1107–1110, ACM, 2010.

[73] A. Adams and M. A. Sasse, Users are Not the Enemy, *Commun. ACM*, 42, 12, 40–46, Dec. 1999.

[74] A. De Luca, M. Langheinrich and H. Hussmann, Towards Understanding ATM Security: A Field Study of Real World ATM Use, *Proceedings of the Sixth Symposium on Usable Privacy and Security*, SOUPS '10, 16:1–16:10, ACM, 2010.

[75] D. Foo Kune and Y. Kim, Timing Attacks on PIN Input Devices, *Proceedings of the 17th ACM Conference on Conference on Computer and Communications Security*, CCS '10, 678–680, ACM, 2010.

[76] Barclays, Contactless - The New and Easy Way to Pay, http://www.barclays.co.uk/Generalinformation/Contactlessthenewandeasywaytopay/P1242572081172/, [Online; last access 4-December-2012].

[77] Nexus S Android Smartphone, http://www.samsung.com/us/mobile/cell-phones/GT-I9020FSTTMB/, [Online; last access 5-November-2012].

[78] ISIS, http://www.paywithisis.com/, [Online; last access 4-December-2012].

[79] Spanish Bank Installs 'First' Contactless ATMs, http://www.nfctimes.com/news/spanish-bank-installs-first-contactless-atms, [Online; last access 4-December-2012].

[80] Naratte, Zoosh, http://www.naratte.com/, [Online; last access 5-November-2012].

[81] NCR makes wireless withdrawals in under 10 seconds at the ATM, http://www.ncr.com/newsroom/resources/mobile-cash-withdrawal-news/, [Online; last access 4-December-2012].

[82] eGo Project, http://ego-project.eu/, [Online; last access 4-December-2012].

[83] Y. Desmedt, Major Security Problems with the 'Unforgeable' (Feige)-Fiat-Shamir Proofs of Identity and How to Overcome Them, *Proceedings of the 6th Worldwide Congress on Conference on Computer and Communications Security and Protection (SecuriCom)*, 1998.

[84] M-PESA, http://www.safaricom.co.ke/, [Online; last access 4-December-2012].

[85] S. Drimer and S. J. Murdoch, Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks, *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, 7:1–7:16, USENIX Association, 2007.

[86] R. Anderson, Position Statement in RFID S&P Panel: RFID and the Middleman, *Proceedings of the 11th International Conference on Financial cryptography and 1st International conference on Usable Security*, FC'07/USEC'07, 46–49, Springer-Verlag, 2007.

[87] L. Francis, G. Hancke, K. Mayes and K. Markantonakis, Practical NFC Peer-to-peer Relay Attack Using Mobile Phones, *Proceedings of the 6th International Conference on Radio Frequency Identification: Security and Privacy Issues*, RFIDSec'10, 35–49, Springer-Verlag, 2010.

[88] L. Francis, G. Hancke, K. Mayes and K. Markantonakis, Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones, arXiv.org ePrint Archive, Report arXiv:1209.0875 [cs.CR], 2012, http://arxiv.org/abs/1209.0875.

[89] M. Roland, Applying Recent Secure Element Relay Attack Scenarios to the Real World: Google Wallet Relay Attack, Cryptology ePrint Archive, Report 2011/618, 2011, http://eprint.iacr.org/2011/618.

[90] K. B. Rasmussen and S. Čapkun, Realization of RF Distance Bounding, *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, 389–402, USENIX Association, 2010.

[91] S. Ganeriwal, C. Pöpper, S. Čapkun and M. B. Srivastava, Secure Time Synchronization in Sensor Networks, *ACM Transactions on Information and System Security (TISSEC)*, 11, 4, 23:1–23:35, Jul. 2008.

[92] S. D. N. A. Inc., Frame Grabber, http://www.sonydna.com/sdna/e/products/framegrabber/index.html/, [Online; last access 5-November-2012].

[93] B. 10, Time Shift, http://us.blackberry.com/campaigns/blackberry-10.html/, [Online; last access 5-November-2012].

[94] A. Giusti and V. Caglioti, Isolating Motion and Color in a Motion Blurred Image, *Proceedings of the British Machine Vision Conference 2007*, British Machine Vision Association, 2007.

[95] J. Jia, Single Image Motion Deblurring Using Transparency, *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 1–8, june 2007.

[96] M. Chau and M. Betke, Real Time Eye Tracking and Blink Detection with USB Cameras, *Boston University Computer Science Technical Report*, 12, 2005.

