Localisation and obfuscation techniques for enhanced multi-factor authentication in mCommerce applications

By

TORBEN KUSELER

Applied Computing Department
The University of Buckingham
United Kingdom

Thesis submitted for the Degree of Doctor of Philosophy in Computer Science to the School of Science in the University of Buckingham

May 2012

Abstract

The focus of this thesis is to investigate solutions that shall enhance the security of remote client authentication for mCommerce applications on phones such as Smartphones or Tablet-PCs. This thesis details three innovative authentication schemes developed during the course of this study. These schemes are based on the use of localisation and obfuscation techniques in combination with multi-factor authentication to enforce the knowledge of “who, when, where and how” necessary for any remote client authentication attempt. Thus, assuring the mCommerce service provider about the genuine client as well as ensuring correct capturing and processing of the client’s authentication data on the remote phone. The author of this thesis believes that these schemes, when developed on commercial mCommerce applications, shall enhance the service provider’s trust into the received client data and therefore shall encourage more service providers to offer their mCommerce services via phone applications to their clients.

The first proposed scheme, called MORE-BAILS, combines multiple authentication factors into a One-Time Multi-Factor Biometric Representation (OTMFBR) of a client, so to achieve robust, secure, and privacy-preserving client authentication. Tests and trials of this scheme proved that it is viable for use in the authentication process of any type of mCommerce phone applications.

The second and third schemes, called oBiometrics and LocAuth respectively, use a new obfuscated-interpretation approach to protect the mCommerce application against misuse by attackers as well as to ensure the real-time and one-time properties of the client’s authentication attempt. The novelty of combining biometric-based keys with obfuscated-interpretation tightly binds the correct mCommerce application execution to the genuine client. Furthermore, integration of the client’s current location and real-time in the LocAuth challenge / response scheme eliminates the risk that an attacker can illegitimately re-use previously gathered genuine client authentication data in a replay attack.

Based on appropriate criteria, the MORE-BAILS, oBiometrics and LocAuth levels of security, user-friendliness and algorithms’ ease-of-implementation are proven in experiments and trials on state-of-the-art Android-based Smartphones.

Acknowledgements

Katja

I am profoundly grateful to Katja for her understanding, patience, kindness and love. Without you, your strength and support, this dream would have been ended before it actually really started. I am so thankful that you joined, escorted and unconditionally helped me all the way.

My parents / Meinen Eltern

Meinen tiefsten Dank für die Liebe und Unterstützung die ich immer von Euch erfahren habe, sowie für alle Möglichkeiten, die Ihr mir eröffnet habt. Ohne Euch hätte ich es niemals bis hierher geschafft. Ich wusste und weiß, dass Ihr immer für mich da sein werdet. Diese Gewissheit gibt mir die innere Stärke, auch schwierige Situation zu meistern und immer daran zu glauben, dass alle Probleme, so groß und unüberwindbar sie auch erst erscheinen mögen, letztendlich doch zu bewältigen sind.

My supervisor, Dr. Ihsan Lami

My greatest thanks go to my supervisor Dr. Ihsan Lami, who continuously support me during the development of this thesis and who always encouraged and motivated me to continue even in difficult times. Without his viable advice, guidance, ideas and challenges, this thesis would have never been come to light.

Prof. Sabah Jassim

My thanks go also to Prof. Sabah Jassim, who offered me this great opportunity and brought me to Buckingham. Without him, this dream would not exist. Beside his valuable academic advice, I would like to thank him and the Department of Applied Computing for the financial support of my studies.

Prof. Dr. Uwe Schmidt

I like to express my appreciation to Prof. Dr. Uwe Schmidt (University of Applied Sciences, Wedel, Germany), who taught and supervised me in my under-/postgraduate studies and who’s enthusiastic way of teaching and conducting research impressed me so much. Thank you for the inspiration and encouragement to start my own academic career.

My friends and colleagues

Last but not least, many thanks to my friends, fellow students and all department members for all your support, valuable suggestions, and important discussions.

Abbreviations

AES	Advanced Encryption Standard
AF	Authentication Factor
API	Application Programming Interface
BCH	Bose–Chaudhuri–Hocquenghem
BDP/FE	Biometric Data Processing / Feature Extraction
BFV	Biometric Feature Vector
DTL	Data-Location-Time
DVM	Dalvik Virtual Machine
DWT	Discrete Wavelet Transform
ECC	Error Correcting Code
EER	Equal Error Rate
FAR	False Acceptance Rate
FRR	False Rejection Rate
FSM	Finite State Machine
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HC/HD	High-Capacity / High-Density
IC	Integrated Circuit
ICCID	Integrated Circuit Card International Identifier
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IT	Information Technology
J2ME	Java 2 Micro Edition
KoP	Key-On-Phone
L&T	Location and Time
LDEA	Location-Dependent Data Encryption Algorithm
LOTA	Location/Obfuscation Techniques for Authentication (this Thesis)
LSS	Location Signature Sensor
LTE	Long Term Evolution
MCSI	Mobile Communication Subscriber Information

MNO	Mobile Network Operator
MORE-BAILS	Multi-FactOR Enhanced Biometric Authentication using Independent Location Sources
oB&L	oBiometrics and LocAuth
OTMFBR	One-Time Multi-Factor Biometric Representation
OTP	One-Time Password
PBS	Password-Based Shuffling
PIN	Personal Identification Number
PMI	Permutation-Based Interpreter
PRNG	Pseudo-Random Number Generator
PV	Proximity Value
PVT	Position, Velocity, and Time
RS	Reed-Solomon
SAGA	Secure Authentication for GPS phone Applications
SHA	Secure Hash Algorithm
SIM	Subscriber Identity / Identification Module
SLC	Spherical Law of Cosines
SMS	Short Message Service
SOM	Secret Orthonormal Matrix
TD	Toleration Distance
UBRP	User-Based Random Projection
UBT	User-Based Transformation
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
USIM	Universal Subscriber Identity / Identification Module
VM	Virtual Machine
WAN	Wide Area Network
WiFi	Wireless-Fidelity
WiFi AP	Wireless-Fidelity Access Point

Table of Contents

List of Figures

List of Tables

Declaration

I, hereby declare that all the work in this thesis is my own work except where due reference is made within the text of the thesis.

I also declare that, to the best of my knowledge, none of the material has ever previously been submitted for a degree in this or any other university.

1 INTRODUCTION

Mobile devices / phones with enhanced capabilities, like Smartphones or Tablet-PCs, become a major part in everybody's daily life. All kinds of mCommerce activities, including financial banking transactions and online shopping, are nowadays performed online via phone applications whilst on the move.

The availability of mCommerce applications on phones creates new challenges in protecting these application, and therefore demands secure and reliable remote client authentication on the phone, because phones are easily lost or stolen. In this case, an attacker has full access to the phone and could illegitimately use the mCommerce application, if no application protection exists.
For the authenticator, the "remote" nature of client-authentication eliminates the immediate knowledge of "who, when, where and how" of the client's authentication attempt, which is clearly and automatically defined in office-based, face-to-face transactions. I.e. impersonation and other distance attacks (e.g. replay attacks) are more likely to happen in mCommerce than in office-based transactions. As a consequence, mCommerce service providers (e.g. financial companies) currently restrict access to their services via phone applications or they do not offer access to their services via phone applications at all.

This research work focuses on "bringing-back" the lost face-to-face features into remote client authentication on phones, as well as protecting the application execution on the client's phone. I.e. enhancing the service provider's trust into the remotely received client authentication data by guaranteeing the freshness and originality of the provided client data. This research should encourage more service providers to offer their mCommerce services via phone applications to their clients.

Integration of the face-to-face features into remote authentication is achieved by a tight and secure combination of client specific information (e.g. Biometrics, Passwords and phone specific tokens to define the "who"), with real-time (to define the "when"), and the client's current location (to define the "where") into a One-Time Multi-Factor Biometric Representation (OTMFBR) of a client.

Correctness of the authentication process execution on the client's phone (to define the "how") is ensured by a software application protection mechanism based on biometric (oBiometrics) and location (LocAuth) obfuscated interpretation. Obfuscated interpretation ensures correct execution of the authentication application and binds the application closely to the genuine client. This shall assure the authenticator about correct capturing and processing of the authentication data, as well as reduces the risk that an attacker is able to use the mCommerce application, even if the attacker is in possession of the genuine client's phone.

Usability and commercial viability are important aspects for mCommerce applications and their market success. Clients will not accept complicated authentication solutions in their everyday life. mCommerce service providers will not utilise solutions, which development and distribution costs exceed the return on investment in terms of added security or increased revenue. To achieve user-friendliness and commercial viability, all the three schemes proposed in this thesis are designed with these issues in mind, i.e. a strong focus towards development and deployment costs, flexibility and adaptability as well as user-friendliness suitable for current and future generations of phones.

1.1 Author's Research Motivation

I finished my first University degree in Commercial Information Technology from The University of Applied Sciences at Wedel in Germany. I chose this degree because I was thoroughly fascinated from the interaction of commercial aspects with our technological evolving and becoming completely mobile world. Right from the time I had the possibility to conduct my own financial transactions (e.g. online shopping) on a personal computer; I wanted to understand the technological, security and usability implications of these two areas that are more and more becoming one.

During my subsequent MSc studies at Wedel, I was offered an internship position in the Applied Computing Department at The University of Buckingham to support the research projects Broadwan [1] and SecurePhone [2]. These two projects helped streamline my interests in wireless / mobile communication technologies and secure authentication for mCommerce applications performed on phones.

In Broadwan, I helped to investigate and design new ad-hoc routing protocols for mobile devices in Wide Area Networks (WANs).
The European co-funded project (IST-2002-506883) SecurePhone aimed at realising a new mobile communication system enabling biometrically authenticated clients to deal m-contracts during a mobile phone call in an easy yet highly dependable and secure way. In SecurePhone, I helped to implement the developed algorithms on the phone. The SecurePhone prototype was developed on the HTC Qtek 2020 PDA and designed in such a way that the complete client authentication process was performed on the phone. Once a client was successfully verified, a digital signature stored on the phone's SIM card was released and used to sign the digital contract at that moment of authentication time. The digitally signed contract was then sent to the other involved contract party.
Client trails on the SecurePhone prototype showed that client authentication can be done in less than one second. This was achieved when solely the feature matching was executed on the SIM card's own processor (Match-On-Card). All other authentication steps (e.g. biometric data acquisition and pre-processing) were executed during these experiments on the host application processor of the phone. The authentication processing time increased to 45 minutes, when all the processing was executed by the SIM processor hosted inside the secure SIM card environment. The tremendous execution time increase was due to low processing power of the SIM card CPU and slow data communication between the SIM CPU / Memory and the phone resources.
Execution of the complete authentication process on the phone is an important aspect for the security evaluation of the SecurePhone approach. The authentication system security is fully based on the security features of the SIM card. The business partner receiving the digitally signed contract has no possibility to check the correctness of the signature release. S/he must trust the security mechanisms of the SecurePhone. However, if an attacker is able to get hold of the phone, the attacker has full control over the phone and SIM card and could eventually break the SecurePhone's security mechanisms and illegitimately sign contracts.

Thinking at the end of my internship about the SecurePhone approach, I identified two questions, which answers can be used to tackle the security impacts and to ensure secure "on-device" authentication for mCommerce applications:

1) Can the risk that an attacker hacks into the “on-device” authentication be reduced? I.e. investigate if the complete authentication process execution on the phone can be in some way security protected, e.g. by using a secure host environment or by protecting the authentication application execution on the phone.
2) Can part of the authentication process as well as the authenticity decision be moved from the client’s phone to the authenticator side? I.e. is it possible that the client authentication is performed remotely instead of locally on the phone? In this scenario, the authenticator does not have to trust the possibly undermined client phone. In contrast, the authenticator can base the authenticity decision fully on his equipment and authentication algorithms. However, this type of remote authentication raises further security concerns, which needed to be carefully addressed. For example, the authenticator must be able to check the freshness of the received authentication data. If this is not possible, then, for example, an attacker could re-use previously gathered genuine client data in a replay attack.

The experiences I have made, the knowledge I have gained and the people I have met during this internship at the University of Buckingham clearly showed me my interest in academic research. It became obvious to me that I would like to carry on my own research in this area of authentication and mobile communication technologies. I wanted to clearly and fully understand the underlying principles and challenges involved in client authentication as well as find answers to the open questions remaining from my internship. I.e. why is the authentication on the SecurePhone’s SIM card so slow and is it possible to enhance the security of the SecurePhone’s prototype further?

After I finished my MSc degree in Wedel, the University of Buckingham offered me a place as a PhD student in the Department of Applied Computing to conduct my own research. I was delighted by this offer and the opportunity to contribute to the research in secure and reliable client authentication on phones.

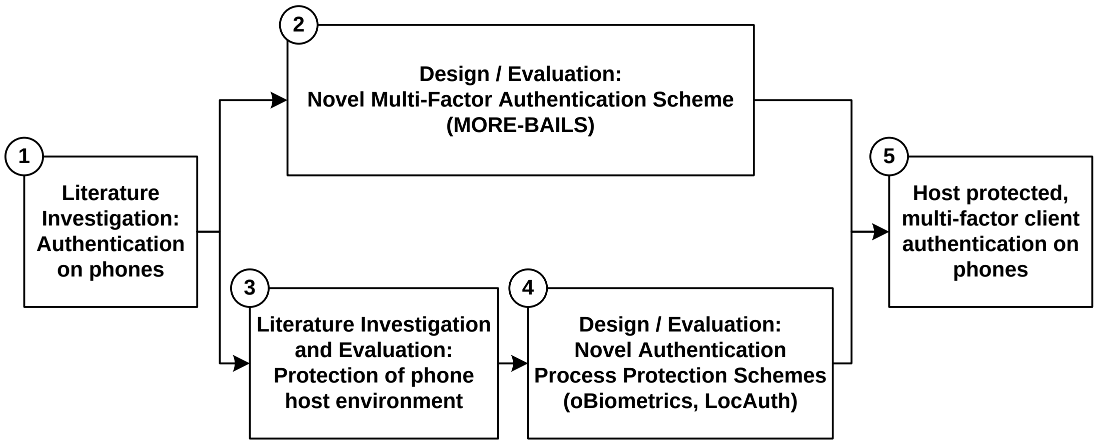
This thesis describes the knowledge I have gained and the achievements and contributions I have made during the last three years investigating “Location and Obfuscation Techniques for secure client Authentication” (LOTA) in mCommerce applications on phones. The “LOTA” acronym will be used in the remainder of this thesis to refer to this research project as a whole. I thoroughly enjoyed these last three years and would love to continue my work on LOTA in the future.

1.2 Research Methodology, Novelties and Achievements

Important to all mCommerce applications on phones that require secure and reliable remote client authentication, LOTA proposes solutions to the following questions:

1) How to achieve secure, robust, reliable, and user-friendly authentication data collection on phones to enhance the correctness of the authenticator's remotely performed authenticity decision?
2) How to protect the authentication process on the phone against malicious attacks (e.g. modification of the authentication process) to ensure the correctness of the authentication data collection, processing and transmission?

Researching these questions for a few months has led to many hypotheses for LOTA. Figure 1 shows the undertaken research methodology.



1
Literature Investigation: Authentication on phones
2
Design / Evaluation: Novel Multi-Factor Authentication Scheme (MORE-BAILS)
3
Literature Investigation and Evaluation: Protection of phone host environment
4
Design / Evaluation: Novel Authentication Process Protection Schemes (oBiometrics, LocAuth)
5
Host protected, multi-factor client authentication on phones

Figure 1: Research methodology

1) LOTA started by investigating the literature on authentication techniques, with a focus on techniques and authentication factors to secure and enhance remote client authentication on phones. Based on the conducted literature investigation, LOTA concluded about the important properties of such techniques. The literature investigation also formed the criteria to identify and develop efficient and secure algorithms for the proposed schemes.

2) Literature investigation was followed by design, implementation, trial and evaluation of a novel multi-factor authentication scheme (called MORE-BAILS) that includes a strong assurance of the client's current location and real-time.
3) LOTA has also identified, during the literature survey on authentication that malicious modification of the phone hosting the authentication process is a critical attack point in remote authentication. To fully understand the impacts and the available countermeasures against such modifications, the literature on protecting the phone as a host for the authentication process was investigated.
4) Based on the gained knowledge and the results from the evaluation of the investigated methods on secure hosting of the authentication process, novel authentication process protection techniques based on biometric and location obfuscated software interpretation (oBiometrics and LocAuth) were designed.
5) Finally, the proposed solutions (MORE-BAILS, oBiometrics, and LocAuth) were combined to achieve a host protected, multi-factor remote client authentication scheme for mCommerce application on phones.

During the work on LOTA, the following novelties were developed and proposed:

1) A novel way (called One-Time Multi-Factor Biometric Representation (OTMFBR)) to securely combine multiple authentication factors (e.g. PINs, tokens, biometrics, location and time) that allows the authenticator to reliably verify the client's claimed identity and phone, his/her current location as well as the freshness of the authentication data and the real-time property of the authentication attempt.
2) Techniques to transfer the client's actual physical location into another secure location-domain. These techniques enable the authenticator to verify the client's claimed location without actually knowing the client's current physical location, i.e. the client's location privacy is not negatively affected.
3) Two novel schemes (called oBiometrics and LocAuth) to protect the authentication application execution on the phone as well as to ensure the freshness of the authentication data. The tight binding of the correct application execution to the genuine client eliminates the risk that an attacker can use the authentication application even if the attacker steals the genuine client's phone.

In addition to the LOTA work, I had the opportunity to support fellow student researchers in their projects. This gave me in return valuable knowledge and feedback about practical issues of techniques evaluated and used in LOTA as well as new ideas for my own research. I would like to highlight exemplarily two of these opportunities and outline, how LOTA benefited from the collaborative work.

1) The first collaboration was with a colleague working on WiMax / WiFi baseband convergence techniques. I supported him in the FPGA chip implementation of his proposed method. LOTA benefited from this experience, because it increased my understanding of the impacts and requirements of custom chip implementation, one method studied and evaluated to secure the authentication process execution environment hosted on the phone.
2) The second collaboration was with a fellow PhD student working on WiFi / Bluetooth mesh networks. I was able to implement aspects of his work on the Android Smartphone operating system platform. LOTA uses Android as the platform to implement and test the proposed solutions. The gathered deep understanding of the Android system during this collaborative work helped tremendously to perform the practical trials and experiments of LOTA on Android Smartphones.

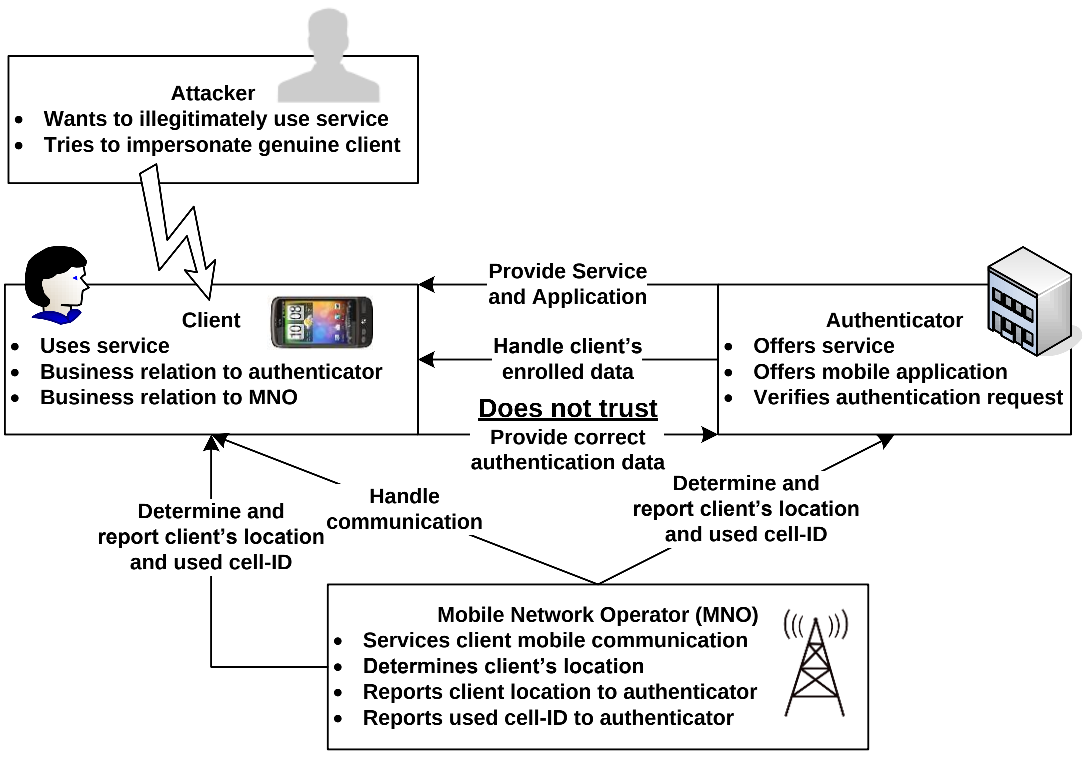
In the space of LOTA, the following papers were published:

1) Torben Kuseler and Ihsan Alshahib Lami, "Using Geographical Location as an Authentication Factor to enhance mCommerce Applications on Smartphones," *International Journal of Computer Science and Security (IJCSS)*, vol. 6, no. 4, p. 10, Aug 2012.
2) Torben Kuseler and Ihsan Alshahib Lami, "dLocAuth: a dynamic multifactor authentication scheme for mCommerce applications using independent location-based obfuscation," in Mobile Multimedia/Image Processing, Security, and Applications 2012, SPIE, Bellingham, WA, Apr 2012.
3) Torben Kuseler, Ihsan Alshahib Lami, and Hisham Al-Assam, "oBiometrics: A Software protection scheme using biometric-based obfuscation," in 2011 African Conference on Software Engineering and Applied Computing (ACSEAC), Cape Town, South Africa, Sep 2011.

4) Torben Kuseler, Hisham Al-Assam, and Ihsan Alshahib Lami, "One Time Multi Factor Biometric Representation (OTMFBR) for remote client authentication," Published Patent, Number 1109832.4, UK Intellectual Property Office, Jun 2011.
5) Torben Kuseler, Hisham Al-Assam, Sabah Jassim, and Ihsan Alshahib Lami, "Privacy preserving, real-time and location secured biometrics for mCommerce authentication," in Mobile Multimedia/Image Processing, Security, and Applications 2011, SPIE, vol. 8063, Bellingham, WA, Apr 2011.
6) Ihsan Alshahib Lami, Torben Kuseler, Hisham Al-Assam, and Sabah Jassim, "LocBiometrics: Mobile phone based multifactor biometric authentication with time and location assurance," in Proc. 18th Telecommunications Forum (IEEE TELFOR 2010), Nov 2010.
7) Torben Kuseler, Ihsan Alshahib Lami, Sabah Jassim, and Harin Sellahewa, "eBiometrics: an enhanced multi-biometrics authentication technique for real-time remote applications on mobile devices," in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 7708, Apr 2010, pp. 77080E.1-77080E.9.
8) Ali Al-Sherbaz, Torben Kuseler, Chris Adams, Roman Marsalek, and Karel Povalac, "WiMAX parameters adaptation through a baseband processor using discrete particle swarm method," International Journal of Microwave and Wireless Technologies, vol. 2, no. 02, pp. 165-171, Apr 2010.
   This paper is a joint publication with colleagues at the University of Brno, Czech Republic, as part of the COST RFCSET (IC0803) project [3].