[97] D. M. H. Zhang and S. V. Raman, CAE-Based Side Curtain Airbag Design, *SAE World Congress, Detroit, Michigan*, Mar. 2004.

[98] E. H. Yilmaz and W. H. Warren, Visual control of braking: a test of the tau hypothesis, *Journal of Experimental Psychology: Human Perception and Performance*, 21, 5, 996–1014, Oct. 1995.

[99] A. D. Cambou and N. Menon, Three-dimensional Structure of a Sheet Crumpled Into a Ball, *Proceedings of the National Academy of Sciences of the United States of America*, 108, 36, 14741–14745, 2011.

[100] D. Simon, B. Aboba and R. Hurst, The EAP-TLS Authentication Protocol, RFC 5216 (Proposed Standard), March 2008.

[101] J. H. Conway, *On Numbers and Games*, AK Peters, Ltd., 2nd edn., 2000.

[102] Y. Desmedt, C. Goutier and S. Bengio, Special Uses and Abuses of the Fiat-Shamir Passport Protocol, *CRYPTO '87: A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, 21–39, Springer-Verlag, 1988.

[103] Y.-C. Hu, A. Perrig and D. B. Johnson, Wormhole Attacks in Wireless Networks, *IEEE Journal on Selected Areas in Communications*, 24, 2, 370–380, feb. 2006.

[104] Y. Desmedt, Major Security Problems with the 'Unforgeable' (Fiat-. Shamir) Proofs of Identify and How to Overcome Them, *SecuriComm '88, SEDEP Paris*, 1988.

[105] C. Cremers, K. B. Rasmussen, B. Schmidt and S. Čapkun, Distance Hijacking Attacks on Distance Bounding Protocols, *33rd IEEE Symposium on Security and Privacy, S&P 2012*, 113–127, IEEE Computer Society, 2012.

[106] R. Anderson and M. Bond, The Man-in-the-Middle Defence, *Cambridge Security Protocols Workshop*, 153–156, Springer-Verlag, 2009.

[107] S. Brands and D. Chaum, Distance-Bounding Protocols (Extended Abstract), *EUROCRYPT'93, Lecture Notes in Computer Science*, 344–359, Springer-Verlag, 1993.

[108] S. G, Z. Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor and Z. Sahinoglu, Localization via Ultra-wideband Radios: A Look at Positioning Aspects for Future Sensor Networks, *IEEE Signal Processing Magazine*, 22, 70–84, 2005.

[109] G. P. Hancke and M. G. Kuhn, An RFID Distance Bounding Protocol, *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, SECURECOMM '05, 67–73, IEEE Computer Society, 2005.

[110] N. O. Tippenhauer and S. Čapkun, ID-based Secure Distance Bounding and Localization, *Proceedings of the 14th European conference on Research in computer security*, ESORICS'09, 621–636, Springer-Verlag, 2009.

[111] S. Capkun and J.-P. Hubaux, Secure Positioning of Wireless Devices with Application to Sensor Networks, *IEEE INFOCOM*, 2005.

[112] A. Harter, A. Hopper, P. Steggles, A. Ward and P. Webster, The Anatomy of a Context-aware Application, *Proceedings of the 5th annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '99, 59–68, ACM, 1999.

[113] L. R. 4919, IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals, http://tools.ietf.org/html/rfc4919/, [Online; last access 4-December-2012].

[114] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen and D. E. Culler, SPINS: Security Protocols for Sensor Networks, *Wireless Networks*, 8, 5, 521–534, Sep. 2002.

[115] L. Eschenauer and V. D. Gligor, A Key-Management Scheme for Distributed Sensor Networks, *Proceedings of the 9th ACM conference on Conference on Computer and Communications Security*, CCS '02, 41–47, ACM, 2002.

[116] D. Liu and P. Ning, Establishing Pairwise Keys in Distributed Sensor Networks, *Proceedings of the 10th ACM conference on Conference on Computer and Communications Security*, CCS '03, 52–61, ACM, 2003.

[117] W. Du, J. Deng, Y. S. Han and P. K. Varshney, A Pairwise Key Pre-Distribution Scheme for Wireless Sensor Networks, *Proceedings of the 10th ACM conference on Conference on Computer and Communications Security*, CCS '03, 42–51, ACM, 2003.

[118] H. Chan, A. Perrig and D. Song, Random Key Predistribution Schemes for Sensor Networks, *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, SP '03, 197–, IEEE Computer Society, 2003.