1.3 System and Trust Model of Proposed Authentication Schemes

The three schemes proposed by LOTA (MORE-BAILS in chapter 3, oBiometrics in section 5.1, and LocAuth in section 5.2) are based on the following system and trust model shown in Figure 2. The main parties of this model are the client, the authenticator, the Mobile Network Operator (MNO), and the attacker. The icons used in Figure 2 to represent the involved parties will be consistently used in all further figures to represent the same party.



Attacker
• Wants to illegitimately use service
• Tries to impersonate genuine client
Client
• Uses service
• Business relation to authenticator
• Business relation to MNO
Provide Service and Application
Handle client's enrolled data
Does not trust
Provide correct authentication data
Authenticator
• Offers service
• Offers mobile application
• Verifies authentication request
Determine and report client's location and used cell-ID
Handle communication
Determine and report client's location and used cell-ID
Mobile Network Operator (MNO)
• Services client mobile communication
• Determines client's location
• Reports client location to authenticator
• Reports used cell-ID to authenticator

Figure 2: System and trust model

The system model in Figure 2 involves the following four parties:

1) The **authenticator** who offers a remote service or mobile application that requires secure authentication of any previously registered client. In mCommerce, the authenticator can be a bank, who offers bank account access and financial transactions via phones to their registered and enrolled clients.
2) The **client** who uses the mobile application typically provided by the authenticator on his/her phone to access the services. The client must have a previously established business connection with the authenticator and has also previously registered and enrolled all required authentication credentials and personal information (e.g. passwords and biometrics, cp. section 2.2.2). The client uses the GPS receiver on his/her phone to establish his/her current location. GPS was chosen in LOTA because of the worldwide availability of the GPS system as well as the increasing number of phones that feature a GPS receiver (cp. section 2.2.3.1). The cell-ID used in the communication between the client's phone and the mobile network (cp. section 2.2.3.1) is available to the client via an Android system call (cp. section 4.3.3.1).
3) The **attacker** who illegitimately wants to use the protected service or application. LOTA assumes that the attacker has white-box attack capabilities (cp. section

4.3.1), i.e. the phone and, as a result, the authentication application on the phone is fully available, respectively fully under control of the attacker, because the phone was stolen or unintentionally lost by the client.

4) The cellular **Mobile Network Operator (MNO)** who services the client's phone and handles the mobile network communication. The MNO also determines the client's current phone location via trilateration during normal operation (cp. section 2.2.3.1). The determined phone location and cell-ID used in the communication between client and mobile network are communicated to the authenticator to be used for independent verification of the client's claimed location. Depending on the system configuration and the established trust and required privacy level between client, authenticator and MNO, the physical location is either directly forwarded from the MNO to the authenticator or the location is prior to communication "protected" by privacy-preserving location transformations (cp. section 3.2.2.4).

LOTA assumes the following trust model between these four parties:

1) The client trusts the authenticator to provide correct functioning mobile applications. Furthermore, the client trusts the authenticator to handle all previously registered and enrolled sensitive client data (e.g. biometrics) correctly. The client also trusts the authenticator to have robust security and privacy mechanisms in place.
2) The client trusts the MNO to handle the mobile communication correctly.
3) Both, client and authenticator, trust the MNO to determine and report the client's current phone location and the used cell-ID to the authenticator accurately. The client also trusts the MNO to protect his/her real physical location before transmitting his/her location to the authenticator.
4) The authenticator does not trust any authentication data received from the client. Specially, the authenticator does not trust the client's claimed location and uses the phone location provided by the MNO to independently verify the client's location claim.

1.4 Thesis Organisation

The rest of this thesis is organised as follows; Chapter 2 gives an introduction to the authentication process. Chapter 2 furthermore outlines the requirements and security aspects, as well as available authentication factors and their limitations for secure remote authentication performed on phones. Chapter 3 presents and discusses the MORE-BAILS scheme, a user-friendly and secure multi-factor authentication scheme designed for mCommerce applications on phones. In chapter 4, three approaches to protect the phone environment, hosting the authentication process against malicious modifications are reviewed and evaluated. Based on the evaluation results, chapter 5 presents and discusses the oBiometrics and LocAuth schemes. These schemes protect and guarantee the correct authentication process execution as well as real-time of the client's authentication attempt through obfuscated interpretation. Finally, in chapter 6, this thesis concludes and makes recommendations towards future work.

2 BACKGROUND AND LITERATURE SURVEY OF AUTHENTICATION TECHNIQUES

2.1 The Authentication Process

Authentication (derived from the Greek word “αυθεντικός” / *authentes*: real, genuine, author) is the process of verifying and acknowledging the claimed properties of an entity (*Am I whom I claim I am?*). Typically, an authentication process consists of three stages [4]:

1) Enrolment stage: Authentication data (also called Authentication Factors (AFs), cp. section 2.2) describing a genuine entity (e.g. a physical object such as a human client) are enrolled and stored within the authentication system.
2) Acquisition or presentation stage: Fresh authentication data is collected to verify the entity authenticity.
3) Verification stage: Correctness of the data and claimed properties are verified by comparing the enrolled and freshly captured authentication data. Depending on the verification result; authenticity of the entity is acknowledged or denied.

Verification is a one-to-one (1-to-1) class problem. This is in contrast to identification, which is the process to find an entity in a set of similar entities, i.e. identification is a one-to-many (1-to-M) class problem. Successful verification / identification follows authorisation. Authorisation ensures that the successful verified / identified client is actually allowed to access certain resources such as applications on phones.

LOTA provides solutions for secure remote verification of known and previously enrolled clients, which use their phones to perform mCommerce applications. Thus, remote client identification will not be addressed by LOTA. “Remote” denotes in LOTA all authentication scenarios, in which client and authenticator (i.e. the authentication server) are connected through a (potentially unsecured) communication link and do not have personal (i.e. face-to-face) contact. “Phone” refers in LOTA to all small and portable devices that feature wireless communication facilities (e.g. GSM, UMTS, LTE, Bluetooth, or WiFi) as well as enhanced sensors and receivers (e.g. camera, GPS-receiver, touch-pad, accelerometer, etc.). Examples of such phones are Smartphones (e.g. iPhone, Android Galaxy Nexus) or tablet-computers (e.g. iPad or Android / Blackberry-based tablets).

2.2 Authentication Factors and Multi-Factor Authentication

During verification, client authentication schemes utilise AFs to verify the client's claimed identity. Multi-factor authentication schemes use more than one AF, preferable received via different channels or stored in different places, with the aim to make a successful attack on the authentication scheme more difficult for an attacker [5]. Instead of relying on a single AF (e.g. password), multi-factor authentication schemes require at least two pieces of valid information, e.g. password and token. If an attacker is able to get hold of one of these AFs, the attacker still requires access to the other AF to get authenticated. AFs can be categorised into classic AFs (cp. section 2.2.1) and new AFs (e.g. location, cp. section 2.2.3), which were recently proposed.

2.2.1 Classic Authentication Factors

Classic AFs can be categorised into three groups [6]:

1) Knowledge-based, or "something you know", e.g. password (cp. section 2.2.1.1).
2) Object-based, or "something you have", e.g. token (cp. section 2.2.1.2).
3) Identity-based, or "something you are", e.g. biometrics (cp. section 2.2.1.3).

2.2.1.1 Knowledge-based Authentication Factors

Knowledge-based AFs rely on a memorised piece of information, e.g. Personal Identification Number (PIN) or password. Long and random passwords can offer a high level of security in authentication systems. However, in practice, clients have huge difficulties to memorise random and strong passwords. This often results in the use of short passwords that are therefore simpler to guess. Or strong passwords are used, but as they are difficult to remember, they are written down or sent by email and thus, are available to attackers. Additionally, passwords are often re-used, which make cross-application attacks easier [7]. Four to five passwords should be the upper limit of not related, regularly used passwords, a client should need to remember [8]. However, employees already have to remember up to sixteen different passwords at their workplace [9].

"Eavesdropping" and "phishing" are further problems of knowledge-based AFs used for the protection of phones. It is likely that a password is "eavesdropped" or that a client will enter a password in a “phishing” application on his/her phone believing that s/he uses a trustworthy application because:

1) Stealing a password is easier when it is typed on the phone in a public place than espy a password typed on a protected personal computer at home.
2) Many phone applications require online access to a web resource to work or use additional online resources to enhance their services. This leads to an increasing number of authentication requests a client has to deal with during the phone application usage. Hence, the client’s attention to each single authentication request reduces, making it easier for a phishing application to steal a password.
3) The number of available applications in Smartphone markets (e.g. Apple’s App Store or Android’s market) increases with a tremendous speed and clients install more of these applications on their phone than they are doing on their personal computer at home or in the office, i.e. the risk to install a phishing application is higher.

Nevertheless, the use of PINs is the major (and most of the time only) mechanism to protect phones. However, only 66% of phones are actually protected by a PIN, leaving a third of all phones completely unprotected [10]. In addition, clients who use the PIN do not do this properly. 45% never change their PIN, 42% change the PIN only once after purchasing the phone, and 36% use the same PIN for more than one service [10]. This allows an attacker to get access to other services, if the phone’s PIN or another service with the same PIN is successfully undermined (cross-application attack).

2.2.1.2 Object-based Authentication Factors

Object-based AFs, e.g. tokens, rely on physical possessions. An object-based AF has the advantage over a knowledge-based AF that clients do not need to memorise anything. This eliminates the risk of attackers guessing passwords easily because simple passwords are used. However, the main security drawback of physical tokens is that, when lost or stolen, the attacker gains unauthorised access.

2.2.1.3 *Identity-based Authentication Factors*

Identity-based AFs, i.e. Biometrics (derived from the Greek words *bios*: life and *metrikos*: measure of) rely on the uniqueness of physiological (e.g. fingerprint, facial features) or behavioural (e.g. hand-writing, speech) characteristics of a client. Biometric-based authentication offers two advantages over the other classic AFs:

1) A legitimate client does not need to remember or carry anything.
2) Biometrics verify the de facto client and not only knowledge of a password or possession of a token, i.e. the genuine client needs to be present at the biometric sensor. This aspect is even more important in remote authentication to assure the authenticator about the genuine client performing this authentication request.

Phones are well suited for biometric-based authentication because they feature various sensors to collect any required biometric data, e.g. camera to capture the client's face or microphone to record the client's voice. However, biometric authentication systems are not perfect and their security can also be undermined as discussed in the following section 2.2.2.

2.2.2 Biometric-based Authentication

Biometric authentication systems can be divided into five main components [11] as shown in Figure 3.

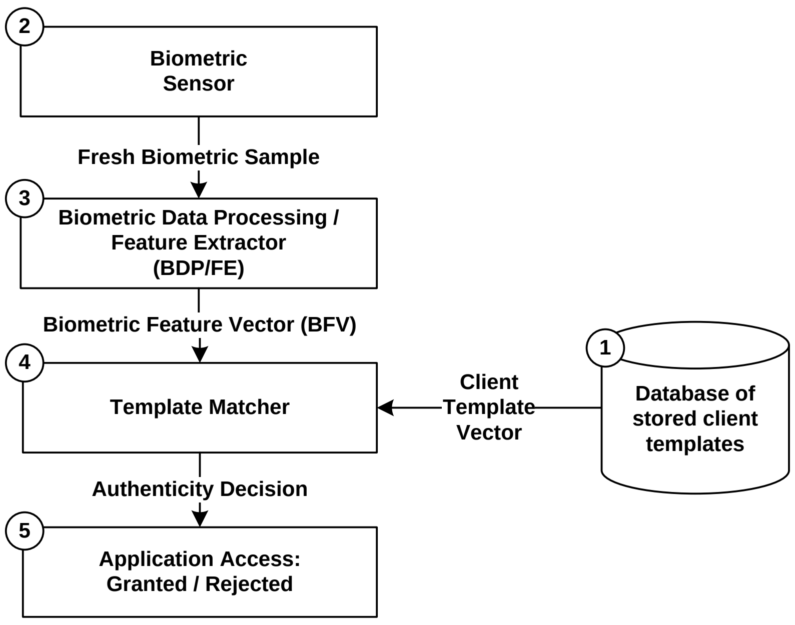

2
Biometric Sensor
Fresh Biometric Sample
3
Biometric Data Processing / Feature Extractor (BDP/FE)
Biometric Feature Vector (BFV)
4
Template Matcher
Client Template Vector
1
Database of stored client templates
Authenticity Decision
5
Application Access: Granted / Rejected

Figure 3: Overview of a biometric authentication system

1) Before a client is able to use a biometric authentication system, s/he has to register (enrol) with the authenticator. In this enrolment stage, the authenticator collects and processes the biometric data of the client and stores it as a biometric template in the authenticator's database. Biometric authentication systems can base upon different modalities, e.g. face [12], voice [13], or gait recognition [14], which differ in their accuracy, collectability or acceptability [15].
2) Once the client requires authentication, s/he presents his/her biometrics to the biometric sensor. The biometric sensor then collects the fresh biometric sample of the client.
3) The fresh biometric sample is handed over to the Biometric Data Processing / Feature Extraction (BDP/FE) component. Depending on the biometric authentication system configuration, this component may perform various (pre-) processing tasks on the collected biometric data to increase the quality of the fresh biometric sample, e.g. illumination or pose corrections for face images [16].
4) The resultant Biometric Feature Vector (BFV) of the BDP/FE component is input to the matcher component, together with the stored biometric template (generated during the enrolment in step 1). The matcher component compares the fresh BFV against the stored template to verify the client.
5) If the difference between the fresh BFV and the stored template is below a pre-defined threshold, then the client is successfully verified and gets access to the application. Otherwise, the authentication system denies application access.

Secure, reliable and correct biometric-based authentication requires a comprehensive and accurate consideration of a number of elements and tasks. For example:

1) Secure handling of the client's sensitive biometric template data must be guaranteed.
2) The authentication system must ensure that the number of clients falsely accepted or rejected is reduced to a minimum. These False Acceptance Rate (FAR) and False Rejecting Rate (FRR) can be, for example, influenced by the quality of the biometric data and the performed (pre-)processing steps, e.g. illumination and pose equalisation for face images captured on phones.

Biometric-based authentication systems can be attacked at eight points as shown in Figure 4 [17].

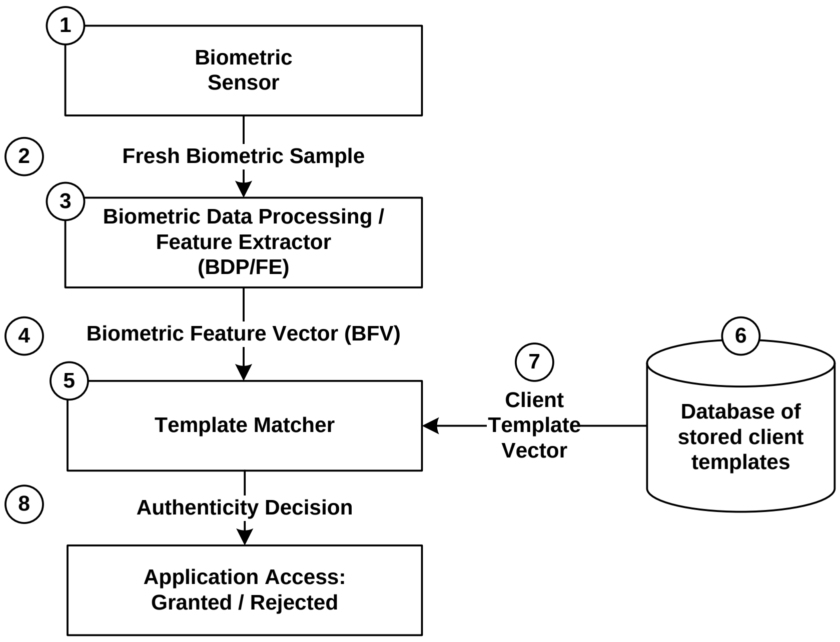

1
Biometric Sensor
2
Fresh Biometric Sample
3
Biometric Data Processing / Feature Extractor (BDP/FE)
4
Biometric Feature Vector (BFV)
5
Template Matcher
6
Database of stored client templates
7
Client Template Vector
8
Authenticity Decision
Application Access: Granted / Rejected

Figure 4: Attacks on biometric authentication system

Attack 1. Present fake biometric: In this attack, a fake biometric sample is presented to the biometric sensor. This attack does not require any malicious change to the biometric authentication system to be successful, i.e. no knowledge about the internal functioning of the system is needed. Instead, this attack is based on the presentation of false, external data to the authentication system biometric sensor. To circumvent this attack, the liveliness of the biometric sample must be checked.

Attack 2. Replay biometric sample: A previously captured, genuine biometric sample is used. To circumvent this attack, the freshness of the biometric sample must be checked.

Attack 3. Compromise BDP/FE component: The BDP/FE component is modified in such a way that it always produces correct and genuine feature values regardless of the actual component input. To circumvent this attack, the correct functioning of the BDP/FE component must be ensured or the component must reside inside a secure environment [18].

Attack 4. Replace feature values: The output of the BDP/FE component is replaced with different feature values. To circumvent this attack, the communication channel must be protected (e.g. encrypted).

Attack 5. Compromise template matcher component: The matcher component is modified in such a way that it always reports a correct match between the fresh biometric sample and the stored template regardless of the actual matching value. To circumvent this attack, the correct functioning of the template matcher component must be ensured or the component must reside inside a secure environment [18].

Attack 6. Modify template database: A new biometric template is added to the system database, or an already stored template is replaced or removed. To circumvent this attack, access to the database must be restricted and security protected.

Attack 7. Modify the template: The stored template is modified during the transfer from the system database to the matcher component. To circumvent this attack, the communication channel must be protected (e.g. encrypted).

Attack 8. Override decision: The output of the matcher component is altered. To circumvent this attack, the communication channel between the authentication system and the application using that biometric authentication system must be protected (e.g. encrypted).

As shown in Figure 4, attacks on biometric-based authentication system can be classified into two groups [19]:

1) Attacks against the system components (attacks 1, 3, 5, and 6). The risk of such attacks can be reduced by protecting the host environment executing the authentication process (cp. chapter 4).
2) Attacks against the communication channels between the system components (attacks 2, 4, 7, and 8). If these communication channels are not secured, an attacker can alter the information during transmission or replay previously gathered data. The risk of such attacks can be reduced by the integration of Location and Time (L&T) to uniquely stamp the data (cp. section 2.2.3.4).

The following are explanations to various biometric functions that are most relevant to LOTA, and described for clarity. However, it is not the intention of LOTA to develop or propose new biometric-based techniques or enhance existing ones. Instead, biometrics are merely used to support LOTA’s main work.

2.2.2.1 Discrete Wavelet Transforms for Biometric Feature Extraction

During client enrolment, biometric features of the client are extracted from the captured biometric sample and the resultant client's BFV (also called biometric template) is stored in the authentication system database indexed by the client's identity information (cp. section 2.2.2).

Discrete Wavelet Transforms (DWTs) are used in LOTA to extract the facial features of the client [20]. DWTs are a special form of wavelet transforms that provide the possibility to efficiently calculate a compact representation of the time and frequency domain of the captured client's face image. Another important property of DWT for facial feature extraction is that wavelets fade to zero outside a small interval. This provides good localisation properties in frequency and time and make DWTs a commonly used technique in various image related application areas, e.g. image compression, face localisation or face recognition [21].

The compact DWT representation is achieved by decomposing the original image into different frequency subbands which can be perfectly reconstructed. DWTs decompose the image signal by successive applying high-pass (H) and low-pass (L) filtering of the time domain signal. The output of a DWT then contains two coefficient groups (cp. Figure 5):

1) The high-pass filter coefficients, which describe the details of the image.
2) The low-pass filter coefficients, which describe the image approximation.

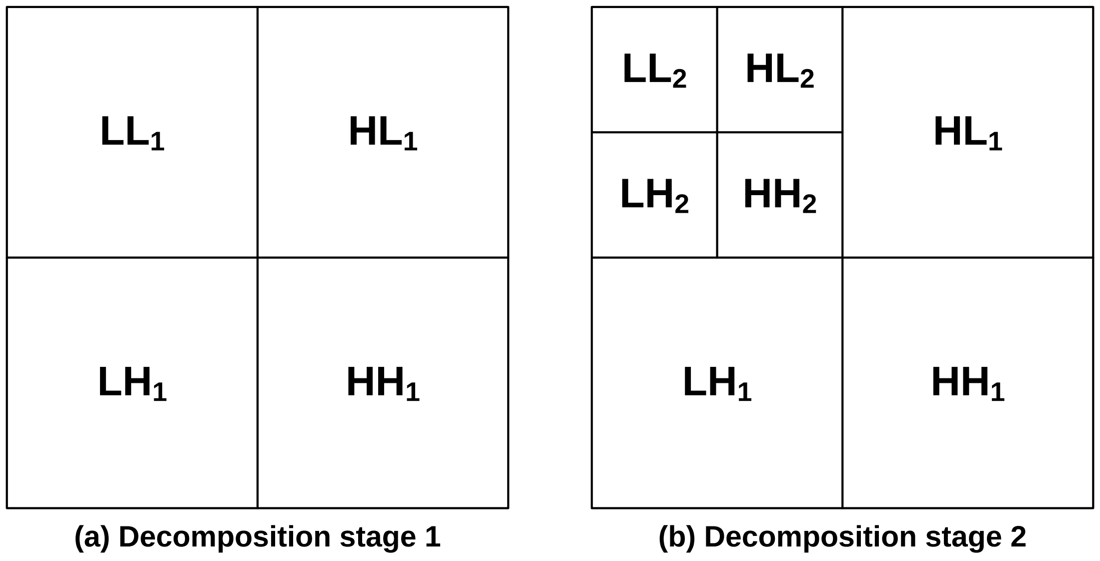


Figure 5: Pyramid decomposition

As shown in Figure 5, at a resolution level of k, the DWT pyramid scheme decomposes an image I into 3k+1 subbands (LL_k; HL_k; LH_k; HH_k; ... ; HL_1; LH_1; HH_1), with LL_k being the lowest-pass subband [22]. The subbands LH_1 and HL_1 contain finest scale wavelet coefficients that get coarser with LL_k being the coarsest. The LL_k subband is considered as the k-level approximation of I, while HL_k, LH_k, and HH_k capture vertical, horizontal and diagonal features of the image. The HL_k subband represents the horizontal high frequencies (vertical edges) such as the left / right boundary of the face (cp. Figure 6). The LH_k represents the vertical high frequencies (horizontal edges) such as eyes / mouth. Finally, the HH_k subband represents the diagonal details (high frequencies) in both directions [20].

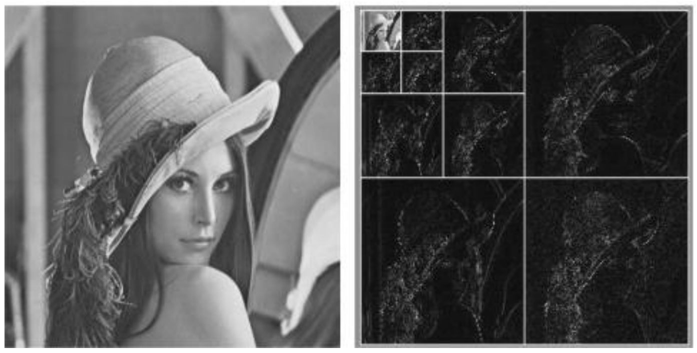


Figure 6: Example of a wavelet transformed image

Different wavelet filters can be used in the transformation stage of the image, e.g. Haar, Daubechie 4, Antonini wavelet filter. These filters differ mainly in their computational complexity and in the accuracy provided during authentication [23]. LOTA uses the Haar wavelet in all experiments, because of the high efficiency of the Haar wavelet computation and the good verification accuracy of Haar [24].

Local binarisation is then used in LOTA to binarise the extracted facial feature coefficients by dividing facial features into 3x3 blocks. Let X={x_1, x_2, ..., x_9} be a 3x3 block, then the binarised facial features B are calculated according to the following formula (1).

$$B(i) = \begin{cases} 1 \; if \; X(i) > mean(x_1,\, x_2, \dots, x_9) \\ 0 \qquad\qquad Otherwise \end{cases} \qquad (1)$$

2.2.2.2 Cancellable Biometrics

Biometric data is permanently associated with the client. In contrast to PINs or tokens, biometric features of the client cannot be replaced or revoked once the biometric data is compromised, i.e. a client cannot simply change his/her face or fingerprints. To overcome this limitation of biometrics, the concept of cancellable biometrics (cBiometrics) based on revocable biometric templates was developed [18]. The principle idea of cBiometrics is to transfer the biometric template into another secure domain by applying a one-way function. Under the condition that applying this one-way function to the enrolled template and the fresh biometric client sample preserves the original distances between these two, then matching can be performed in the secure domain without affecting the accuracy. cBiometrics offer the following advantages [25]:

1) cBiometrics preserve the client's privacy because restoring the original biometric data is computationally difficult and time-consuming.
2) cBiometrics prevent cross application matching of the biometric data because each application can use a different configuration of the one-way function.
3) The biometric template can be revoked, if this template is compromised. A new template version can be generated using a modified configuration of the one-way function.

User-Based Transformations (UBTs) are the most common tools to create revocable-BFVs. UBTs use transformation keys typically generated from PINs, which are assumed secret between client and authenticator, and are agreed at the enrolment stage (cp. section 2.2.2) together with the applied UBT. Two types of UBTs are used in LOTA:

1) User-Based Random Projection (UBRP) that uses Secret Orthonormal Matrices (SOMs) as a secure transformation for biometric templates to meet the revocability property by projecting existing data points into other spaces. SOMs ensure that the distances between all data points before and after the transformation are preserved [22]. Once a biometric template is compromised, this compromised template is revoked and a new template will be generated using a different SOM. Typically, UBRP is applied in two stages:
   a) Generate user-based orthonormal $n*n$ matrix A, where n is the size of BFV z.

b) Transform the original BVF *z* to a secure domain using matrix product *y=Az*.

2) Password-Based Shuffling (PBS) that uses a value retrieved from a token or password-hash as a shuffling key [16]. Figure 7 illustrates a generalised version of the PBS algorithm. The algorithm starts by dividing the data into k blocks, where k is the size of the shuffling key.

For i = 1,2,...,k if the bit of key(i) is equal to"1", then the corresponding BVF "data block" is moved to the beginning of the new cBiometrics BVF; otherwise, the block is moved to the end.

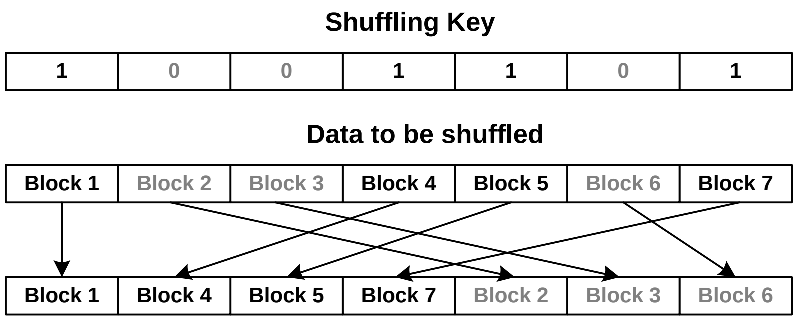


Figure 7: Password-based shuffling

2.2.2.3 Error Correcting Codes

Biometric data is "fuzzy" due to the differences between the client's captured fresh biometric data on the phone and the enrolled biometric sample stored by the authenticator, i.e. two biometric samples of the same client will never be same. In fact, the variance of BFV samples belonging to the same client can be considered as "noise". Error Correcting Codes (ECCs) [26] are used to eliminate the effect of this noise. For a specific biometric modality, the number of bits that needs to be corrected to cope with the intra-class variations is determined empirically [27]. I.e. in biometric-based authentication, an ECC encoding algorithm is carefully selected after analysing error patterns of inter-class and intra-class variations of biometric samples. In other words, the selected ECC tolerates (corrects) up to a fixed number of bits (threshold). This threshold should be carefully chosen to balance the trade of between FAR and FRR.

Handling the differences between the locations determined by the client and authenticator can also be considered as handling noisy data. Due to the different localisation techniques used on the client and authenticator side (cp. section 2.2.3.1), these locations will never be exactly the same and need therefore also be error corrected prior to verification.

To deal with these two types of fuzzy data (biometrics and location), MORE-BAILS selected the Reed-Solomon (RS) ECC [28] to handle the biometric data fuzziness (cp. section 3.2.2.2) and the Bose–Chaudhuri–Hocquenghem (BCH) ECC [29] to handle the location differences (cp. section 3.2.2.1).

Both, BCH and RS, are linear cyclic block codes, i.e. the input bit-stream to these coding schemes is divided into non-overlapping blocks, which are then independently encoded. The main difference between these ECCs is that they are able to correct a different number of errors depending on the block size and added parity-check bits. The BCH(n,k,t) code can correct up to t errors in the input bits k (i.e. information symbols) by using the encoded block length n. I.e. the BCH(n,k,t) fulfils the following properties [30]:

1) Block length: $n = 2^m – 1$.
2) Number of parity-check bits: $n - k \leq m*t$.
3) Minimum distance: $d_{min} \geq 2*t + 1$.

The RS(n,k,t) ECC is a subclass of the BCH(n,k,t) code and has the following properties [30]:

1) Block length: $n = q – 1$.
2) Number of parity-check bits: $n - k = 2*t$.
3) Minimum distance: $d_{min} = 2*t + 1$.

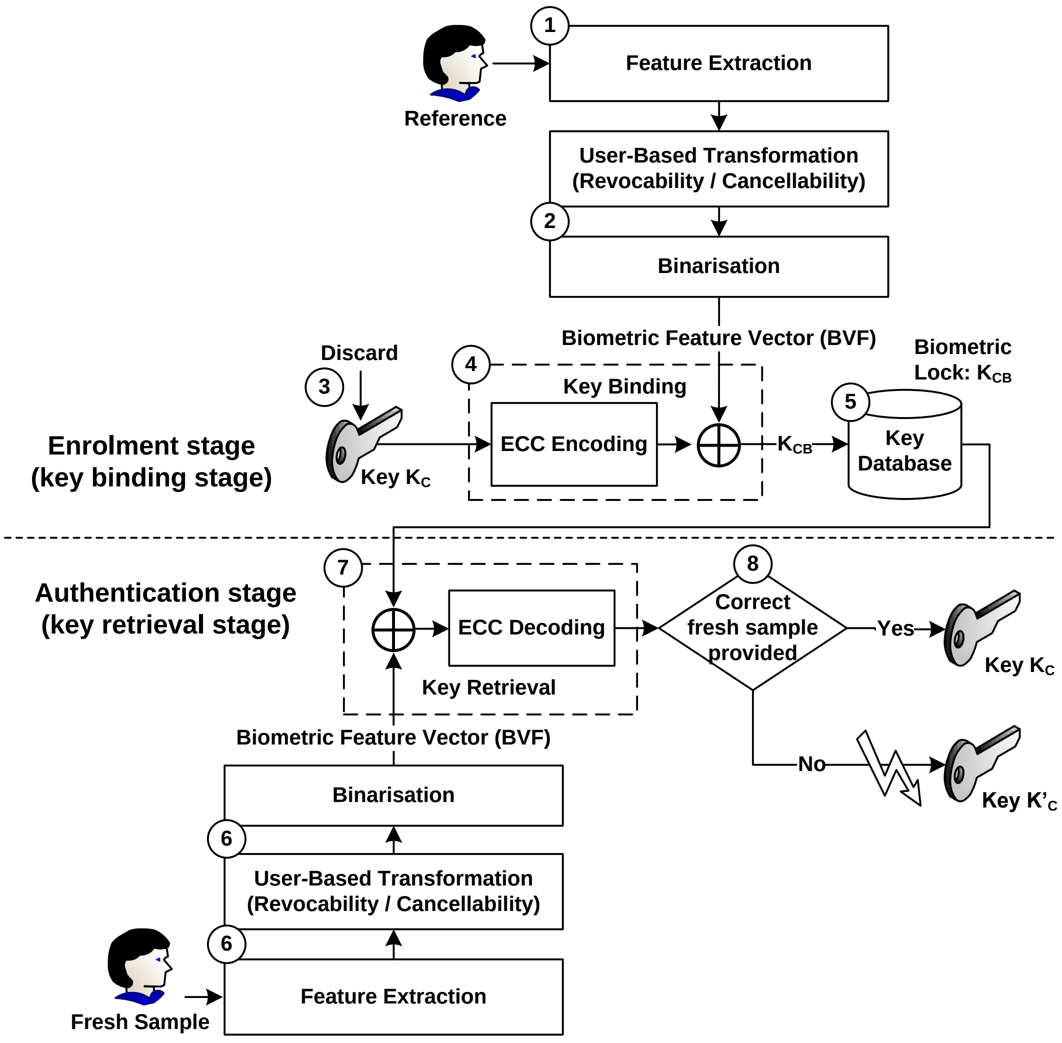
The clearly defined algebraic structure of these ECCs support the implementation of efficient (de)coding schemes [30], which makes the BCH and RS Codes widely used in biometric authentication [31].

2.2.2.4 Biometric-based Keys

To achieve stronger security mechanisms, biometrics are combined with cryptography techniques. The idea is to derive a strong client specific cryptographic key from the captured biometric sample. Biometric-based keys are used in oBiometrics (cp. section 5.1) to protect the application on the phone and to achieve a tight binding between the client and the correct interpretation of the protected application. Generation of biometric-based cryptographic keys can be categorised into three approaches [17]:

1) Key release: The cryptographic key and the biometric data of the client are stored as two separate identities at different hosts. The key is released, when an authentication attempt of the client is successful. This method is straightforward and easy to implement but has two major drawbacks [32]:
   a) Biometric templates are not secure and the matcher component can be overridden (cp. section 2.2.2).
   b) Cryptographic keys are not secure because they are not combined with biometric data.
2) Key generation: The cryptographic key is directly derived from the biometric data without storing it anywhere. Typically, biometric features are represented as a binary string and the robust bits are selected as the key. A drawback is the high FRR, which makes the key generation approach impractical [33].
3) Key binding: Biometric template and key are combined in such a way which makes it computationally infeasible to retrieve the key without previous knowledge of the client's biometric data [17].

The conceptual steps of biometric key binding are illustrated in Figure 8 [27]. oBiometrics uses the biometric key binding approach to construct client specific keys from the client's face that are then used to (de)obfuscate the oBiometrics protected application (cp. section 5.1.2). Because biometric data is fuzzy and cryptographic keys need to be 100% precise and repeatable every time the key is required, ECCs (cp. section 2.2.2.3) are used in steps 4 and 7 in Figure 8 to bridge the gap between the fuzziness of biometrics and the preciseness of cryptographic keys.



Reference
1
Feature Extraction
User-Based Transformation (Revocability / Cancellability)
2
Binarisation
Biometric Feature Vector (BVF)
Biometric Lock: K_CB
Discard
3
4
Key Binding
5
Enrolment stage (key binding stage)
Key K_C
ECC Encoding
K_CB
Key Database
7
8
Authentication stage (key retrieval stage)
ECC Decoding
Correct fresh sample provided
Yes
Key K_C
Key Retrieval
Biometric Feature Vector (BVF)
No
Key K'_C
Binarisation
6
User-Based Transformation (Revocability / Cancellability)
6
Feature Extraction
Fresh Sample

Figure 8: Biometric key binding process

1) At the enrolment stage (key binding stage), a BFV is calculated from the client's provided fresh biometrics (cp. section 2.2.2).
2) A UBT is applied on the BFV to achieve the biometric revocability (cp. section 0).
3) A cryptographic key K_C is randomly generated and passed on to the ECC encoder. After K_C is passed on to the ECC decoder, K_C is discarded and will not be stored anywhere.
4) K_C is fed into a "key combination" process that uses an ECC encoder to handle the fuzziness of the biometrics and combines (XOR) the key with the binary representation of the cancellable BVF.
5) The resultant biometric key K_{CB} is stored as a "biometric lock" in the system database.

6) At the authentication stage (key retrieval stage), the binary BFV is calculated using a fresh biometric sample in the same way as described in steps 1 and 2.
7) The binary BFV is then XORed with the stored biometric lock K_{CB}. The ECC is used in the decoding mode to tolerate the biometric fuzziness.
8) If the ECC decoding is successful (i.e. the difference between the biometric reference sample and the freshly provided biometric is within a certain pre-defined threshold), then the original and correct cryptographic key K_C will be constructed. Otherwise, if the fresh biometric sample is not within the threshold, the retrieval stage produces an incorrect key K'_C.

2.2.3 Location and Time in Remote Authentication

Classic AFs (cp. section 2.2.1) focus on verifying the client's claimed identity, i.e. all of these classic AFs are employed in the verification stage of the authentication process (cp. section 2.1). This focus on verification is acceptable in local authentication scenarios (e.g. login to a local computer), but it introduces problems for remote authentication, because classic AFs can be, for example, obtained by an attacker via man-in-the-middle, Trojans, eavesdropping, and other types of remote attacks [5]. L&T aim to overcome these drawbacks of classic AFs and can be employed during acquisition as well as verification to re-integrate the missing information about the "where and when" of the authentication attempt into remote authentication. This reduces for example the possibility of replay attacks, because the authentication data is uniquely stamped by L&T information (cp. section 3.2.3).

The following section 2.2.3.1 gives a brief explanation of the main technical aspects of localisation techniques for phones. This section is followed by a review of L&T as an AF (cp. section 2.2.3.2) and an overview of methods to generate location-based keys (cp. section 2.2.3.3), which are required to securely and tightly combine location information with other AFs [34].

2.2.3.1 Technical Background of Methods to Locate Phones

Three techniques are commonly used to establish the position of phones [35]:

1) Global Positioning System (GPS) [36]: GPS-based positioning has become the positioning technique mostly used on phones. Nearly all new developed phones feature a GPS receiver. GPS positioning is based on the reception of signals

continuously transmitted from satellites. These signals contain the precise time they were sent, as well as the location of the satellite in orbit. The GPS receiver uses the received signals of four or more satellites to calculate the current position based on trilateration. Communication between the GPS receiver onboard the phone and other phone applications requiring location is done via NMEA messages [37]. When outdoors, current GPS receivers onboard phones are able to reduce the positional error to few metres [38]. However, GPS requires to work a line of sight to the satellites. Because of that, GPS cannot be used (or the use is limited and the position becomes imprecise) indoors or in urban areas with many high glass-front buildings, where a direct line of sight to the satellites is not available.

2) WiFi Access Points (WiFi APs): WiFi-based positioning uses WiFi APs to determine the position of the phone. WiFi APs continuously transmit beacons, including an AP identifier, to their surrounding area to inform potential WiFi clients, such as a phone, about their existence. Over the last years, several databases of APs and their corresponding geographical locations were collected, for example by the Skyhook company [39]. The phone can use the received AP identifier enclosed in the beacons and these databases, via an internet link, to determine the locations of the surrounding APs, by searching the identifier in the database. Depending on the number of APs in range, the achieved location accuracy of WiFi-based positioning can vary between 10 to 100 metres. WiFi-based positioning can be used indoors as well as outdoors, as long as the AP transmitted beacon can reach the phone. However, the number of available APs differs greatly between urban and rural areas, making WiFi-based positioning a technique to be mainly used in cities with lots of existing and known APs [38].
3) Cellular network: Cellular network based positioning uses trilateration techniques to calculate the current phone location [40]. The cellular network is divided into cells, in which each cell has a unique identifier, the cell-ID. Depending on the trilateration technique used to determine the current phone location (e.g. U-TDOA [41]) and the cell size, cellular network based positioning accuracy ranges between 50 metres to a few kilometres [38]. The MNO continuously determines the phone’s location during his normal operation and also knows the cell-ID currently used in the communication process between the client’s phone and the mobile network (cp. section 1.3).

2.2.3.2 Phone Location and GPS Time as Authentication Factors

L&T were recently introduced as new AFs. Location verification is defined as the process in which a client's claimed location is verified by an authentication server [42]. Compared to classic AFs (cp. section 2.2.1), location as an AF introduces restrictions and requirements to the authentication system because:

1) Location of a phone is publicly available knowledge. Therefore, location on its own cannot be used to securely and uniquely identify individual clients during authentication. Location can be easier gathered by an attacker, then, for example, undisclosed password. An attacker could simply follow a client and use the knowledge of the client's whereabouts to get illegitimately authenticated, if location is the only AF used.
2) Integration of location as an AF requires an appropriate key-generation function to transfer the physical location into an Information Technology (IT) compatible form (i.e. a location-based key, cp. section 2.2.3.3). Utilising an inadequate transfer function can result in simple to guess location-based keys.
3) The number of locations on earth is limited. This restricts the number of possible location values to be used as an AF, i.e. the location AF key-space is restricted. In contrast, the number of available passwords can be arbitrary increased by the increase of the password length.

   Figure 9 [43] shows the maximum number of location-based keys possible on earth, if the earth's surface is divided into a grid of equal-distance squares. For a square size of one metre, $5.1*10^{14}$ different location-based keys can be generated. This number is comparable to the number of eight character long passwords (0-9a-zA-Z) that is $2.2*10^{14}$. However, depending on the technique used to determine the clients location (cp. section 2.2.3.1), the location key-space can be much less, because the location determination technique does not achieve such a high accuracy. The facts that location is public knowledge (as mentioned in 1) and only a small part of the earth is inhabited also reduce the available key-space because:

   a) An attacker can assume that a client is in a specific area where the client normally lives or works.
   b) It is unlikely that a client is for example in the Antarctic.

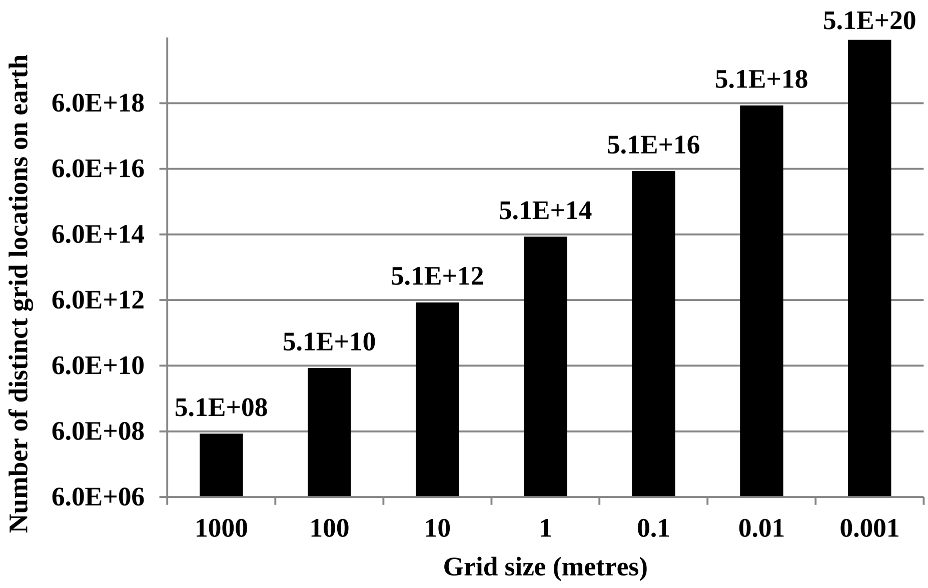


Figure 9: Maximum number of possible location-based keys

Location as an AF tries to reduce remote attacks, i.e. to stop an attacker from claiming to be at a location, where the attacker actually is not [44]. To achieve this, a unique identifier (digital signature) can be derived from GPS-based location and real-time on a Location Signature Sensor (LSS) [45]. The generated digital signature is then combined with other digital data in such a way that it stamps the data with L&T in a forgery-proof way. The security of the digital signature bases upon the fact that bit values of GPS signals change every 20 milliseconds and so the resultant digital signature changes accordingly. However, current GPS receivers available on phones cannot be used to generate such unique and trusted signatures, because these GPS receivers compute longitude and latitude from the received signals straightaway. Also, dedicated LSS are required to verify the client's location signature. This aspect prevents that an LSS-based system can be deployed on a large scale, e.g. country wide, because this requires installation of thousands of LSSs for verification. Thus, the LSS-based system is more suitable for localised areas like company premises.

A similar approach with global availability is Secure Authentication for GPS phone Applications (SAGA) [46]. In contrast to other GPS-based systems, SAGA can be used to determine the current location using the phone's onboard GPS receiver as well as used to verify the claimed location. Security analyses of SAGA concluded that SAGA offers reliable and secure location verification, with the advantage that a GPS-based system is available worldwide [47]. However, to perform location verification, the SAGA system also requires additional trusted receivers at several known locations that are used to receive reference signals from the GPS satellites. This introduces further costs for installation and maintenance of these receivers for practical authentication applications on phones.

“Location cross-checking” does not require additional receivers to be installed for location verification. Instead, location cross-checking compares the actual location of the phone with a pre-agreed set of known points of businesses related to the registered client (e.g. an ATM machine) to counter distance attacks [48]. However, location cross-checking requires an ongoing phone monitoring to track the current location of the phone (to help identify abnormal activities or attacks to the system). This ongoing phone monitoring is difficult to maintain as phones might be switched off by the client to save energy or might be used outside the traceable area. A further downside of location cross-checking is the dependence on the pre-agreed points of businesses, which are difficult to define and maintain in mCommerce applications.

“Location proofs” try to overcome the drawbacks of location cross-checking [49]. A location proof is a piece of data generated by a stationary sender (e.g. WiFi AP) that is then sent to the phone on request. The phone stores the received proof for immediate or later use and attaches it to communication messages in the case a location proof is required. An advantage of location proofs over location cross-checking is that the points of businesses must not be defined in advance. However, location proofs require trusted stationary senders instead, which should not be easily susceptible to manipulation.