[119] C. Karlof, N. Sastry and D. Wagner, TinySec: a Link Layer Security Architecture for Wireless Sensor Networks, *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, 162–175, ACM, 2004.

[120] D. Balfanz, D. K. Smetters, P. Stewart and H. C. Wong, Talking to Strangers: Authentication in Ad-hoc Wireless Networks, *Symposium on Network and Distributed Systems Security*, 2002.

[121] A. Kumar, N. Saxena, G. Tsudik and E. Uzun, Caveat Emptor: A Comparative Study of Secure Device Pairing Methods, *International Conference on Pervasive Computing and Communications (PerCom)*, 1–10, March 2009.

[122] N. Saxena and M. B. Uddin, Secure Pairing of "Interface-Constrained" Devices Resistant against Rushing User Behavior, *Proceedings of the 7th International Conference on Applied Cryptography and Network Security*, ACNS '09, 34–52, Springer-Verlag, 2009.

[123] C. Boyd and A. Mathuria, *Protocols for Authentication and Key Establishment*, Springer, 2003.

[124] M. G. Kuhn, Electromagnetic Eavesdropping Risks of Flat-panel Displays, *Proceedings of the 4th International Conference on Privacy Enhancing Technologies*, PET'04, 88–107, Springer-Verlag, 2004.

[125] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier and R. Dahab, NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks, *Proceedings of the 5th European Conference on Wireless Sensor Networks*, EWSN'08, 305–320, Springer-Verlag, 2008.

[126] L. B. Oliveira, M. Scott, J. Lopez and R. Dahab, TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks, *5th International Conference on Networked Sensing Systems (INSS)*, 2008.

[127] M. Čagalj, J. Hubaux, S. Čapkun, R. Rengaswamy, I. Tsigkogiannis and M. Srivastava, Integrity (I) Codes: Message Integrity Protection and Authentication Over Insecure Channels, *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, SP '06, 280–294, IEEE Computer Society, 2006.

[128] A. Francillon and C. Castelluccia, TinyRNG: A Cryptographic Random Number Generator for Wireless Sensors Network Nodes, *Int. Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 1–7, IEEE, 2007.

[129] R. C. Fairfield, R. L. Mortenson and K. B. Coulthart, An LSI Random Number Generator (RNG), *Proceedings of CRYPTO 84 on Advances in Cryptology*, 203–230, Springer-Verlag New York, Inc., 1985.

[130] T. E. Tkacik, A Hardware Random Number Generator, *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, 450–453, Springer-Verlag, 2003.

[131] K. D. Murray, 8-bit AVR Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash, http://www.atmel.com/, [Online; last access 4-March-2008].

[132] J. Walker, Hotbits, http://www.fourmilab.ch/random/, [Online; last access 4-March-2009].

[133] A. Rukhin, J. Soto, J. Nechvatal, E. Barker, S. Leigh, M. Levenson, D. Banks, A. Heckert, J. Dray, S. Vo, A. Rukhin, J. Soto, M. Smid, S. Leigh, M. Vangel, A. Heckert, J. Dray and L. E. B. III, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, http://csrc.nist.gov/rng/, 2001, [Online; last access 4-March-2009].

[134] K. Yuksel, J. Kaps and B. Sunar, Universal Hash Functions for Emerging Ultra-Lowpower Networks, *Proceedings of the Communications Networks and Distributed Systems Modeling and Simulation Conference*, 2004.

[135] L. Carter and M. N. Wegman, Universal Classes of Hash Functions, *Journal of Computer and System Sciences*, 18, 2, 1979.

[136] J. von Neumann, Various Techniques Used in Connection With Random Digits, *The Monte Carlo Method*, 12, 36–38, National Bureau of Standards, Applied Mathematics Series, 1951.

[137] G. Shapiro and G. C. Stockman, *Computer Vision*, Prentice-Hall, 2001.

[138] Z. Alliance, ZigBee Specification (Document 053474r06, Version 1.0), http://www3.nd.edu/~mhaenggi/ee67011/zigbee.pdf/, June 2005, [Online; last access 4-December-2012].

[139] S. Zhu, S. Setia and S. Jajodia, LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks, *Proceedings of the 10th ACM conference on Conference on Computer and Communications Security*, CCS '03, 62–72, ACM, 2003.