Location proofs are also critical from a client’s privacy point of view [50]. Requesting a location proof discloses the client’s identity to the stationary sender, i.e. the proof issuer. This information could then, for example, be used to generate a location profile of the client. To overcome this problem, the VeriPlace architecture [50] includes two separated and trusted entities for managing location and identity information of the client. This ensures that client’s location and client’s identity are never available at the same time to one entity.

The Privacy-Preserving Location proof Updating System (APPLAUS) does not base upon stationary senders to issue location proofs [51]. Instead, other phones in the close neighbourhood serve as location proof issuers and communicate the proof to the requestor via Bluetooth in a peer-to-peer approach. The benefit of APPLAUS is that no specific network infrastructure or specialised trusted senders are required. However, security and reliability of APPLAUS bases completely upon the number and trustworthy of the neighbouring phones that issue the location proofs.

Localisation and certification services are used to tag digital content (e.g. photos or communication messages) with a location and timestamp DTL-certificate (Data-Location-Time) [52]. The DTL-certificate enables the receiver of the stamped content to verify where the content was originally created. To ensure the producer's privacy, the DTL-certificate does not include any information about the producer's identity. Thus, only T&L of the generated content can be verified by the receiver. This is in contrast to location proofs, which aim to identify the location of the client.

Challenge / response One-Time Password (OTP) techniques can be used to reduce the risk of replaying previously gathered genuine client data in a remote attack [53]. Real-time integration of OTPs into a challenge eliminates that attacker can re-use an intercepted challenge at later times (cp. 5.2 for the proposed LocAuth scheme). The challenge is only valid for a short time duration (i.e. the "when" is defined). The OTP is generated based on the International Mobile Equipment Identity (IMEI) and International Mobile Subscriber Identity (IMSI) number of the phone (cp. section 4.1.1) as well as the current time obtained from the MNO to ensure correct time synchronisation between client and authenticator [54]. The generated OTP is then used in a symmetric key algorithm to protect the authentication communication with the authenticator. However, integration of phone-based identifiers (e.g. IMEI, IMSI) should be avoided, because phone-based identifiers are not a secret and can be easier guessed than completely random generated identifiers [55].

Time and Location Based One-Time-Passwords (TLB OTP) utilises the current time and the estimated location of the client to calculate a TLB OTP [56]. This TLB OTP is then used as a key to decrypt / encrypt all further communication messages between client and authenticator. Adding location to classic OTP schemes, which are merely time-dependent, strengthens the authentication security, because it is more difficult for an attacker to determine the client's current location and precise time simultaneously [56].

To enhance and ensure correctness of the client's location and future position estimation, the client sends periodically update information about the client's current location and movement to the authenticator. Because the client's location cannot be always correctly estimated, a secondary authentication mechanism based on Short Message Service (SMS) is integrated into the TLB OTB scheme. However, this fall-back strategy requires active client involvement to deal with the SMS and hence, decreases the user-friendliness.

2.2.3.3 Location-based Keys

The geographical location of a client needs to be converted into a different representation (i.e. a location-based key) to be securely combined with other AFs. It is important that the generated key does not directly relate to the client's physical location (e.g. latitude and longitude values). I.e. a key for any location should not be predictable from a known location / key pair. If this property is not satisfied, then:

1) Large areas of the earth (e.g. Arctic, Antarctic) can be eliminated by an attacker from the available key-space area, because it is unlikely that a client is in these areas.
2) The number of keys an attacker needs to try in a brute-force attack reduces tremendously, if the client's location is approximately known.

Figure 10 shows an example of a location conversion function, which does not fulfil the two above defined properties. For this function, the relevant geographical area of the application is divided into equal-distance squares. Each square is then associated with a successive increasing number, which is used as the location-based key. If an attacker expects, for example, the client to be inside the area of Buckingham and the attacker also knows, that the key "57255" is associated with one part of this area, then the attacker can simple try the eight surrounding keys to cover the complete Buckingham area.

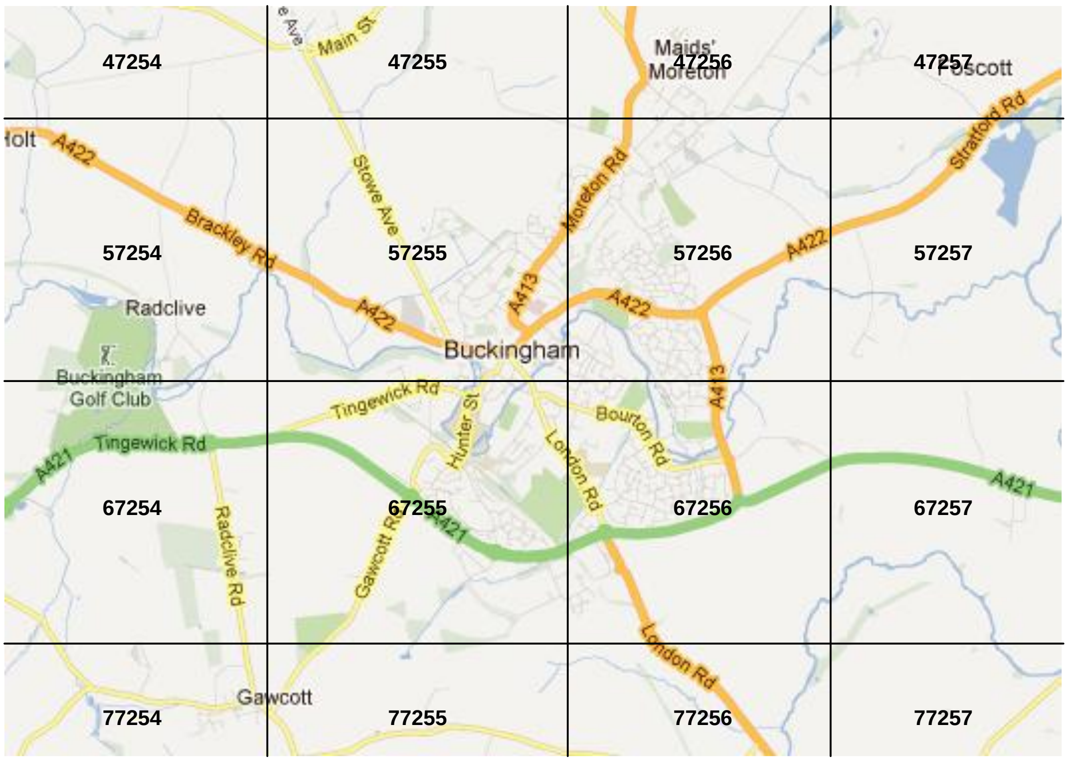


Figure 10: Insufficient function to generate location-based keys

GeoEncryption uses location as a source to generate keys for the encryption of digital messages [57]. GeoEncryption extends a classic hybrid cryptographic algorithm with a GeoLock functionality to ensure a secure location binding of the digital message. A geo-encrypted message can be opened successfully (decrypted) by a receiver, if the receiver's actual location is inside the required area. A general GeoLock mapping function, based on the estimated Position, Velocity, and Time (PVT) on the recipient is used to generate the message encryption and decryption key. However, GeoEncryption does not specify a practical and secure PVT mapping function nor does it handle support of mobile and moving recipients.

In the GeoEncryption mobility model, the encrypted message receiver continuously updates the sender about his/her current location [58]. The sender then uses this information to dynamically adjust the decryption area in which the receiver can decrypt the message.

A practical mapping function for GeoEncryption uses square areas [59]. This function was then improved to cover any shape [60]. A problem of these GeoEncryption mapping functions is that the generated encryption key merely bases upon the geographical coordinates (longitude and latitude values) of the decryption area and the used hash function as shown in Figure 11 [59]. As the geographical coordinates of the target region (decryption area) can be estimated by an attacker, if the attacker is in close proximity to the recipient, the complete security lies in the secrecy of the hash function. If the used hash function is also known to the attacker, then the attacker is able to decrypt the message.

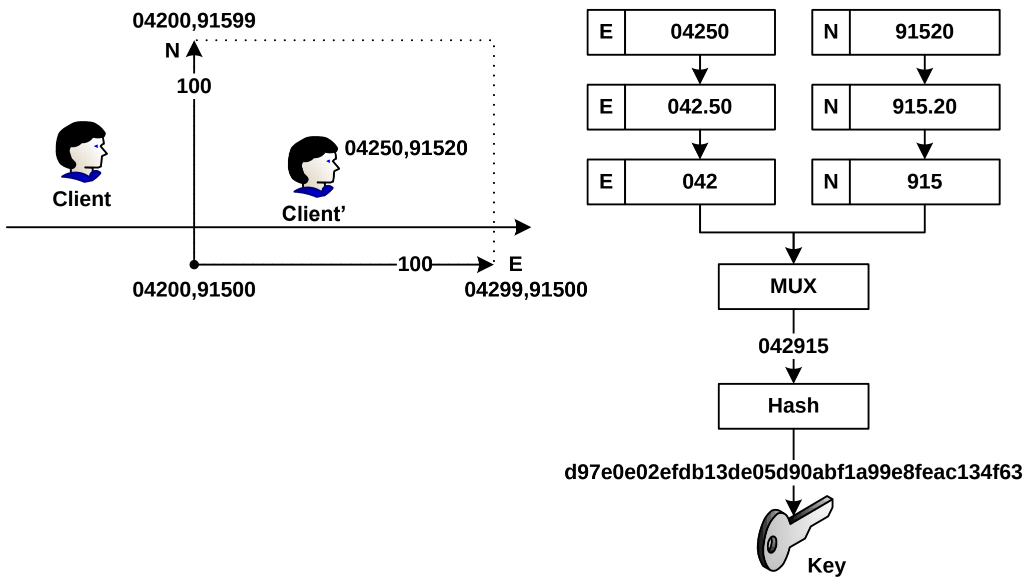


Figure 11: GeoEncryption key generation function

The practical mapping function for GeoEncryption requires that the recipient's direction of movement is known during encryption of the message to correctly define the decryption area [59]. To achieve this, the receiver transmits periodically movement updates, which are then used to calculate the correct decryption area. The importance of these updates can be seen in the example of Figure 11. The starting point of the decryption region is chosen to be at "E04200" and "N91500", and the phone is assumed to travel eastwards (E) in a maximum range of 100 metres (i.e. location will be equal to "E04299" after 100 metres). If the client is within the 100 metres (e.g. at position "E04250 / W91520" as shown in Figure 11), then the same key will be generated. However, if the client travels more than 100 metres, a different key is produced. For example, travelling 110 metres results in "E04310" and hence the "muxed" and hashed values are completely different. A similar problem occurs if the direction of the client's movement is unknown. In this case, the phone needs to move only one metre westwards instead of eastwards (i.e. to location E04199) to produce a different key. A second drawback of this key generation function is that the function does not work always correctly (i.e. produce unique keys), if the target region is near to the equator or the prime meridian. In these cases, two different locations can result in the same hash value, because the locations are equal distance away from the equator in north-south direction respectively from the prime meridian in east-west direction.

The Location-dependent Data Encryption Algorithm (LDEA) introduces a Toleration Distance (TD) during encryption of the messages to overcome the receiver's movement uncertainty [61]. The TD shall guarantee that always the same key is generated on both sides, if the receiver is within the TD area. LOTA analysed, if LDEA can be used to generate location-based keys as required in the proposed MORE-BAILS (cp. section 3.2) and LocAuth (cp. section 5.2) schemes. LOTA then concluded that LDEA is not able to generate always the correct decryption key even if the client is within the defined TD area (cp. section 5.2.1.1). Thus, LDEA cannot be used in the proposed MORE-BAILS and LocAuth schemes.

2.2.3.4 Conclusion on Location and Time in Remote Authentication

Integration of L&T as AFs into remote authentication systems enhances the security of such systems.

1) Remote attacks can be reduced, because integration of location information into the authentication data "grounds" the authentication attempt to a specific place. If the client's location claim is then independently verified, an attacker cannot pretend to be at a different place. Independent verification is necessary, because the location determined on the phone can be manipulated. For example, the GPS receiver of the phone can be manipulated or an IP-address-based location determination can be fooled by using a proxy server.
2) Replay attacks can be reduced, because the authentication data can be uniquely stamped with the current time. I.e. an attacker cannot re-use previously gathered genuine client authentication data, because of the time-stamp expiry.

LOTA proposes to utilise the already existing MNO cellular network infrastructure (cp. section 2.2.3.1) to independently verify the client's claimed location. This minimises the additional costs to set up the required infrastructure of the proposed MORE-BAILS (cp. section 3.2) and LocAuth (cp. section 5.2) schemes. This is a clear commercial advantage over systems like LSS or SAGA, which require specialised hardware to be installed for verification of the client’s claimed location. Also, using the MNO infrastructure ensures that even if the phone is switch off for a long time, instant location verification can be performed once the phone is connected to the cellular network once again. This eliminates the requirement of continuous phone monitoring or establishment of pre-agreed points of sale as required, for example, in location cross-checking. In addition, the wide coverage of the cellular network minimises the possibility that the phone is outside the area which can be monitored.

Independent location verification using the MNO infrastructure also eliminates the risk that an attacker can manipulate the location information used for verification, because the attack does not have access to the MNO infrastructure. In location verification systems, which use for example location proofs or DTL-certificates, these proofs / certificates are stored on the client’s phone. This means that the proofs / certificates are available to the client / attacker, because they are stored on the phone and thus, are subject to potential manipulation.

Location proofs and DTL-certificates also miss a tight binding between the location proof and the client. The generated location proofs can be given away and used by others, which could undermine the authentication system security, i.e. an attacker could use a stolen location proof to impersonate a genuine client. In the proposed MORE-BAILS (cp. section 3.2) and LocAuth (cp. section 5.2) schemes, the immediate and tight combination of biometrics (to identify the client) with location information minimises this problem.

Systems like APPLAUS may be adequate for low-value services that require a location proof, e.g. downloading a digital brochure of a museum for free if the client has previously visited the museum. For high-value mCommerce transaction authentication, such peer-to-peer architectures do not offer enough security and reliability, because the neighbouring phones, which issue the location proofs, cannot be fully trusted. Thus, LOTA concluded not to investigate such peer-to-peer systems any further.

Integration of client’s location into the authentication process raises legitimate privacy concerns about the correct use of the client’s location information as, for example, addressed in APPLAUS. To overcome these location privacy concerns, LOTA proposes to transfer the client’s location into a different, secure domain prior to transmission from the MNO to the authenticator (cp. section 3.2.2.4).

To securely combine location information with other AFs used in the authentication system, location-based keys need to be generated from the client’s location. LOTA uses location-based keys for two purposes:

1) In MORE-BAILS, the client location is used as an AF. Client’s location is binarised and then tightly combined by an XOR-function with the other AFs to form the final OTMFBR message (cp. section 3.2.1).
2) In LocAuth, the location-based obfuscation key is used to protect the challenge / response program (cp. section 5.2.1).

To generate secure and reliable location-based keys, LOTA has defined the following conditions for a location-based key generation technique:

1) The location-based keys should be determined completely independent from each other, i.e. without the need of any further sending of location information or movement direction to establish the same key on the client and authenticator side. This condition is required to guarantee that the authenticator can completely independently verify and consequently trust the claimed location of the client.

   GeoEncryption does not fulfil this requirement because it needs additional information to be sent between sender and receiver to work. Consequently, LOTA will not consider GeoEncryption techniques any further to achieve secure location-based key generation for the proposed schemes.
2) The same location-based key needs to be generated within a specified tolerance region. This tolerance region is necessary, because the two methods to determine and verify the client’s claimed location differ in their accuracy (cp. section 2.2.3.1). A TD should be used to handle this difference as, for example, integrated in LDEA [61].
3) All location-based keys outside the tolerance region need to be different to the key representing the tolerance region. This condition ensures that the client is

actually inside the tolerance region and not at a different place, which might produce the same key.

4) The key-space of the generated location-based keys must be large enough to minimise the risk of a brute-force attack (cp. section 2.2.3.2 and Figure 9).
5) The location-based key should incorporate location information as well as further, more secret data. This eliminates the risk that an attacker is able to calculate the location-based key correctly, if the attacker knows the client’s current whereabouts (cp. Figure 11). To overcome this problem, LOTA uses the client’s location with further dynamically changing data (e.g. cell-ID) and secrets shared between client and authenticator (e.g. Key-On-Phone (KoP)) to calculate the location-based key (cp. section 5.2.1.2).

2.3 Security Aspects of Remote Authentication

Remote client authentication based on IT-systems has security deficits compared to authentication performed by humans in direct contact, e.g. office-based authentication performed by a bank advisor in a branch. The face-to-face aspect of the authentication attempt and the personal knowledge of each other are missing. In remote authentication, the authenticator has not got automatic knowledge or guaranteed information about **who**, **how**, **where** and **when** of the client. In human / office-based authentication, these attributes are clearly and automatically defined, i.e. the contract was signed (**how**) by the known client (**who**), at that moment in time (**when**), the client was present in the authenticator’s branch (**where**).

Human / office-based authentication is mainly done through:

1) Personal knowledge of the client.
2) Combination of personal client knowledge with other forms of ID (e.g. passport, driving license), if the authenticator (e.g. bank advisor) does not know the client personally long and well enough to establish the required trustworthy.

Also, office-based authentication focuses mainly on the verification stage (cp. section 2.1), i.e. checking the claimed properties (e.g. does the signature on the provided document looks like the client’s stored example signature?). Data acquisition (e.g. where does the signed document come from?) is of minor relevance in office-based, face-to-face authentication, because the authenticity decision is based on the entity properties present to the authenticator at that moment in time.

In the absence of face-to-face contact or clearly defined client’s L&T in remote authentication, the authenticator bases his/her decision on the authentication data received from the client. Focusing on the verification stage is critical in such remote authentication because it introduces attack points on the remote authentication system as discussed in the following section 2.3.1.

2.3.1 Attack Points in Remote Authentication Systems

In remote authentication systems, the eight attack points of biometric-based authentication systems (cp. section 2.2.2) are distributed between the client’s phone and the authenticator side. Depending on the computational capabilities of the phone and the desired security properties of the authentication system, the authentication tasks (e.g. “BDP/FE” or “matching”) can be split in several ways between phone and authenticator. For example:

1) For basic phones, a possible authentication scenario is that the phone captures the fresh biometric data via the phone onboard sensors and then sends the captured data directly to the authenticator. All other computational intensive tasks are then performed by the authenticator. However, this scenario is unusual because most of today’s phones have the computational capabilities to perform at least the “BDP/FE”.
2) A more common remote authentication scenario for phones is illustrated in Figure 12. In this scenario, data acquisition through the biometric sensor and “BDP/FE” are performed on the phone. Matching against the stored template and final decision making are then executed on the authenticator side.

Figure 12 also illustrates the possible additional attack points (9 to 13) in such a remote authentication system (cp. Figure 4 for the attack points 1 to 8).

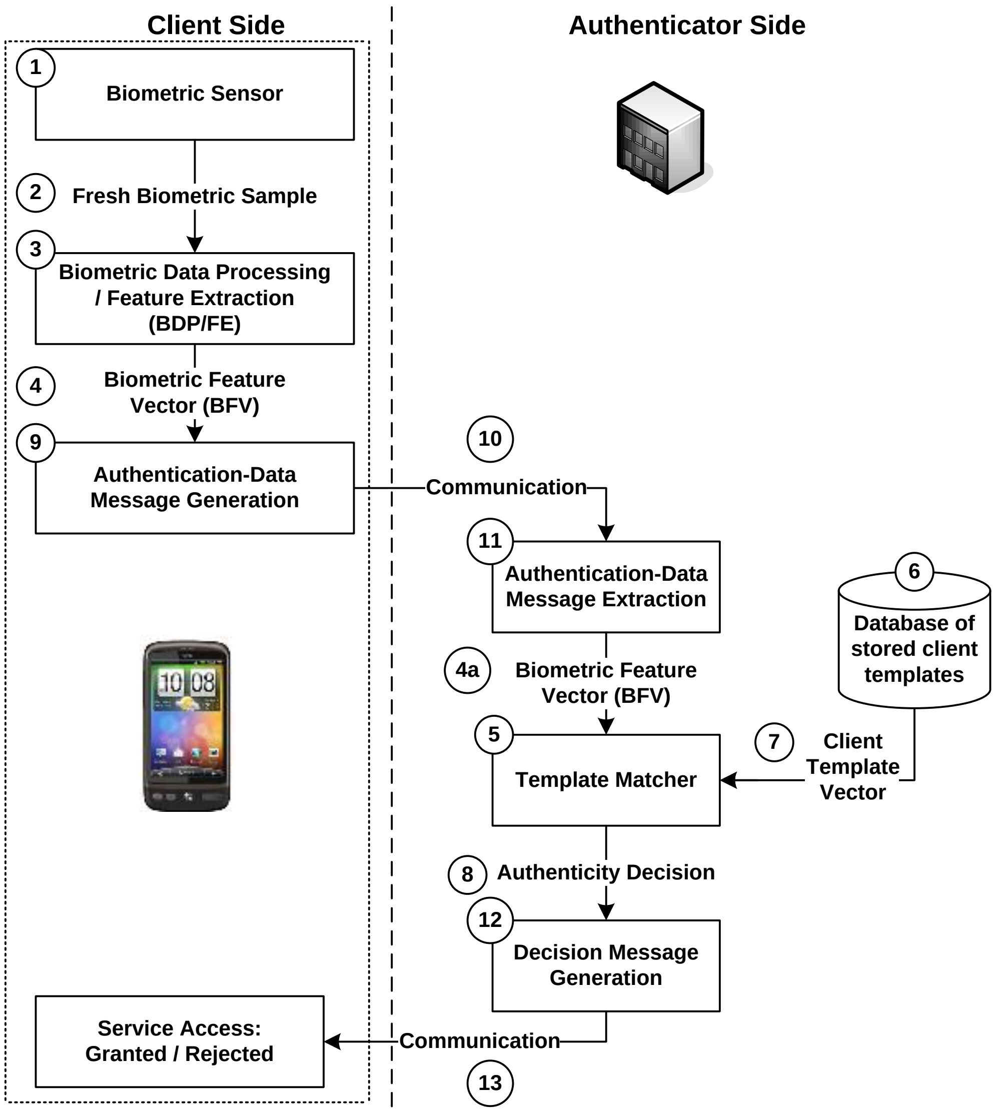

Client Side
Authenticator Side
1
Biometric Sensor
2
Fresh Biometric Sample
3
Biometric Data Processing / Feature Extraction (BDP/FE)
4
Biometric Feature Vector (BFV)
9
Authentication-Data Message Generation
10
Communication
11
Authentication-Data Message Extraction
6
Database of stored client templates
4a
Biometric Feature Vector (BFV)
5
Template Matcher
7
Client Template Vector
8
Authenticity Decision
12
Decision Message Generation
Service Access: Granted / Rejected
Communication
13

Figure 12: Attacks on remote authentication system

Attack 9. The component that generates the authentication-data message is modified in such a way that it always sends genuine authentication data to the authenticator regardless of the actual component input. To circumvent this attack, the correct functioning of the component must be ensured or the component must reside inside a secure environment.

Attack 10. The authentication-data message is changed during the wireless transmission between client and authenticator. To circumvent this attack, the communication channel must be protected (e.g. encrypted).

Attack 11. The component that extracts the authentication-data message is modified in such a way that it always extracts genuine authentication-data regardless of the actual component input. To circumvent this attack, the correct functioning of the component must be ensured or the component must reside inside a secure environment.

Attack 12. The component that generates the authentication decision message is modified in such a way that it always sends an “access granted” to the client regardless of the actual authenticity decision. To circumvent this attack, the correct functioning of the component must be ensured or the component must reside inside a secure environment.

Attack 13. The decision message is changed during the wireless transmission between authenticator and client. To circumvent this attack, the communication channel must be protected (e.g. encrypted).

Attacks on a remote authentication system can be classified into three groups as shown in Figure 12:

1) Attacks against the system components (attacks 1, 3, 5, 6, 9, 11, and 12).
2) Attacks against the communication channels between the system components (attacks 2, 4, 4a, 7, and 8).
3) Attacks against the communication channels between client and authenticator (attacks 10 and 13).

In remote authentication, these authentication system components are distributed between the client’s phone and the authenticator, which results in the following security impacts:

1) It is difficult for an attacker to perform attacks 5, 6, 7, 11, and 12, because these components reside at the authenticator’s secure host environment. I.e. the attacker does not have physical access to these components. The same is true for attacks against the component communication channels at the authenticator side (i.e. attacks 4a, 7, and 8).
2) Attacks 1, 3, and 9 can be easier performed by an attacker, because these components reside inside the client’s phone (indicated by the dotted rectangle in Figure 12). The same is true for attacks against the component communication channels at the client’s phone (i.e. attacks 2 and 4). If the client’s phone is lost or stolen, then the attacker has full control over these authentication components (i.e. over the software of the authentication system and the hardware of the

phone). Thus, the attacker’s manipulation capabilities towards the phone correspond to the white box attack model (cp. section 4.3.1).

3) Attack 10 is similar to attack 2, e.g. an attacker replays previously captured genuine client data to fool the authentication system (cp. Figure 4). However, attacking the communication channel in remote authentication is easier, because the attacker does not have to be physically at the sensor of the authentication system. In contrast, the attacker can transmit the hacked / replayed authentication-data message from anywhere in the world.
4) Attack 13 is less dangerous compared to attack 10. Manipulating the decision message does not directly introduce a security risk to the authentication system. Even if an “access denied” message is hacked by an attacker into an “access granted” message, the attacker does not have real access to the system. However, changing the decision message can be seen as a “Denial of Service” attack to genuine clients, if the attacker always changes an “access granted” to an ”access denied” message. Hence, stop a genuine client to work with the system.

To tackle the attack possibilities in remote authentication, the following three security issues are important and should be carefully addressed in the design of any remote authentication system [62]:

1) How can the client be assured that s/he communicates with the correct authenticator to eliminate the risk that an attacker pretends to be a genuine authenticator so to collect data for a subsequent replay attack?
2) How can the data transmissions between client and authenticator be secured to hinder an attacker from getting access to the genuine data?
3) How can the authenticator be assured that the received authentication data is representing the genuine client and is actually freshly collected at that moment in time?

Several methods were proposed to solve the first two aspects, e.g.:

1) Mutual authentication using the phone’s camera to assure the client and authenticator of each other [63].
2) Encrypted communication with SSL certified servers via HTTPS to protect the data transmission [64].

The protection challenges for the third aspect are an ongoing research. To establish robust techniques to re-integrate the authentication data freshness and to protect the authentication process host environment in remote authentication systems using phones, the following questions need to be addressed:

1) How can **who** (cp. section 2.2.2), **where** and **when** (cp. section 2.2.3) information, that is clearly defined in office-based authentication, be re-integrated into remote authentication to assure the authenticator about the genuine client and the freshness of the received authentication data?
2) What technologies are available to secure the authentication process host environment on the phone and how can the integrity of the authentication process execution on the phone be ensured (i.e. **how** was the authentication data collected, cp. chapter 4)?

2.3.2 Client Acceptance of Additional Security Measures

Clients that use their phones for mCommerce applications are aware of possible threats and attacks (cp. section 2.3.1) towards these mCommerce applications [65]. However, many clients are:

1) not using the basic security features already available on the phone (e.g. PIN protection, cp. section 2.2.1.1),
2) reluctant to the integration of further security measures [10].

This conflictive client attitude results from the inconvenience introduced by more sophisticated security measures [66]. From a practical point of view, the amount of reduced user-friendliness should be smaller than the costs for the client arising from an unauthorised, illegitimate application use. Otherwise, additional security measures and authentication steps are just seen as an unacceptable and unnecessary overhead by the clients.

Application specific security definitions (i.e. the number of integrated AFs, cp. section 3.2.3 for the proposed MORE-BAILS scheme) are used to overcome this client reservation against additional security measures [66]. The provided security level and thus, the usability impact on the client are adjusted to the security requirements of the application.

3 MORE-BAILS: MULTI-FACTOR ENHANCED BIOMETRIC AUTHENTICATION USING INDEPENDENT LOCATION SOURCES

The conducted literature review on remote client authentication techniques for phones (cp. chapter 2) concluded that authentication techniques lack the assurance of client's location and real-time. However, this assurance is an important aspect of secure and reliable remote client authentication, because it "uniquely grounds" the client's authentication attempt in "space and time". To overcome these limitations of client authentication schemes, the MORE-BAILS algorithm has been implemented in a scheme that securely combines:

1) classic AFs (e.g. Password / PIN and KoP token, cp. section 2.2.1),
2) freshly captured client biometric data (cp. section 2.2.2),
3) current phone location and real-time assurance (cp. section 2.2.3).

Therefore, this MORE-BAILS scheme reconstitutes the missing face-to-face features of office-based transactions into remote authentication and eliminates replay as well as distance attacks (cp. section 2.3.1). MORE-BAILS algorithm can be added to any mCommerce application that requires strong and reliable client authentication on phones. "MORE-BAILS protected application" denotes in LOTA all mCommerce applications that use the MORE-BAILS authentication algorithm.

Development of the MORE-BAILS algorithm and implemented scheme involved researchers from two focus teams in the department:

1) the Authentication Team,
2) the Wireless Localisation Team.

In particular, MORE-BAILS was developed with the assistance of Mr. Hisham Al-Assam (PhD candidate at the University of Buckingham), who provided the biometric experimental dataset and configured the BDP/FE component.

The novel MORE-BAILS concept scheme (cp. section 3.1) uses two independent localisation sources to verify the client's claimed location in combination with further AFs to reliable authenticate clients on phones [67]. This MORE-BAILS concept scheme was then extended to:

1) enhance the algorithm security by combining all employed AFs into one binary representation that does not leak any useful information to an attacker if any of the AF is compromised,
2) preserve the client's location privacy.

This security enhanced version of MORE-BAILS is described in more detail, including the technical implementation and experimental results, in section 3.2.

3.1 *Concept of MORE-BAILS*

The steps performed on the client side in this MORE-BAILS concept scheme are shown in Figure 13.

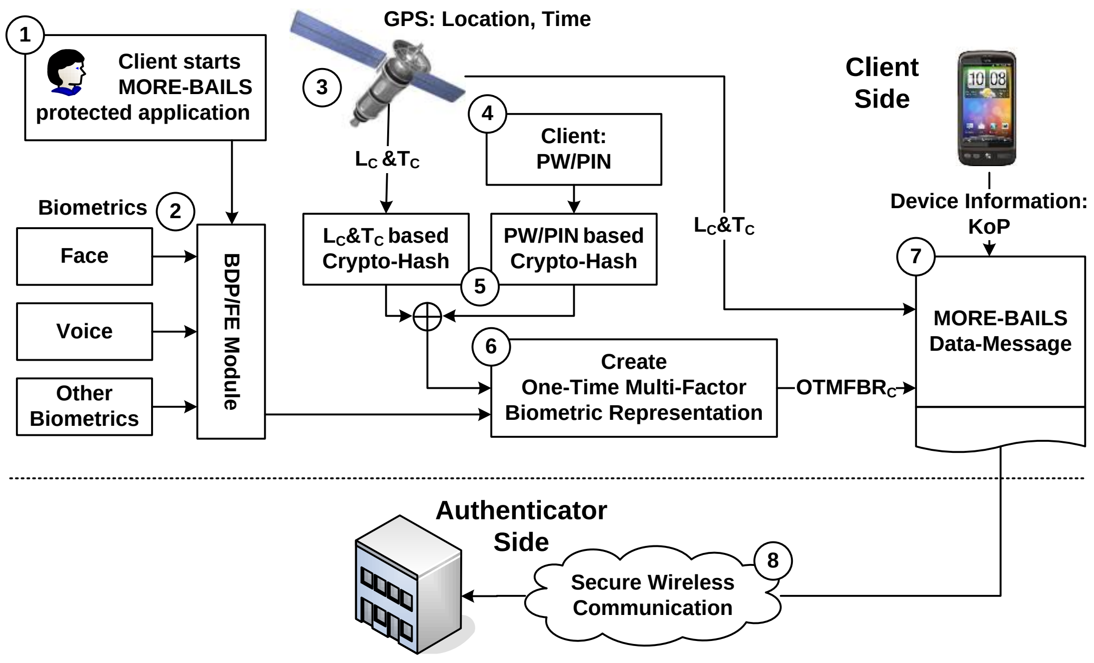


Figure 13: MORE-BAILS concept scheme, client side

1) The MORE-BAILS scheme starts upon a client opens a MORE-BAILS protected application on his/her phone.
2) This application opening triggers a process that captures fresh client biometrics using the phone onboard sensors and computes the BFV of the client (cp. section 2.2.2.1).

3) The current L_C&T_C (subscript letter "C" denotes in LOTA data collected or generated on the client side, while letter "A" denotes the authenticator side) are determined using the phone onboard GPS receiver. L_C&T_C will be thereby extracted from the Global Positioning System Fix Data message generated every second by the GPS receiver from the received satellite signals (cp. section 2.2.3.1).
4) The client enters his/her personal credentials (e.g. password / PIN).
5) The client entered credentials and the determined L_C&T_C are then crypto-hashed and afterwards XORed.
6) The result of XORing credentials and L_C&T_C is then XORed with the calculated BFV of the client to form the One-Time Multi-Factor Biometric Representation ($OTFMBR_C$) of the client.
7) To the resultant $OTMFBR_C$, further information about the phone (KoP) and a copy of the fresh L_C&T_C information is attached to form the MORE-BAILS data-message.
8) This MORE-BAILS data-message is then sent to the authenticator using a secure wireless communication channel (e.g. SSL protocol over 3G cellular link) for verification.

The steps performed by the authenticator to verify the received MORE-BAILS data-message are shown in Figure 14.

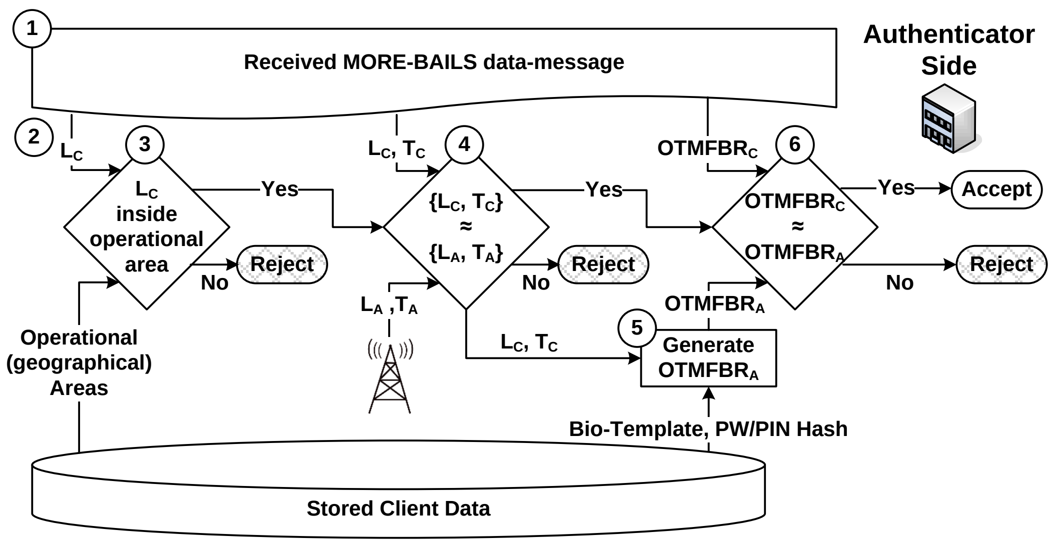


Figure 14: MORE-BAILS concept scheme, authenticator side

1) The authenticator receives the MORE-BAILS data-message sent by the client.
2) The authenticator then extracts L_C&T_C information and $OTMFBR_C$ from the received MORE-BAILS data-message for verification.
3) The authenticator ensures that the location of the client is valid by comparing the stored pre-agreed geographical areas with the received client's location L_C. If L_C is not within these pre-agreed areas, then the authentication attempt is rejected. This can be considered as a distance attack.
4) The current location of the client's phone is requested from the MNO to verify independently that the phone is actually at the claimed location, i.e. within a pre-defined threshold (cp. section 2.2.3). If the claimed location cannot be verified, then the authentication attempt is rejected. This can be considered as an impersonation attack in which the attacker does not know the exact genuine client's current location.
5) The authenticator uses the stored client data (enrolled biometric template, password / PIN hash value) and the crypto-hashed L_C&T_C information extracted from the received MORE-BAILS data-message to calculate an $OTMFBR_A$.
6) This calculated $OTMFBR_A$ is then compared with the received $OTMFBR_C$ to verify the client's claimed identity. If this final step is successful, then the authentication attempt will be accepted. Otherwise, the rejected authentication attempt can be considered as an impersonation attack in which the attacker does not know the biometrics and / or the password / PIN of the client.

The XOR-operation is used in MORE-BAILS as shown in Figure 13 and Figure 14 to combine AFs at several stages, because:

1) It is impossible to reverse the XOR-operation without the knowledge of at least one of the initial XOR argument values. This means that an attacker cannot gain any useful information from an eventually intercepted $OTMFBR_C$ message.
2) The XOR-operation is always entirely reversible (which is in contrast for example to the AND or OR-operation), if the result and one of the initial arguments are available. This XOR-operation property enables the authenticator to sequentially verify the combined AFs.

In this MORE-BAILS concept scheme, client’s L_C&T_C information needs to be added to the data-message to allow calculation of the $OTMFBR_A$, because:

1) It is impossible to extract L_C&T_C information from the $OTMFBR_C$ because of the applied crypto-hash function.
2) It is impossible for the authenticator to determine the exact client’s location due to the different accuracies of the used localisation techniques on the client and authenticator side (cp. section 2.2.3.1).

However, attaching L_C&T_C information directly to the MORE-BAILS data-message introduces security and privacy drawbacks. For example, in the event of an attacker intercepting the client's MORE-BAILS data-message, the client's privacy is breached as a result of disclosing L_C&T_C information. Furthermore, the security of this MORE-BAILS concept scheme is vulnerable, because the L_C&T_C based crypto-hash key can be obtained from L_C&T_C information as shown in Figure 13. Therefore, capturing $OTMFBR_C$ and the L_C&T_C based crypto-hash key might leak information about the biometrics, because the security then relies solely on the complexity of the password / PIN-based crypto-hash function. This means that the MORE-BAILS concept scheme becomes vulnerable to replay attacks if the attacker knows the client’s current whereabouts.

After these security and privacy drawbacks were identified by LOTA, a security enhanced version was developed and is detailed in the following section 3.2.

3.2 Security Enhanced MORE-BAILS

The security enhanced MORE-BAILS scheme eliminates the drawbacks of the MORE-BAILS concept scheme.

1) It is no longer necessary to attach L_C&T_C information to the MORE-BAILS data-message. Instead, L_C&T_C will become integral parts of the OTMFBR. This means that the authenticator does not need to calculate an $OTMFBR_A$ for comparison with the received $OTMFBR_C$ as in the MORE-BAILS concept scheme. This enhanced scheme sequentially verifies all AFs of the received $OTMFBR_C$ and rejects the authentication attempt if any AF cannot be successfully verified.

2) To deal with the difference between the position accuracy of the two independent localisation techniques used, this scheme implementation uses an ECC technique (named ECC_{Loc}) to reconcile this location difference within a certain threshold area (cp. section 2.2.3.1). A second ECC (named ECC_{Bio}) is used to tolerate intra-class variations of biometric samples (cp. section 2.2.2.3).

3.2.1 *Scheme Overview*

The MORE-BAILS steps executed on the client's phone are shown in Figure 15.

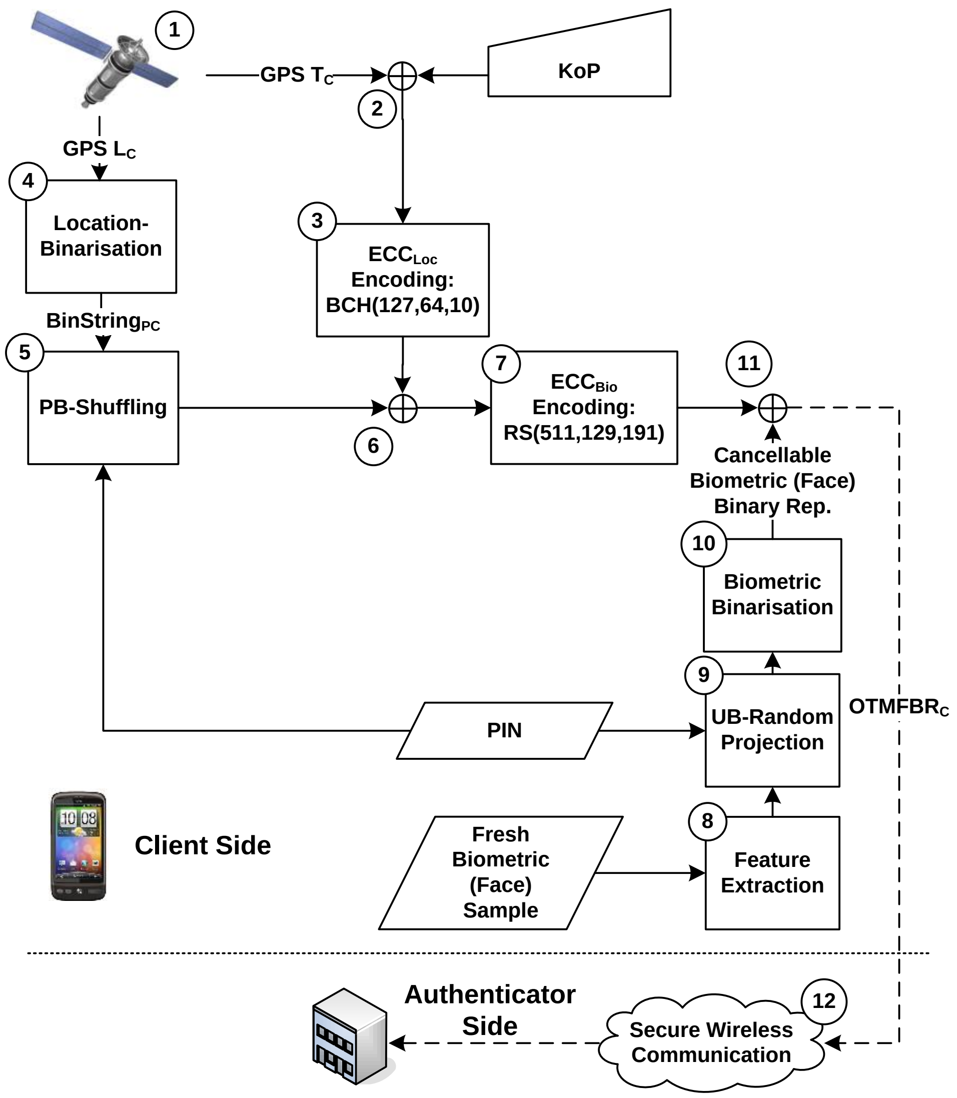
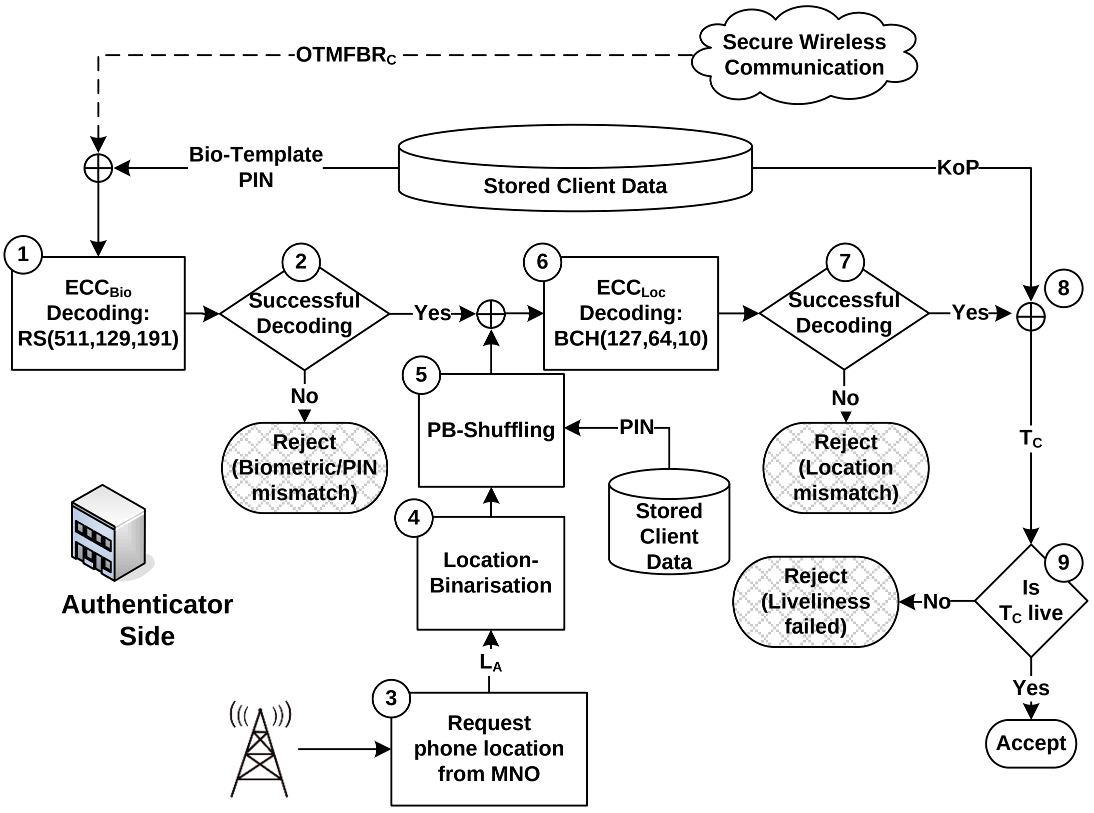


Figure 15: Security enhanced MORE-BAILS scheme, client side

1) The MORE-BAILS scheme starts upon a client opens a MORE-BAILS protected application on his/her phone. MORE-BAILS determines the current L_C&T_C using the phone onboard GPS receiver.