[140] J. Deng, C. Hartung, R. Han and S. Mishra, A Practical Study of Transitory Master Key Establishment For Wireless Sensor Networks, *First International Conference on Security and Privacy for Emerging Areas in Communications Networks SECURECOMM05*, 289–302, 2005.

[141] D. Liu, P. Ning and W. Du, Group-Based Key Pre-Distribution in Wireless Sensor Networks, *Proceedings of the 4th ACM Workshop on Wireless Security*, WiSe '05, 11–20, ACM, 2005.

[142] M. Ramkumar and N. Memon, An Efficient Key Predistribution Scheme for Ad-hoc Network Security, *IEEE Journal on Selected Areas in Communications*, 23, 3, 611–621, 2006.

[143] C. Castelluccia and P. Mutaf, Shake Them Up!: A Movement-based Pairing Protocol for CPU-constrained Devices, *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, MobiSys '05, 51–64, ACM, 2005.

[144] J. Lester, B. Hannaford and G. Borriello, "Are You with Me?" - Using Accelerometers to Determine If Two Devices Are Carried by the Same Person, *Pervasive Computing*, 3001 of *Lecture Notes in Computer Science*, 33–50, Springer Berlin Heidelberg, 2004.

[145] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl and H. W. Gellersen, Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts, *Proceedings of the 3rd International Conference on Ubiquitous Computing*, UbiComp '01, 116–122, Springer-Verlag, 2001.

[146] R. Mayrhofer and H. Gellersen, Shake Well Before Use: Two Implementations for Implicit Context Authentication, *Adjunct Proc. UbiComp 2007*, 72–75, September 2007.

[147] R. Anderson, H. Chan and A. Perrig, Key Infection: Smart Trust for Smart Dust, *Proceedings of the 12th IEEE International Conference on Network Protocols*, ICNP '04, 206–215, IEEE Computer Society, 2004.

[148] F. Stajano and R. Anderson, The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks, *Proceedings of the 7th International Workshop on Security Protocols*, 172–194, Springer-Verlag, 2000.

[149] C. Kuo, M. Luk, R. Negi and A. Perrig, Message-In-a-Bottle: User-Friendly and Secure Key Deployment for Sensor Nodes, *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, SenSys '07, 233–246, ACM, 2007.

[150] C. Soriente, G. Tsudik and E. Uzun, HAPADEP: Human-Assisted Pure Audio Device Pairing, *Proceedings of the 11th International Conference on Information Security*, ISC '08, 385–400, Springer-Verlag, 2008.

[151] J. M. McCune, A. Perrig and M. K. Reiter, Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication, *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, SP '05, 110–124, IEEE Computer Society, 2005.

[152] R. Mayrhofer and M. Welch, A Human-Verifiable Authentication Protocol Using Visible Laser Light, *Proc. ARES 2007: 2nd International Conference on Availability, Reliability and Security*, 1143–1147, IEEE CS Press, April 2007.

[153] R. Roman and J. Lopez, KeyLED - Transmitting Sensitive Data Over Out-of-Band Channels in Wireless Sensor Networks, *5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008*, 796 –801, 2008.

[154] Mica2 specifications, https://www.eol.ucar.edu/rtf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf/, [Online; last access 4-December-2012].

[155] T. Perkovic, I. Stancic, L. Malisa and M. Cagalj, Multichannel Protocols for User-Friendly and Scalable Initialization of Sensor Networks, *5th Int. ICST Conference on Security and Privacy in Comm. Networks (Securecomm)*, 228–247, 2009.

[156] F. L. Wong and F. Stajano, Multichannel Security Protocols, *IEEE Pervasive Computing, Special Issue on Security and Privacy*, 6, 4, 31–39, Oct. 2007.

[157] C.-H. O. Chen, C.-W. Chen, C. Kuo, Y.-H. Lai, J. M. McCune, A. Studer, A. Perrig, B.-Y. Yang and T.-C. Wu, GAnGS: Gather, Authenticate 'n Group Securely, *Proceedings of the 14th Annual International Conference on Mobile Computing and Networking, MOBICOM*, MobiCom '08, 92–103, ACM, 2008.

[158] R. Nithyanand, N. Saxena, G. Tsudik and E. Uzun, Groupthink: Usability of Secure Group Association for Wireless Devices, *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, Ubicomp '10, 331–340, ACM, 2010.

[159] S. Laur and S. Pasini, User-Aided Data Authentication, *International Journal of Security and Networks*, 4, 1/2, 69–86, Feb. 2009.