2) The timestamp T_C is XORed with the KoP.
3) The resultant output of XORing T_C and KoP is then encoded by ECC_{Loc} (cp. section 2.2.2.3).
4) A binary representation of the location L_C is produced (cp. section 3.2.2.3).
5) This L_C is fed into a PBS with a PIN-based shuffling key (cp. section 0).
6) The ECC_{Loc} output (step 3) is then XORed with the shuffled binary representation of L_C.
7) The result gets encoded by ECC_{Bio} (cp. section 2.2.2.3).
8) Simultaneously to step 2 and 4, the client's BFV is extracted from the client's biometrics freshly captured using the phone onboard sensors (cp. section 2.2.2.1).
9) A UBRP is applied on the extracted BFV (cp. section 0).
10) The projected BFV is binarised (cp. section 2.2.2.1) to produce a cBiometric version of the client's biometrics (cp. section 0).
11) The ECC_{Bio} output is XORed with the cBiometrics to produce the $OTMFBR_C$.
12) The $OTMFBR_C$ is sent to the authenticator via a wireless communication link.

The MORE-BAILS steps executed by the authenticator are shown in Figure 16.



OTMFBRC
Secure Wireless Communication
Bio-Template PIN
Stored Client Data
KoP
1
ECCBio Decoding: RS(511,129,191)
2
Successful Decoding
Yes
No
Reject (Biometric/PIN mismatch)
6
ECCLoc Decoding: BCH(127,64,10)
7
Successful Decoding
Yes
No
Reject (Location mismatch)
8
TC
5
PB-Shuffling
PIN
Stored Client Data
4
Location-Binarisation
LA
3
Request phone location from MNO
9
Is TC live
No
Reject (Liveliness failed)
Yes
Accept
Authenticator Side

Figure 16: Security enhanced MORE-BAILS scheme, authenticator side

1) The received $OTMFBR_C$ is XORed with the client's stored cBiometrics (cp. section 0) and decoded by ECC_{Bio} (cp. section 2.2.2.3).
2) Successful ECC_{Bio} decoding means authenticity of the client's provided biometrics and PIN due to the applied UBRP as shown in step 9 in Figure 15.
3) The client's location L_A is obtained via MNO (cp. section 2.2.3.1).
4) A binary representation of L_A is generated (cp. section 3.2.2.3).
5) The binary representation of L_A is shuffled (PBS) based on the stored PIN (cp. section 0).
6) The output of PBS is XORed with the ECC_{Bio} decoding output (step 1) and fed into the ECC_{Loc} decoding process (cp. section 2.2.2.3).
7) Successful ECC_{Loc} decoding means that client's location has been verified together with the correct shuffling key.
8) ECC_{Loc} output is XORed with the stored KoP. If the client has used the correct KoP, the timestamp T_C used at the client side (step 2 in Figure 15) is retrieved.
9) T_C is compared with the current time. If the time difference is within a pre-defined threshold, then the retrieved T_C passes the liveliness test and the client's authentication attempt will be accepted.

3.2.2 Experimental Data and Implementation Remarks

The following subsections explain the chosen implementation and simulation configuration details for the location (cp. section 3.2.2.1) and the biometric AF (cp. section 3.2.2.2) that were used to test and verify the feasibility and security of the MORE-BAILS scheme. Also, proposed algorithms to binarise the locations determined by client and MNO (cp. section 3.2.2.3) as well as algorithms to preserve the client's location privacy (cp. section 3.2.2.4) are detailed. Experimental results, together with a discussion of these results, are then presented in section 3.2.3.

3.2.2.1 Location Dataset

MORE-BAILS tolerates up to 320m difference between the locations provided by the used GPS and MNO localisation techniques (cp. section 2.2.3.1). The 320 metres are chosen to minimise false rejection of genuine clients and are supported by the FCC E-911 directive [68].

For the experiments, a location dataset containing 20 measurements in the city of London, UK was collected as shown in Table 1. This dataset contains in the "L_C"-column in Table 1 the latitude and longitude values of the client's location determined by the GPS-receiver onboard an Android HTC-Wildfire Smartphone. The corresponding MNO-based location of the client's phone is shown in column "L_A" (cp. section 2.2.3.1). The mobile phone tracking service "FollowUs" [69] is used in the trials to "simulate" the tasks of the MNO. Direct involvement of a major MNO (e.g. British Telecom or Vodafone) requires extensive business negotiations and legal work, which is outside the scope of LOTA. Instead, the FollowUs service offers a simple to use, immediately available, and accurate way to determine the phone location independently via the cellular network. During the trials carried out on Android Smartphones (cp. section 4.3.3), FollowUs provided reasonable locations. Furthermore, the accuracy of the provided location measurements is consistent to MNO-based measurements [70].

The "L_T"-column represents the geographical location of the serving MNO cell. This L_T location is required to calculate the binary representation of the two locations (L_C and L_A, cp. section 3.2.2.3) for the ECC_{Loc} comparison. The last two columns show the difference between the GPS and MNO determined location in metre and the resultant difference in bits after the location binarisation method has been applied.

Table 1: Location measurements

	GPS-based Location (L_C)		**MNO-based Location (L_A)**		**MNO Cell Location (L_T)**		**Distance L_C, L_A**	
	Lat L_C (deg)	**Lon L_C (deg)**	**Lat L_A (deg)**	**Lon L_A (deg)**	**Lat C (deg)**	**Lon C (deg)**	**Decimal (m)**	**nBits**
1	51.5042	0.0008	51.5080	-0.0020	51.4978	0.0048	463.8063	14
2	51.5020	-0.0013	51.4980	-0.0240	51.5016	-0.0193	1633.1882	48
3	51.4977	0.0076	51.4970	0.0070	51.4979	0.0051	91.6758	2
4	51.4989	-0.0513	51.4990	-0.0500	51.5027	-0.0563	90.1893	2
5	51.5143	-0.1486	51.5140	-0.1470	51.5147	-0.1463	117.2180	4
6	51.5141	-0.1442	51.5120	-0.1470	51.5147	-0.1463	297.6467	9
7	51.5132	-0.1420	51.5120	-0.1390	51.5125	-0.1395	244.1811	8
8	51.5105	-0.1427	51.5100	-0.1420	51.5087	-0.1411	76.1176	3
9	51.5139	-0.1470	51.5120	-0.1470	51.5147	-0.1463	206.2836	5
10	51.5138	-0.1401	51.5120	-0.1390	51.5125	-0.1395	210.7231	6
11	51.5137	-0.1381	51.5120	-0.1390	51.5125	-0.1395	196.4461	5
12	51.5128	-0.1469	51.5140	-0.1471	51.5142	-0.1489	136.3620	4
13	51.5139	-0.1439	51.5130	-0.1400	51.5142	-0.1489	289.4840	9
14	51.5138	-0.1428	51.5130	-0.1420	51.5147	-0.1463	107.4662	3
15	51.5084	-0.1509	51.5090	-0.1510	51.5126	-0.1470	71.9936	3
16	51.5163	-0.1375	51.5180	-0.1370	51.5125	-0.1395	188.9682	6
17	51.5076	-0.1510	51.5090	-0.1510	51.5126	-0.1470	154.3240	4
18	51.5160	-0.1370	51.5180	-0.1370	51.5125	-0.1395	224.2693	6
19	51.5049	-0.1484	51.5120	-0.1470	51.5126	-0.1470	143.6615	4
20	51.5161	-0.1382	51.5160	-0.1310	51.5125	-0.1395	238.3377	7

Measurements in row 1 and 2 in Table 1 are chosen to be outside the accepted maximum distance between the GPS and MNO localisation technique of 320m. These two measurements result in 14 (respectively 48) bits difference in the binary representation, which will not be accepted in the location verification step 7 of Figure 16. All other measurements (3 - 20) are well within the valid distance and should be therefore accepted by MORE-BAILS.

The acquired GPS-based location L_C on the client side is binarised to produce a 127-bit location representation (step 3 in Figure 15). The 320m error tolerated is equivalent to 10-bit out of 127-bit according to the chosen setting. Therefore, BCH(127,64,10) ECC (ECC_{Loc}) was selected to deal with the location accuracy differences. This ECC_{Loc} takes an 64-bit input to produce a codeword of size 127, and is capable of correcting up to 10-bit errors. The 64-bit input comes from XORing the 64-bit KoP with 64-bit representation of time as shown step 2 in Figure 15.

3.2.2.2 Biometric Dataset

The face is used as the biometric modality (cp. section 2.2.2) in the MORE-BAILS experiments and trials on the phone. The face was chosen because:

1) Capturing face images is easy due to the wide availability of cameras on phones and because it does not require expansive and / or specialised hardware.
2) Capturing face images is an unobtrusive task that most clients do not have any concerns with. In contrast, for example, retina / iris recognition is seen with much more reservations by many clients [15].
3) Combination of face biometrics with DWT (cp. section 2.2.2.1) to extract the face BVF is an efficient and accurate client verification method on phones [23].

In the experiments, face images from the Extended Yale-B database are used [71]. This database has 28 subjects under 9 different poses. Because it can be expected that the client is always in frontal pose towards the camera during an authentication attempt, only this frontal pose is used. Each of the subjects has 64 images captured under different illumination conditions. Hence, the total number of frontal face images in the database is 1792. These images are divided into five subsets according to the direction of the light-source from the camera axis as shown in Figure 17.

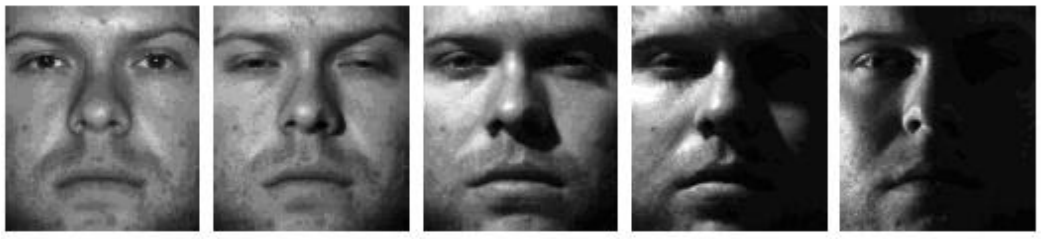


Figure 17: Image samples in different illumination subsets

In all experiments, the first three images per subject from subset 1 are selected to represent client images captured during enrolment, i.e. these images are stored in the authenticator's database (cp. section 2.2.2). All remaining images are used to represent freshly taken client images during an authentication attempt.

Implementation remarks about the used face biometric modality:

1) The Haar DWT filter (cp. section 2.2.2.1) was selected to extract the facial features (step 8 in Figure 15) from these subject images. The used size of the facial BFV extracted by DWT is 504-features.
2) A cancellable version of the client's BFV is created using UBRP (steps 9 and 10 in Figure 15). The biometric binarisation produces a 504-bit cancellable face representation. After analysing the error patterns of inter- and intra-class variations of face images, LOTA concluded that 38% of the binary face BFVs needs to be corrected, or 191 out of 504-bits. In other words, if the Hamming distance [26] between two binary face BFVs is less than 191, then the two binary face BFVs are accepted as belonging to the same client.
3) To deal with intra-class variations of face samples, the RS(511,129,191) ECC is used (step 7 in Figure 15). This ECC_{Bio} takes 129 symbols as input to produce a codeword of 511 symbols, and therefore corrects up to 191 errors.
4) The final $OTMFBR_C$ is obtained by XORing the codeword output of RS encoding with the 511-bit cancellable face binary representation.
5) At the authenticator side, the corresponding RS(511,129,191) ECC_{Bio} (step 1 in Figure 16) respectively BCH(127,64,10) ECC_{Loc} decoder (step 6 in Figure 16) are used to deal with BFV and location variations.

3.2.2.3 Location Binarisation

MORE-BAILS adopts binarised location information on the client and authenticator side to:

1) Combine the location with the other AFs (e.g. biometrics), which exist in binary format.
2) Calculate the actual distance between the two, independently determined locations based on the number of different bits in the generated binary location strings.

This binary XOR combination (chosen to integrate two streams of data in several stages as shown in steps 2, 6, and 11 in Figure 15) is useful, because it minimises the risk that an intercepted OTMFBR leaks any useful information about any of the employed AFs. To generate the binary representation of the location, the developed MORE-BAILS algorithm uses only longitude and latitude information, represented as double values on Android-based phones (cp. section 4.3.3). Altitude is currently not used, because the MNO does not provide any information about the altitude of the tracked phone. However, once the horizontal location comparison has been successful, the altitude supplied by the GPS can be further compared with the client's current location altitude, available from Google earth for example. This feature has not been considered in the current algorithm.

The location binarisation algorithm binarises the geographical location (latitude / longitude) in such a way that all bits in the resultant binary location string represent the same distance, i.e. all bits have the same significance. This is important, because the ECC_{Loc} algorithm (cp. step 3 in Figure 15) that deals with the fuzziness of the two locations treats all bits in the binary location string equally. I.e. the significance of all bits must be the same and every bit difference should reflect the same distance between the two locations.

In the location binarisation algorithm, the location of the serving MNO cell is used as a reference point, independently available to client and authenticator. The serving MNO cell (cell-ID) is known on the client's phone through the communication process. The authenticator will request this cell-ID, in addition to the phone location, from the MNO. The corresponding geographical location is accessible by client and authenticator through the MNO by agreement, or by various (publicly available) online databases, e.g. Google Mobile Maps [72]. This location reference point is required to generate a short binary location string, because the total number of bits to represent the location is limited due to the later following XORing with the binarised BFV as shown in step 11 in Figure 15. I.e. the BFV length defines the maximum length of the binary location string.

On the client and authenticator side, all distances between the client's phone location, and the location of the serving MNO cell are calculated using the "Spherical Law of Cosines" (SLC) [73] shown in formula (2).

$$d = \arccos(\sin(lat_1) * sin(lat_2) + cos(lat_1) * cos(lat_2) * cos(long_2 - long_1)) * R \tag{2}$$

Where:	lat_1, $long_1$	: Latitude and longitude of location one
	lat_2, $long_2$	: Latitude and longitude of location two
	R= 6371	: Radius of earth in kilometre

The SLC formula is generally used to calculate distances between two points on a sphere [74], e.g. two locations described by their longitude and latitude values on earth. The SLC formula was chosen in the LOTA experiments over other possible formulas (e.g. Vincenty's formula) to calculate the required distances because:

1) The SLC formula can be calculated more efficiently on phones than other formulas because of its simplicity,
2) Today's phone processors offer high enough precision on floating point number operations to calculate the SLC formula,
3) The introduced distance error of the SLC formula is well within the available precision of the localisation techniques (cp. section 3.2.2.4), i.e. a more precise formula which is, as a result, more complex and time consuming to calculate on the phone would not offer any advantage over the chosen SLC formula.

Figure 18 details the steps of the location binarisation algorithm (step 4 in Figure 15) performed on the client side. Similar steps are performed by the authenticator to binarise the received MNO-based location (step 4 in Figure 16) during verification.

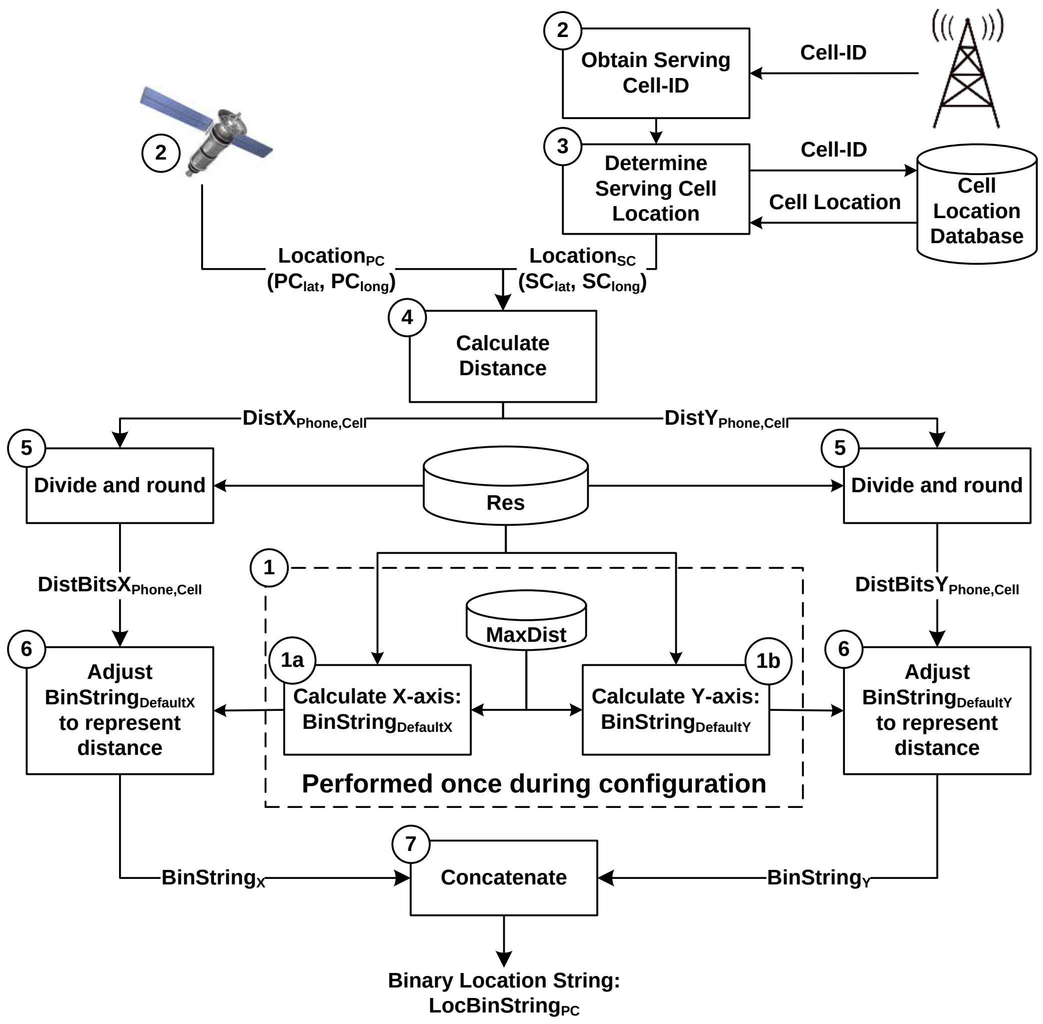

2
Obtain Serving Cell-ID
Cell-ID
3
Determine Serving Cell Location
Cell-ID
Cell Location
Cell Location Database
2
Location$_{PC}$ (PC_{lat}, PC_{long})
Location$_{SC}$ (SC_{lat}, SC_{long})
4
Calculate Distance
DistX$_{Phone,Cell}$
DistY$_{Phone,Cell}$
5
Divide and round
Res
5
Divide and round
DistBitsX$_{Phone,Cell}$
DistBitsY$_{Phone,Cell}$
1
MaxDist
1a
Calculate X-axis: BinString$_{DefaultX}$
1b
Calculate Y-axis: BinString$_{DefaultY}$
Performed once during configuration
6
Adjust BinString$_{DefaultX}$ to represent distance
6
Adjust BinString$_{DefaultY}$ to represent distance
7
Concatenate
BinString$_X$
BinString$_Y$
Binary Location String: LocBinString$_{PC}$

Figure 18: Location binarisation algorithm

To clarify the detailed description of the algorithm steps, the following example values shown in Table 2 will be used for a client located (latitude, longitude) at the University of Buckingham, UK.

Table 2: Example values for location binarisation algorithm

Name	Value	Description
PC_{lat}	51.9940	Latitude of phone, determined by client
PC_{long}	-0.9816	Longitude of phone, determined by client
PA_{lat}	51.9986	Latitude of phone, determined by MNO
PA_{long}	-0.9776	Longitude of phone, determined by MNO
SC_{lat}	51.9980	Latitude of serving cell
SC_{long}	-0.9697	Longitude of serving cell
Res	100	Resolution, distance represented by one bit
MaxDist	1000	Maximum distance between phone and serving cell
$Bits_{MaxDist}$	10	Bits required to express MaxDist, i.e. $Bits_{MaxDist}$ = MaxDist / res

The location binarisation algorithm performs the following steps to convert a geographical location (latitude, longitude values) into a binarised location:

1) During an initial configuration step (indicated by the dotted rectangle in Figure 18), a default binary location string ($BinString_{Default\{X,Y\}}$) is created. This $BinString_{Default\{X,Y\}}$ consists of $Bits_{MaxDist}$ "1s" and $Bits_{MaxDist}$ "0s" to represent the distance between phone and serving MNO cell on the X-axis (1a) and the same number of "1s" and "0s" to represent the distance on the Y-axis (1b) respectively. The $BinString_{Default\{X,Y\}}$ will be adjusted in step 6 of this algorithm (i.e. the number of "0s" and "1s" will be changed) in accordance to the actual distance between phone and serving MNO cell.

   Example:

   The total number of bits required for $BinString_{Default\{X,Y\}}$ is:

   $BinString_{DefaultX} = V * Bits_{MaxDist} = 2 * 10 = 20$ bits

   $BinString_{DefaultY} = H * Bits_{MaxDist} = 2 * 10 = 20$ bits

   Where

   V expresses if the phone is to the east or west of the serving MNO cell

   H expresses if the phone is to the north or south of the serving MNO cell

   By changing the "Res" and "MaxDist" values, the length of the resultant $BinString_{Default\{X,Y\}}$ can be adjusted to the requirements of later MORE-BAILS functions, i.e. the $BinString_{Default\{X,Y\}}$ length can be tailored to the length of the BFV (cp. step 3 in Figure 15).

   Example:

   $BinString_{DefaultX}$ = 11111111110000000000

   $BinString_{DefaultY}$ = 11111111110000000000

2) The binarisation algorithm obtains the phone location latitude and longitude values and the currently used cell-ID using the Android Application Framework (cp. section 4.3.3.1).
3) The corresponding geographical location of the serving MNO cell is determined using the available online databases.
4) The distances on the X- ($DistX_{Phone,Cell}$) and Y-axis ($DistY_{Phone,Cell}$) between the phone and the serving MNO cell are calculated using the SLC formula.

   Example:

   $DistX_{Phone,Cell} = SLC(SC_{lat}, SC_{long}, SC_{lat}, PC_{long}) = -815m$

$DistY_{Phone,Cell} = SLC(SC_{lat}, SC_{long}, PC_{lat}, SC_{long}) = -446m$

The negative values in this example mean that the phone is to the west (X-axis) and south (Y-axis) of the serving MNO cell.

5) The absolute values of the two calculated distances $DistX_{Phone,Cell}$ and $DistY_{Phone,Cell}$ are divided by "Res" and the result is rounded up / down to the nearest integer. The calculated $DistBitsX_{Phone,Cell}$ (respectively $DistBitsY_{Phone,Cell}$) are the number of bits required to express the distance between phone and serving MNO cell. In this example each bit represents 100m.

   Example:

   $DistBitsX_{Phone,Cell} = round(abs(DistX_{Phone,Cell}) / Res) = round(815/100) = 8$

   $DistBitsY_{Phone,Cell} = round(abs(DistY_{Phone,Cell}) / Res) = round(446/100) = 4$

6) $BinString_{Default\{X,Y\}}$ is adjusted to represent the actual distance between phone and serving MNO cell. If the value of $DistX_{Phone,Cell}$ is less than 0 (i.e. the phone is to the west of the MNO cell), then $DistBitsX_{Phone,Cell}$ "1s" in $BinString_{DefaultX}$ are changed to "0s". Otherwise, if $DistX_{Phone,Cell}$ is greater or equal to 0, then $DistBitsX_{Phone,Cell}$ "0s" in $BinString_{DefaultX}$ are changed to "1s". The same operation is then performed for the Y-axis.

   Example:

   a) $DistX_{Phone,Cell}$ is -815m and, as a result, eight "1s" will be changed $BinString_{DefaultX}$ to "0s" to represent the actual distance in X expressed by:
      $BinString_X$ = 11000000000000000000
   b) $DistY_{Phone,Cell}$ is -446m, thus four "1s" will be changed to "0s" in $BinString_{DefaultX}$ to represent the actual distance in Y expressed by:
      $BinString_Y$ = 11111100000000000000

7) $BinString_X$ and $BinString_Y$ are concatenated to form the binary location string "$LocBinString_{PC}$", which represents the actual distance between phone and serving MNO cell.

   Example:

   $LocBinString_{PC}$ = 11000000000000000000 11111100000000000000

The resultant binarised location "$LocBinString_{PC}$" is then fed into the PBS as shown in step 5 in Figure 16.

On the authenticator side, the above algorithm steps are repeated. The difference is that the authenticator uses the client's phone location (PA_{lat}, PA_{long}) provided by the MNO to calculate the corresponding binarised location $LocBinString_{PA}$.

Example:

$LocBinString_{PA}$ = 11110000000000000000 11111111111000000000

The actual distance between phone and serving MNO cell represented by $LocBinString_{PC}$ and $LocBinString_{PA}$ can now be calculated using the XOR-function.

Example:

$LocBinString_{Dist} = LocBinString_{PC} \oplus LocBinString_{PA}$

$LocBinString_{Dist}$ = 00110000000000000000 00000011111000000000

The remaining "1s" in $LocBinString_{Dist}$ multiplied by "Res" represent the distance between the GPS-based phone location and the phone location provided by the MNO in X and Y respectively.

Example:

$DistX_{PC,PA}$ = 2 * Res = 2 * 100m = 200m

$DistY_{PC,PA}$ = 5 * Res = 5 * 100m = 500m

In MORE-BAILS, ECC_{Loc} (step 6 in Figure 16) deals with the fuzziness of the two binarised locations and corrects in the current setup up to 10 bit difference between $LocBinString_{PC}$ and $LocBinString_{PA}$.

3.2.2.4 Privacy Preserving Usage of Location Information

Introduction of client's physical location into MORE-BAILS raises "privacy concern", i.e. tracking clients' location without their consent. To overcome this concern, LOTA developed methods that preserve the client's location privacy and, at the same time, enable the authenticator to verify the client's claimed location independently [75].

SOMs (cp. section 0) are used in MORE-BAILS to generate a transformed version of any location information to incorporate the location privacy protection. The actual location of the client is never disclosed to anybody including the authenticator. The client applies the SOM on the obtained GPS-based location prior to the location binarisation process (cp. step 4 in Figure 15). At the authenticator side, every time the authenticator requests the client's location from MNO, the MNO responds by sending a transformed version of the client’s current location based on the usage of the same SOM (cp. step 3 in Figure 16).

To verify the actual distance between the original two locations, applying SOMs must preserve this distance. This was proven for the Euclidean distance [76]. Although MORE-BAILS uses the SLC formula as distance measure between the two geographical locations on earth, the difference between the Euclidean distance and the used SLC distance is negligible when dealing with a few hundred metres (cp. section 3.2.2.1).
Simulations showed that the average error introduced by the SOM transformation using SLC is about 0.4%, i.e. if the distance between the original two locations is 100m, then the distance between the two transformed locations will be 100 ± 0.4m.
The current MORE-BAILS scheme configuration tolerates up to 320m difference (cp. section 3.2.2.1). This means that the error introduced by the privacy-preserving location authentication is small, just over one metre.

3.2.3 Results, Discussion and Conclusion on MORE-BAILS

This section presents the experimental results and discusses the security impacts in the cases that one or more of the employed AFs are compromised. The section finishes with a conclusion about the proposed MORE-BAILS authentication scheme.

In the current MORE-BAILS scheme, five AFs are used:

1) Biometrics, as a “something you are” AF that are captured from the client during enrolment and stored in the authenticator’s database (cp. section 2.2.2).
2) PIN, as a “something you know” AF that is agreed between client and authenticator and stored in the authenticator’s database (cp. section 2.2.1.1).
3) KoP, as a “something you have” AF (cp. section 2.2.1.2). The KoP is stored in the authenticator’s database and on the phone, e.g. main memory or SIM card.
   It is important that the KoP is a unique, randomly generated key. It is not recommended to use already available Mobile Communication Subscriber Information (MCSI) as KoP, e.g. the phone’s IMEI number (cp. section 4.1.1).

This MCSI is not completely secret and can therefore easier be discovered than a randomly generated key [55].

4) Location, independently verified via two different localisation techniques (cp. section 2.2.3).

   If desired and agreed by client and authenticator, the use of the MORE-BAILS protected application can be limited to certain geographical areas. This limitation is useful to prevent the application usage from "unusual" locations, because the phones was, for example, lost or stolen on holidays. In this case, an attacker cannot use the application from Asia, if the application is registered for the UK. To use the "area limitation" feature, the authenticator stores the client's areas of operation. "Area limitation" is integrated into MORE-BAILS as an optional feature, because it introduces an additional burden and sometimes undesired restriction to the client. "Area limitation" can be disabled, if the client:

   a) does not want to be limited in where s/he is able to use the application,
   b) does not accept the additional efforts required to continuously update his/her area of operation, because s/he is a frequent traveller.

5) Time to uniquely stamp the authentication attempt (cp. step 2 in Figure 15).

The MORE-BAILS scheme is designed to achieve the highest possible security. However, depending on lower security requirements of some applications, it is possible to drop one or more of the MORE-BAILS AFs to minimise the impact on the client during the application usage. For example, the KoP value could be used as the PIN, too. This releases the client from remembering another PIN, but lowers the MORE-BAILS security. However, in all simulated scenarios (see below), all five AFs are present in MORE-BAILS. The number of all possible scenarios (combinations) of compromising zero, one or more of the MORE-BAILS AFs is calculated by formula (3).

$$\sum_{k=0}^{n} \binom{n}{k} = \sum_{k=0}^{5} \binom{5}{k} = 32 \qquad (3)$$

Where:	n=5	: Number of AFs.
	k	: Number of compromised AFs to be simulated.

It is not feasible and reasonable to simulate all 32 scenarios, because some of these scenarios are only theoretically possible and therefore unlikely to happen in real life. For example, if an attacker is able to get hold of a client’s phone, the attacker automatically knows the KoP, as well as L&T. Accordingly, it is not reasonable to simulate a scenario in which KoP and location (but not time) are compromised AFs. Therefore, the following 14 most relevant scenarios are simulated. It is assumed that all AFs, which are not compromised, are secure in the scenario.

Scenario 1) “Comparison scenario” that uses Biometrics only, no other AF is employed. This scenario is used to compare the performance achieved by biometrics only with the performance of the proposed and evaluated multi-factor MORE-BAILS scheme.

Scenario 2) “All secure”, using all AFs under the assumption that all of them are secure and none is compromised.

Scenario 3) Simulating a scenario where the Biometrics are compromised.

Scenario 4) Simulating a scenario where the PIN is compromised.

Scenario 5) Simulating a scenario where the KoP is compromised. This is possible when an attacker gets access to the client’s phone where the KoP is stored.

Scenario 6) Simulating a scenario where Location is compromised. This could happen when an attacker is physically close enough to the client or knows the client’s current whereabouts.

Scenario 7) Simulating a scenario where Time is compromised.

Scenario 8) Simulating a scenario where PIN and KoP are compromised. In this scenario, the attacker knows the client’s personal credentials (for example through previously successfully employed phishing attacks), but does not know the client’s current location.

Scenario 9) Simulating a scenario where Location and Time are compromised. In this scenario, the attacker knows the client’s current whereabouts, but does not know the client’s personal credentials.

Scenario 10) Simulating a scenario where PIN, KoP, and Location are compromised. In this scenario, the attacker knows the personal client credentials and also knows or is able to estimate the client's current location.

Scenario 11) Simulating a scenario where KoP, Location, and Time are compromised. In this scenario, the attack has full access to the phone (because it was lost or stolen), but the attacker does not know the client's personal PIN.

Scenario 12) Simulating a scenario where PIN, KoP, and Time are compromised. In this scenario, the attacker knows the client's personal credentials and is also able to create a valid timestamp.

Scenario 13) Simulating a scenario where PIN, KoP, Location, and Time are compromised. In this scenario, the attacker knows the client's personal credentials and also knows the client's current whereabouts.

Scenario 14) Simulating a scenario where all AFs are compromised. This scenario is included for completeness. No security can be expected in the case that the attacker knows all used AFs.

In the first scenario, the performance of a biometrics only system is evaluated. The obtained results are then used to compare the performance of the proposed MORE-BAILS scheme against the biometrica only system. Additionally, this biometrics only scenario result serves as the decision-making basis, which Haar subband (cp. section 2.2.2.1) is used in the further scenarios (2 to 14) to represent the biometric AF.

Table 3 shows the obtained face recognition accuracy of the biometrics only system, in terms of the Equal Error Rates (EERs %) of the four wavelet subbands at the third level of decomposition (LL_3, HL_3, LH_3, and HH_3), and across all the five subsets of the extended Yale-B dataset (cp. section 3.2.2.2). As shown in Table 3, apart from subset 3, the LH_3 subband achieves better authentication accuracy compared to the other subbands. Only in subset 3 the HL_3 subband slightly outperforms LH_3. Consequently, the LH_3 subband coefficients are selected to represent the biometric AF for the rest of the experiments.

Table 3: Face recognition accuracy of biometrics only system

	LL_3	**HL_3**	**LH_3**	**HH_3**
Subset 1	0.00	0.00	0.00	0.00
Subset 2	0.90	0.81	0.04	3.36
Subset 3	1.33	0.41	0.56	0.71
Subset 4	15.48	14.57	6.06	9.36
Subset 5	17.15	22.25	4.53	11.36

Table 4 reports the performance of the proposed MORE-BAILS scheme in terms of FAR and FRR for the 14 simulated scenarios.

Table 4: MORE-BAILS scheme performance

		Subset 1		**Subset 2**		**Subset 3**		**Subset 4**		**Subset 5**	
Simulated scenario		**FAR (%)**	**FRR (%)**	**FAR (%)**	**FRR (%)**	**FAR (%)**	**FRR (%)**	**FAR (%)**	**FRR (%)**	**FAR (%)**	**FRR (%)**
1	Biometrics only	0	0.00	0.04	0.04	0.56	0.56	6.06	6.06	4.53	4.53
2	All secure	0	5.00	0	5.03	0	5.53	0	10.75	0	9.30
3	Biometric compromised	0	5.00	0	5.03	0	5.53	0	10.75	0	9.30
4	PIN compromised	0	5.00	0	5.03	0	5.53	0	10.75	0	9.30
5	KoP compromised	0	5.00	0	5.03	0	5.53	0	10.75	0	9.30
6	Location compromised	0	5.00	0	5.03	0	5.53	0	10.75	0	9.30
7	Time compromised	0	5.00	0	5.03	0	5.53	0	10.75	0	9.30
8	PIN & KoP compromised	0	5.00	0	5.03	0	5.53	0	10.75	0	9.30
9	Location & Time compromised	0	5.00	0	5.03	0	5.53	0	10.75	0	9.30
10	PIN & KoP & Location compromised	0	5.00	0	5.03	0	5.53	0	10.75	0	9.30
11	KoP & Location & Time compromised	0	5.00	0	5.03	0	5.53	0	10.75	0	9.30
12	PIN & KoP & Time compromised	0	5.00	0	5.03	0	5.53	0	10.75	0	9.30
13	PIN & KoP & Location & Time compromised	0	5.00	0.038	5.03	0.53	5.53	5.76	10.75	4.30	9.30
14	All compromised	95	5.00	94.97	5.03	94.47	5.53	89.25	10.75	90.70	9.30

To correctly evaluate the MORE-BAILS security and obtained performance results, it is important to recall the functioning of the MORE-BAILS scheme. At the authenticator side, the MORE-BAILS process goes through three sequential stages (cp. Figure 16):

1) Successful ECC_{Bio} decoding (step 1 in Figure 16) authenticates two AFs: Biometrics and UBRP-transformation key (i.e. PIN).
2) Successful ECC_{Loc} decoding (step 6 in Figure 16) authenticates two more AFs: Location and shuffling key.
3) Obtaining fresh and valid time T_C (step 9 in Figure 16) authenticates both KoP and timestamp.

This MORE-BAILS design offers the following security advantages:

1) A failure in any of these three stages leads to failure of the entire authentication process. This makes the proposed MORE-BAILS scheme secure, because an attacker needs to break all employed AFs. This also explains the 0% FAR in all scenarios from 2 to 12 in Table 4. I.e. an attacker is unable to compromise the MORE-BAILS scheme, even when the attacker has a number of AFs compromised.
2) The proper combination (XORing at three levels: steps 2, 6, and 11 in Figure 15) of the AFs to produce the OTMFBR makes the transmitted OTMFBR leak no information about the employed individual AFs. This ensures that an attacker cannot benefit from a partially successful attack (i.e. the attack has breached some of the AFs) for later attacks.
3) The client's claimed location is verified via a second, completely independent localisation technique on the authenticator side (cp. section 2.2.3.1). This ensures that the client cannot forge his/her location.
4) Integration of GPS real-time into OTMFBR makes the MORE-BAILS scheme robust against replay attacks, i.e. intercepted MORE-BAILS data-messages cannot be replayed because of the timestamp expiry.

A detailed analysis of Table 4 discloses the following observations:

1) In the first scenario (face biometrics only) the use of LH_3 as an efficient feature extraction method has been simulated. The operating point (decision threshold) for each subset is selected at the EER, i.e. FAR = FRR.

2) From scenario 2 to 12, the proposed MORE-BAILS scheme achieves 0% FARs against any attacking attempt even when one or more AFs are compromised.

   Although the MORE-BAILS scheme achieves the same FAR and FRR in all scenarios 2 to 12, the scenarios do not offer the same level of security. An attacker in scenario 2 “All secure”, for example, needs to guess / fake all AFs. In scenario 8, the attacker needs to break only two AFs (PIN and KoP). This means that scenario 2 is much more secure compared to scenario 8. This fact illustrates that FARs and FRRs on their own are not sufficient enough to evaluate the security of multi-factor schemes. Hence, LOTA concluded that other security characteristics such as the robustness against attacks (e.g. replay attack) and the level of difficulty to break an AF should be taken into consideration. This will be further investigated by LOTA.

3) The FRR of “All secure” and the other scenarios (3 to 14) is higher than the biometrics only scenario because of the used location AF. Table 1 shows that two genuine locations out of the twenty collected locations will be falsely rejected. I.e. 5% of the FAR in all these scenarios (2 to 14) that are using location comes from these (out of range) measurements. For example, the FAR in the biometrics only scenario is 4.53% in subset 5, i.e. 95.47% of authentication attempts will be accepted in this first scenario.

   Location verification (step 6 in Figure 16) takes place after biometric verification (step 1 in Figure 16). This means that 5% of the 95.47% will be falsely rejected, i.e. the overall FAR of scenarios 2 to 14 will be 9.3%.

4) The falsely rejected location AF mentioned in 3) also explains the lower FAR in scenario 13 (all AFs except biometrics are compromised) compared to the first scenario (biometrics only). It can be expected that the FARs in these two scenarios are equal. However, the location FRR contributes to a lower FAR in scenario 13, because the first two location measurements will be rejected (step 6 in Figure 16), i.e. the FAR percentage becomes lower.

5) It is feasible in “All secure” scenario and the other scenarios (3 to 14) especially when using two or more secure AFs to get less FRRs. This requires adjusting the operating threshold of biometric and / or location AF, e.g. tolerating more than 191-bit for the face BVF (cp. section 3.2.2.2) or more than 320m for location (cp. section 3.2.2.1). However, this adjusted operating threshold results in high FAR

when other AFs are compromised. Therefore, the operating threshold of biometric / location AF should remain the same when using additional AFs [77].

To summarise, the MORE-BAILS authentication scheme has been proven to be:

1) Viable for use in any type of strong client authentication requiring mCommerce application, because of the robust and secure verification of the client's authentication data.
2) User-friendly, because the client only needs to capture his/her face image using the phone's camera and to enter their PIN. All other AFs (KoP, location, and time) will be silently and without the need of any further client action determined in the background though the MORE-BAILS scheme.
3) Easy-to-implement, because MORE-BAILS bases on the phone's onboard sensors to capture the client's biometrics, onboard GPS receiver for location, keypad for PIN, and SIM card to securely store the KoP.
4) Privacy preserving, because of the use of cancellable client biometrics and privacy preserving transformation of the client's location.

4 REVIEW OF SECURE AUTHENTICATION IMPLEMENTATION ON PHONES

This chapter describes three possible approaches to securely host and execute the authentication process on phones:

1) Using SIM cards to host authentication activity (cp. section 4.1),
2) Using custom chips for authentication (cp. section 4.2),
3) Using software-based protection of authentication process (cp. section 4.3).

Protecting the authentication process host environment is important in remote authentication from a security point of view, because:

1) It guarantees correct capturing and processing of the authentication data on the phone and thus, increases the authenticator's trust in the received client authentication data.
2) It stops attackers from using the authentication application on the phone to get illegitimately authenticated.

From a commercial point of view, protecting the host environment becomes even more important because:

1) Number of mCommerce applications that require secure client authentication increases. I.e. the protection technique needs to be adaptable to the security requirements of these varying mCommerce applications.
2) Number of clients using mCommerce applications on their phones increases. I.e. the protection technique needs to be user-friendly and cost-effective distributable to the clients.
3) Financial damage of successful attacks on mCommerce applications increases, because transactions with higher monetary value can be now performed on phones.

Each of the three approaches will be discussed based on criteria covering:

1) Offered security: Can the approach protect the authentication process execution and thus, guarantee the correctness of the execution? I.e. how difficult is it for an attacker to circumvent the integrated security mechanism?
2) Performance: Is it possible to execute the authentication process within a few seconds on the phone to guarantee that the client will accept (i.e. not refuse because of a long delay) the authentication technique in practical use?
3) Implementation complexity: Which resources (e.g. equipment and knowledge) are required to implement the approach and how complex is the implementation?
4) Flexibility: Can the authentication system design be changed during implementation and be tailored to further phone models?
5) Availability: Are all components required to implement the approach available for the mass-market?
6) Costs: How much does it cost to implement the approach on the phone and to distribute the developed solution to the client?

A literature review was conducted to understand the boundaries of each approach. The gained knowledge helped to establish if the approach offers a practical and commercial viable solution to secure the authentication process hosting and execution.

4.1 Using SIM Cards to host Authentication Activity

SIM cards are successfully used for decades to authenticate billions of subscribers to the MNO. SIM cards are small and portable, simple and convenient to use and familiar to all MNO clients. However, first trials to host the complete authentication process on the SecurePhone SIM card resulted in an immense run-time delay, which is inacceptable in real-world authentication applications (cp. section 1.1). LOTA investigated the technical limitations and security features of SIM cards to establish:

1) What types of SIM cards are currently available and what are their technical capabilities and limitations (cp. section 4.1.1)?
2) Can the authentication process security be strengthened by SIM cards? I.e.:
    a) Is it viable to perform the complete authentication process on the SIM card?
    b) Is it secure to perform only template matching on the SIM card, i.e. use the SIM card mainly as a secure storage device?

4.1.1 SIM Cards - Technical Background

Subscriber Identity Modules (SIMs) are small plastic cards with an embedded Integrated Circuit (IC). SIM cards are mainly used in telecommunication areas, where SIM cards store information about subscribers for mobile telephone services. Each SIM card has a unique IMEI, an Integrated Circuit Card International Identifier (ICCID), which identifies the SIM card chip, as well as stores the IMSI and the authentication key of the subscriber as part of the MCSI. SIM cards can be transferred between phones, which enable clients to switch their mobile network subscriptions easily.

SIM cards were designed for the use in the Global System for Mobile communications (GSM, 2G) networks (cp. Figure 19). Together with the standardisation and commercial availability of the 3rd generation (3G) mobile communication service (Universal Mobile Telecommunications System (UMTS)), a second generation of SIM cards, the Universal Subscriber Identity Module (USIM) was introduced in 2001. These USIM cards feature stronger subscriber authentication and encryption algorithms as well as more memory of up to 128 Kilobyte. LOTA uses “SIM cards” to denote 2G-SIM and/or 3G-USIM cards in the remainder of this thesis.

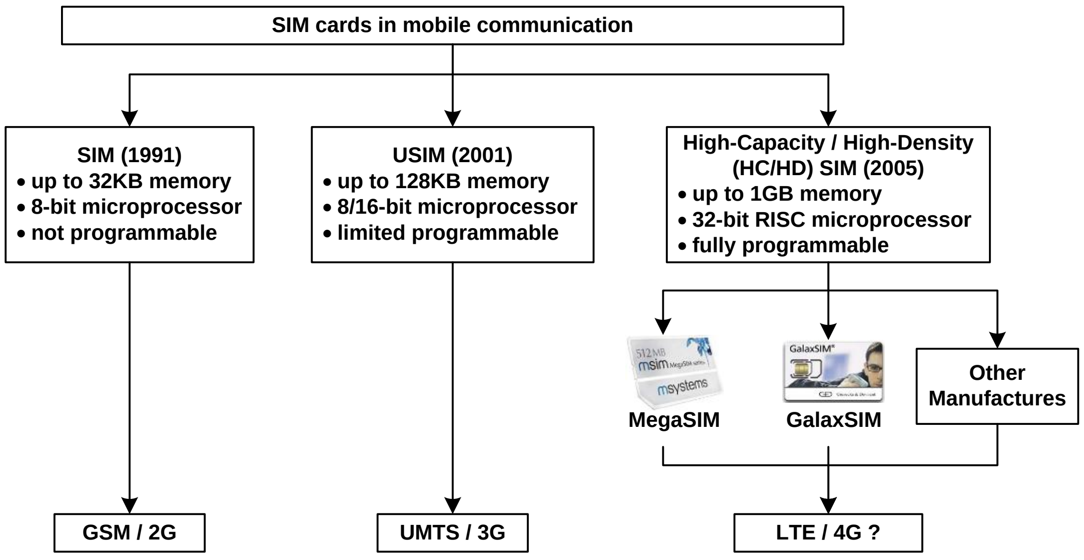


Figure 19: SIM cards in mobile communication

“High-Capacity / High-Density” (HC/HD) SIM cards were announced in 2005 as the next generation of SIM cards. HC/HD SIM cards like the MegaSIM (manufactured by M-Systems) or the GalaxSim (by Giesecke & Devrient) offer up to 1 Gigabyte memory, a full 32-bit RISC microprocessor architecture, “SecurCore” co-processors and various cryptographic algorithms for enhanced security and data access control, as well as an advanced Application Programming Interface (API) to develop HC/HD SIM card applications [78,79]. HC/HD SIM cards maintain backward compatibility to the 3G communication standard, but their main area of operation is expected to be 4G (Long Term Evolution, LTE).