[160] P. Rost and G. Fettweis, On the Transmission-Computation-Energy Tradeoff in Wireless and Fixed Networks, *CoRR*, abs/1008.4565, 2010.

[161] Y.-H. Lin, A. Studer, H.-C. Hsiao, J. M. McCune, K.-H. Wang, M. Krohn, P.-L. Lin, A. Perrig, H.-M. Sun and B.-Y. Yang, SPATE: Small-group PKI-less Authenticated Trust Establishment, *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, MobiSys '09, 1–14, ACM, 2009.

[162] J. M. Berger, A Note on an Error Detection Code for Asymmetric Channels, *Information and Control*, 4, 68–73, Mar. 1961.

[163] R. Prasad and N. Saxena, Efficient Device Pairing Using "Human-Comparable" Synchronized Audiovisual Patterns, *Proceedings of the 6th International Conference on Applied Cryptography and Network Security*, ACNS'08, 328–345, Springer-Verlag, 2008.

[164] M. Bellare and P. Rogaway, Entity Authentication and Key Distribution, *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, 232–249, Springer-Verlag, 1993.

[165] Y. W. Law, G. Moniava, Z. Gong, P. H. Hartel and M. Palaniswami, KALwEN: A New Practical and Interoperable Key Management Scheme for Body Sensor Networks, *Security and Communication Networks*, 4, 11, 1309–1329, 2011.

Curriculum Vitae

Toni Perković

Toni Perković was born 31. of December 1983. in Split, Croatia. After finishing elementary school he attended general-program secondary school in Split where he graduated in 2002. After that he enrolled Electrical Engineering studies at Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture at University of Split. He received his B.Sc. degree with excellent marks in May 2007, with thesis "Application of EL FAROL BAR to simulation of communication networks" leaded by mentor dr. sc. Dinko Begušić and co-mentor dr. sc. Mario Čagalj.

He is employed as a young researcher at Electrical Engineering studies at Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, where he is actively involved in courses in the fields of cryptography and network security, wireless security, wireless sensor networks and human-computer interaction. He is a part of team involved in project "Advanced Heterogeneous Network Technologies" with leadership of dr. sc. Dinko Begušić. He is enrolled at PhD. studies at Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture at University of Split from 2007 leaded by mentor dr. sc. Mario Čagalj. His main research interests are usability, design, and analysis of security protocols for wireless networks and the design of secure authentication protocols. He is a student member of IEEE, member of ACM and a secretary of ACM Croatia.

Životopis

Toni Perković

Toni Perković rođen je 31. prosinca 1983. godine u Splitu. Nakon završene osnovne škole pohađao je srednju opću gimnaziju gdje je i maturirao 2002. godine, a iste godine upisao je studij Elektrotehnike, smjer Elektronika na Fakultetu elektrotehnike, strojarstva i brodogradnje u Splitu. Diplomirao je s izvrsnim uspjehom na smjeru Elektrokomunkacije u svibnju 2007. godine obranivši diplomski rad pod naslovom "Primjena problema EL FAROL BARA pri simulaciji komunikacijskih mreža" pod vodstvom mentora dr. sc. Dinka Begušića i komentora dr. sc. Marija Čaglja.

Nakon diplomiranja zapošljava se kao znanstveni novak na Fakultetu elektrotehnike, strojarstva i brodogradnje, te aktivno sudjeluje u nastavi na više kolegija iz područja računalne i mrežne sigurnosti, interakcije korisnika sa računalom, te je uključen u rad na znanstvenim projektu Ministarstva znanosti, obrazovanja i sporta (MZOS): "Napredne heterogene mrežna tehnologije", pod vodstvom dr. sc. Dinka Begušića. Poslijediplomski doktorski studij Elektrotehnike i informacijske tehnologije pri Fakultetu elektrotehnike, strojarstva i brodogradnje Sveučilišta u Splitu upisuje 2007. godine pod mentorstvom dr. sc. Marija Čaglja. Glavni istraživački interesi u području doktorske disertacije su mu upotrebljivost, dizajn i analiza sigurnih protokola u bežičnim mrežama te dizajn sigurnih autentifikacijskih protokola te iz tih područja objavljuje radove na međunarodnim znanstvenim skupovima i u časopisima s međunarodnom recenzijom. Toni Perković je član međunarodne organizacije IEEE i ACM. Ujedno je i tajnik udruge ACM Hrvatska.