4.1.2 Literature Review of SIM Card-based Authentication

SIM cards are used as a “secure” place to store authentication and client data as well as to authenticate the subscriber to the MNO. This literature review focuses on:

1) Can SIM cards support remote authentication over wireless channels?
2) Can SIM cards be used to implement more sophisticated, multi-factor authentication techniques (cp. section 2.2)?
3) Can SIM cards securely and reliably host the complete authentication process (cp. section 2.3.1)?

SIM cards can be used to enhance authentication and protect mCommerce applications as follows [80]:

1) The subscriber information stored on the SIM card is used to authenticate the phone to mCommerce applications [81]. In this approach, the phone becomes a mobile payment device that could replace for example credit cards. However, if the phone gets lost or stolen, then this approach does not add any security, because the attacker could illegitimately use it. I.e. a combination with further AFs (similar to the PIN for the credit card) is necessary to prevent unauthorised phone usage.
2) The SIM card stores additional information (e.g. encryption keys or certificates), that is then used to protect and authenticate the caller to the MNO or the transaction between client and authenticator [82]. Here, the SIM card acts as a secure storage device. However, if an attacker is able to read or change sensitive information on the SIM card, then authentication security is undermined [83].

These two approaches use the SIM card to store required authentication data and hence, support the authentication process performed outside the SIM card. To host the complete authentication process, for example, a web-server on a Java SIM card can be used [84]. A web application running inside this web-server handles the communication and access control between phone and SIM card. The web application is based on the Java 2 Micro Edition (J2ME). J2ME is a subset of Java designed for embedded systems with limited computational power [79], like SIM cards. The Security and Trust Services API of Java SIM cards and the encrypted communication between SIM card and phone proved to offer strong security mechanism [85]. However, although the J2ME environment simplifies application development for SIM cards and offers optimised security related functions, the available API to develop complete authentication systems on Java SIM cards is limited and not all security features and required computations of authentication systems can efficiently be implemented on SIM cards [86]. This is due to the limited computational capabilities of these SIM cards and the slow data communication rate between the SIM card's CPU and the phone's CPU [87]. In SecurePhone, these limitations increased authentication time from a few seconds to 45 minutes once the complete authentication process was executed on the SIM card.

SIM cards cannot be considered as 100% secure or tamper proof [83]. This is because logical, physical, or side channel attacks can be used to read or change sensitive data in the protected area of the SIM cards, so to:

1) clone complete SIM cards [88] for the illegitimate use in more than one phone,
2) change the subscriber and authentication data stored on the SIM card [89] to deceive the MNO,
3) remove the SIM card lock, e.g. of an iPhone [90], to use the phone with other MNOs.

The new generation of HC/HD SIM cards overcomes some of these drawbacks of SIM cards. HC/HD SIM cards feature much higher processing-power (up to 1GHz processor), larger memory (up to 1Gigabyte), advanced security and encryption functions as well as fast data communication based on the Universal Serial Bus (USB) architecture or the memory card interface (MultiMediaCard) between HC/HD SIM card and phone. This may allow performing the complete authentication process inside the secure HC/HD SIM card environment. However, although several card manufactures already announced production of HD/HC SIM cards in 2005/6 [78], no HC/HD SIM card has been deployed to the end-consumer phone market thus far for the following reasons:

1) Higher manufacturing costs for HD/HC SIM cards, which are double the costs of SIM cards [78].
2) Costs to implement the required faster HD/HC SIM card communication interface on the phone [79].
3) Deployment of new HD/HC SIM cards to existing clients is difficult, because billions of SIM cards are already sold and in use worldwide. An exchange of these SIM cards to new HC/HD SIM cards is a big logistic challenge and introduces further costs [79].

4.1.3 Conclusion on SIM Cards to host Authentication Activity

The use of SIM cards as a secure host to execute the entire authentication process is practically limited at this point in time. Further techniques to protect the authentication process on the phone are required.

Due to low processing power and slow data exchange rate between SIM card and phone, SIM cards are currently mainly used to store subscriber data [91]. The storage area offered by SIM cards can support the authentication process. Client credentials or cryptographic keys can be stored on the SIM card and integrated as AFs into the authentication process, i.e. the SIM card acts as a “secure” storage device. Due to the fact that SIM cards are not 100% secure or tamper proof, additional security features should be used. MORE-BAILS (cp. section 3.2) combines authentication data stored on the SIM card (KoP) with further client specific AFs (e.g. PIN). The KoP ensures, together with the employed client specific PIN and location that the used phone belongs to the genuine client and thus, helps increasing the mCommerce authentication security.

The new generation of HC/HD SIM cards are not yet available for the end-consumer market. Once these advanced cards are deployed, they could offer a secure host environment to perform the complete authentication process on the card.

4.2 Using Custom Chips for Authentication

LOTA concluded in section 4.1.3 that currently available SIM cards are not a viable option to securely host the authentication process. LOTA identified custom chips as a second possible option to securely host the authentication process and explored:

1) What are the technical impacts, e.g. implementation complexity, memory and performance, hardware and space requirements of custom chips?
2) What are the costs for implementation, distribution and integration of custom chips into phones?
3) How does a custom chip for authentication compare to an authentication solution implemented in software and executed on the phone's main processor?

A custom chip is an application-specific hardwired microchip that performs most / all relevant tasks of a system within one single chip. Custom chips can be achieved by a full-custom chip or a programmable chip [92]. Authentication systems can be implemented on full-custom chips or on programmable chips. The chosen type of custom chip impacts the financial and human resources required to design and manufacture the custom chip [93].

Custom chips can be used for authentication on phones in two ways:

1) The custom chip performs all authentication tasks (from BDP/FE to decision making) [94]. No other hardware / software component son the phone or external authentication resources are used.
2) The custom chip is used to be part of other components either on the same phone (e.g. tasks implemented in software and executed inside the phone's main processor) or on external resources (e.g. remote authentication server) [95].

The second option is of interest to LOTA, because LOTA provides solutions for remote client authentication (i.e. the authenticity decision is done remotely by the authenticator). In this option, the custom chip is used to perform the BDP/FE and to generate the authentication-data message (cp. Figure 12). I.e. a secure implementation of these components in a custom chip can eliminate the attack points 3, 4, and 9 shown in Figure 12. However, custom chip designs cannot be considered as 100% secure or tamper-proof. Beside the implementation errors, which can occur during the complex design and lead to security flaws in the implemented system, custom chips are vulnerable to different types of attacks [96]. For example, a fault injection attack was used to extract the encryption keys of the Advanced Encryption Standard (AES) implementation on current Altera and Xilinx custom chips [97]. This attack can be similarly used to modify the BDP/FE component (cp. attack point 3 in Figure 12) or overwrite the generated client's BFV (cp. attack point 4 in Figure 12) in biometric-based authentication system (cp. section 2.3.1.) implemented as a custom chip.

Beside the fact that custom chips do not offer 100% security, the use of custom chips for authentication on phones introduce the following drawbacks [98]:

1) Integration of the custom chip requires space on the phone's main board. It also requires an interconnection and hence compatibility between the custom chip and the rest of the circuitry of the phone platform. This is critical because of the limited available space on the phone's main board.
2) Design, manufacturing and integration of custom chips for authentication on phones involve high costs.
3) The long and rigid development cycles of custom chips can hamper the update of existing or integration of the newest authentication algorithms into the client's phone.

LOTA has therefore concluded that, even if there are custom chip solutions that might come to light in the future, then opting for a software-based solution allows implementation of these solutions in software with similar gains, more flexibility and less costs [98].

4.3 Software-based Protection of Authentication Process

LOTA so far concluded that neither SIM cards (cp. section 4.1.3) nor custom chips (cp. section 4.2) are optimal options to enhance security of the authentication process execution on phones. LOTA continued with an investigation, how the resources of the phone (e.g. processor, memory etc.) can be employed to overcome the identified drawbacks of SIM cards and custom chips (e.g. limited processing power, high development and integration costs). Using the phone onboard resources has two advantages:

1) Current and future generations of phones include powerful processors and large memory that can be used to implement additional security features.
2) No additional internal or external hardware is needed. This reduces the overheads of implementing authentication techniques on the phone.

Software protection uses the phone onboard resources to secure software against illegitimate use or modifications. In remote authentication, software protection can further protect the mCommerce application on the phone against misuse. Software protection therefore offers two security advantages for applications requiring remote authentication:

1) Such protected applications cannot be used by an attacker even if the phone is lost or stolen.
2) The authenticator will be assured that the authentication software was executed correctly on the phone and that the authentication process was not tampered with, i.e. the authentication data received from the client was properly calculated on the client’s phone.

Investigation of software protection and the experience gained from work on SecurePhone led to the conclusion that combining software protection with biometrics and location can:

1) Bind the correct application execution tightly to the genuine client to prevent an attacker to use the application even if the phone is stolen.
2) Verify the proper functioning of the application to ensure correct authentication data collection and processing.
3) Include location to challenge the client in a challenge / response technique to ensure freshness and real-time of the authentication attempt.

4.3.1 Literature Review of Software Protection for Authentication Security

Since the early 1980s [99], various software protection techniques were proposed [100]. These techniques differ in the underlying assumption of the attacker capabilities and access to the protected system (attack model) as well as in the aim (protection type) of the software protection. Capabilities of an attacker to attack protected software can be distinguished into three attack models [101]. Software protection on phones can be within all three models, depending on the usage scenario.

1) Black box model (attacker has only external access to the functionality of the software):
   An attacker who remotely tries to take over an mCommerce transaction from a client has “black box” access to the transaction and software. This attack can be considered as a distance attack without any further internal knowledge of the phone and / or used software.
2) Grey box model (leakage information is available to the attacker):
   An attacker gains additional knowledge about the phone or software used for the transaction, which can be used to simplify later attacks. This can be happening, if an attacker is in close proximity to the client, whilst the client performs a transaction in a public place.
3) White box model (attacker has full control over software and execution host):
   In the case that the client’s phone is in possession of the attacker (e.g. because the phone was stolen), the “white box” model has to be applied.

Beside the attack models, the software protection technique used to secure the application on the phone is important. Software protection techniques can be divided into three categories, depending on the objective of the protection technique [102]:

1) Software tamper resistance, to protect against application integrity threats.
2) Software obfuscation, to avoid reverse engineering.
3) Software watermarking, to discourage intellectual property theft or prove ownership.

LOTA provides solutions to ensure and protect the authentication process integrity and correct process execution. Therefore, software tamper resistance (cp. section 4.3.1.1) and software obfuscation (cp. section 4.3.1.2) techniques are investigated and assessed in detail. Watermarking techniques will not be considered any further in LOTA.

4.3.1.1 Software Tamper Resistance

Software tamper resistance tries to detect and prevent illegal attempts to modify the software [103]. To achieve software tamper resistance, the software application contains two additional functions:

1) A tamper detection function to identify possible attacks.
2) A tamper response function to organise the countermeasures against an attack.

Tamper detection was formerly based on static check-summing techniques [103]. A major drawback of these techniques is that they are simple to locate inside the software code, because static check-summing is an unusual operation in software. Once an attacker has identified the check-summing segment, this technique can be easily bypassed, because the attacker needs to modify only one single-point to avoid the check. Dynamic integrity checking techniques try to overcome this drawback of static check-summing by eliminating the special code segment [104]. Instead, the dynamic integrity checks are spread all over the code and are closely integrated into the application operation flow. This makes it more difficult for an attacker to identify and remove these code segments. However, dynamic integrity checking techniques also introduce new drawbacks. Integration of many code-checking segments adds a substantial computational overhead to the software execution and it is more difficult to spread the code-checking segments uniformly and unnoticeable over the entire code.

Not all tamper detection techniques are suitable and applicable on phones, e.g. running on the Android OS, because they require, for example, modifications of the binary source code [105], which is not available on Android phones (cp. section 4.3.3.2). Also, the application byte-code verifier inside the virtual machine (VM, cp. section 4.3.3.3) used on these phones restricts the code modifications, which can be used in the tamper detection function. Control flow checking of Java applications, based on regular expression statements to describe and check valid application flows, is not rejected by the byte-code verifier [106]. The additionally required Java instructions to implement the regular expression comply with normal application statements and thus, are not detected by the byte-code verifier.

To overcome software tamper resistance, attackers use reverse-engineering techniques as a first step in the attempt to modify the protected application. Software reverse-engineering transforms low-level source code into a higher level of abstraction [107], e.g. binary code into C source code. Once an application is successfully reverse-engineered, the source code can be easier understood and subsequently modified. To make reverse-engineering more difficult, software obfuscation techniques (cp. section 4.3.1.2) are applied on the software code.

4.3.1.2 Software Obfuscation

Software obfuscation transforms a software application into an equivalent application that has the same behaviour but is harder to understand and thus, becomes more difficult to reverse-engineer. Ideally, software obfuscation behaves similar to public-key cryptosystems wherein it is more costly to decrypt a message without the key (i.e. deobfuscate the application) than it is to encrypt the message (i.e. obfuscate the application). However, it is impossible to obfuscate application codes systematically and only a few non-reversible (i.e. one-way) obfuscation technique exist [108]. For example, layout transformations (e.g. removing the source code format) are one-way obfuscation techniques. All obfuscation techniques that change the application structure (e.g. introduce additional if / switch branches) are reversible, because they have to be reversed during normal application execution to allow the application to run. An attacker can perform the same steps to reverse (deobfuscate) the application code as performed during the normal run, if the attacker knows these steps. In addition, deobfuscation of certain obfuscation techniques is NP-Easy and therefore not always a challenge for an attacker [109]. But nevertheless, it is common sense in the software protection research community that obfuscation is able to “raise the bar” for an attacker substantially, if the obfuscation techniques are implemented properly. For example, a layout “identifier renaming” obfuscation technique results in double the time needed to understand the real semantics of an application even against skilled attackers [110].

The security added by an obfuscation technique against reverse-engineering depends on the following characteristics [111]:

1) Sophistication / Strength of the obfuscation transformations employed.
2) Deobfuscation knowledge and experience of the attacker (i.e. the deobfuscator).

3) Power of the available deobfuscation algorithms.
4) Amount of resources (e.g. time, computational power) available to the attacker.

Execution of obfuscated software comes with a trade-of between the added security and the run-time overhead introduced by the obfuscation technique. Layout obfuscation techniques (e.g. renaming of variable names or removal of formatting) do not decrease the run-time efficiency of the obfuscated application when executed on the phone. All more complex obfuscation techniques, which are consequently more difficult to attack, introduce one or more of the following drawbacks:

1) Increased application size, i.e. more instructions in the code.
2) Run-time overhead, i.e. more instructions to execute.
3) Larger memory requirement, i.e. to store or execute the application.

The practical impact of these drawbacks on the actual execution of the obfuscated application depends on the host platform (e.g. the phone) that runs the application. Powerful computers are able to handle more complex and computational intensive obfuscation techniques than small and constraint phones. In general, the increased application size has the least and the additional required memory has the most critical impact on the possibility to run an obfuscated application on the intended host platform [111]. This impact is even more important on phones, because the available memory to execute the application is much smaller compared to the system storage. If the phone does not have enough memory to execute (i.e. to deobfuscate) the obfuscated application, then the application will not run at all.

As well as these drawbacks, the added security is an important factor to evaluate the quality and effectiveness of obfuscation techniques. Software obfuscation techniques can be evaluated using the following criteria [112]:

1) Potency: How much more difficult is it for a skilled human attacker to understand the application after obfuscation?
2) Resilience: How good is the obfuscation technique to withstand deobfuscation attempts performed by specialised deobfuscation computer applications?
3) Execution Cost: How much computational overhead or memory requirements are added to the application execution?

Similar criteria can be used to evaluate the quality of obfuscation techniques for Java [113]. Java is of particular interest to LOTA, because all applications running on Android Smartphones are written in Java and then compiled to Dalvik byte-code (cp. section 4.3.3). Evaluation models for obfuscation were then extended by the integration of the attacker's aim as a further criterion [114]. Without defining the attacker's aim as well as considering the attack model (cp. section 4.3.1), the obfuscation technique cannot be correctly evaluated [115].

Software obfuscation techniques are categorised into the following groups [112]:

1) Layout obfuscation techniques, which remove the source code formatting or the intended meaning of procedures and variables by changing their names [116]. Potency and resilience of these techniques are low, but they introduce no extra execution costs. To strengthen potency and resilience of layout obfuscation techniques, more advanced renaming techniques like overusing identifier names or introduction of illegal names can be used [117]. These advanced techniques confuse deobfuscation attempts using de-compiler applications, e.g. Jad – the fast Java de-compiler.
2) Data or operator level obfuscation techniques (e.g. hiding variables in bit-fields, splitting / merging variables or conversion of array structures) [118], which have a higher potency and resilience. However, these techniques add run-time execution overhead, e.g. operator level obfuscation techniques decrease the run-time performance by up to 40% [119].
3) Control flow obfuscations (e.g. adding dead code, additional control layers like if- / switch-statements, or modification of function and procedure calls) [120], which are the most powerful, yet most expensive, obfuscation techniques. Control flow obfuscation techniques decrease the run-time performance by up to 60% [119]. Inside the group of control flow obfuscations, integration of opaque variables achieves the highest potency and resilience, for example based on concurrency or exception handling obfuscation [121]. The value of an opaque variable is known at obfuscation time, but analytically difficult to determine for an attacker during deobfuscation. If opaque variables are combined with additional control layers and dead code, then it becomes difficult for an attacker to understand the real semantics of the application without knowing the correct values of the opaque variables [111].

Dynamic code mutation uses an opaque variable as a seed for a Pseudo-Random Number Generator (PRNG) [122]. The PRNG output is then used to control an edit script, which dynamically modifies the software instructions during run-time. Dynamic code mutation requires that the edit script is integrated as additional code into the application. This required integration can be exploited by an attacker. The attacker can try to identify the edit script inside the application code and remove it. Thus, further obfuscation techniques are required to protect the edit script, i.e. the edit-script must be hidden inside the application's code.

4.3.1.3 Obfuscated Interpretation

Obfuscated interpretation also uses an opaque variable to protect any type of software interpreted (i.e. executed) on ARM or x86 processors such as the embedded processors used in phones [123]. The novelty of obfuscated interpretation is that it obfuscates the application interpretation and not the application source code. Obfuscated interpretation was implemented in hardware as a Finite State Machine (FSM). This FSM "retranslates" the instruction stream of a previously obfuscated application into the original instructions during obfuscated interpretation of the application as shown in Figure 20 [123].

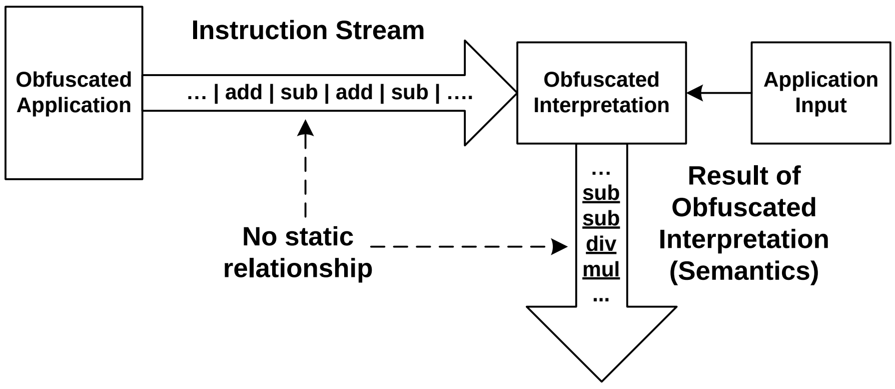

Obfuscated Application
Instruction Stream
… | add | sub | add | sub | ….
Obfuscated Interpretation
Application Input
No static relationship
…
sub
sub
div
mul
…
Result of Obfuscated Interpretation (Semantics)

Figure 20: Concept of obfuscated interpretation

The obfuscated application is "secured and protected" against reverse-engineering, because the attacker does not know the deobfuscation transition rules, i.e. how is an instruction deobfuscated. It is impossible for an attacker to understand the real semantics of the obfuscated application without having these deobfuscation transition rules, even if all instructions are available to the attacker.

Interpretation (deobfuscation) of an obfuscated instruction is not always the same. The result of deobfuscating an instruction depends on the obfuscated instruction and on the previous state of the FSM. For example, the first “add” instruction in Figure 20 is translated into a “sub” instruction, whereas the second “add” instruction is translated into a “div” instruction. No static relationship exists between the obfuscated and deobfuscated instructions, which makes it impossible to determine the next deobfuscated instruction without having the FSM transition rules.

Changing these transition rules is difficult in the hardware-based FSM implementation during development. This makes an application update costly and inflexible. Also, costs for integration of the FSM into the client’s phone and the FSM space requirements on the phone’s platform make the FSM uneconomical and complicated if applied to millions of phones.

The FSM requires the injection of dummy instructions into the obfuscated code to guarantee correct instruction interpretation regardless of the actual application input. Without these dummy instructions, for example, interpretation of branch instructions (e.g. if, case) can lead to wrong state transitions depending on the application input. These dummy instructions can leak information about the real semantics of the application code. This makes attacks easier, because a large number of dummy instructions are unusual in software code and thus, easy to identify for an attacker.

The framework for phone software protection [124] attempted to eliminate these security and flexibility problems by replacing the hardware-based FSM with a Permutation-Based Interpreter (PMI) implemented in software. The PMI allows an easier change of the transition rules, and also it does not require specialised interpretation hardware or dummy instructions to successfully execute an obfuscated application. Instead, the fixed FSM transition rules are replaced by adaptable, one-to-many mappings described in software, which become auxiliary input parameters (Au_X) to the PMI as shown in Figure 21 [124].

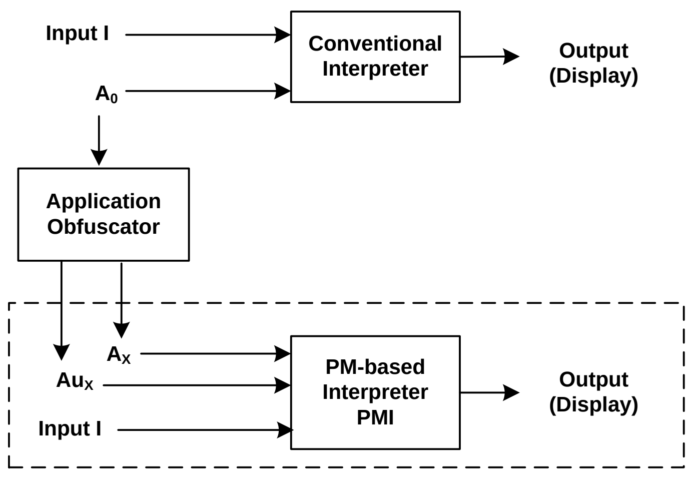


Figure 21: Software-based obfuscated interpretation framework

To obfuscate an application, the original application A_0 is translated by the obfuscator into an obfuscated application version A_X. Beside A_X, the obfuscator returns the auxiliary input Au_X (i.e. a one-to-many mapping key) of this obfuscation process. These two elements (A_X and Au_X) are used together with the client's application input I as input parameters to the PMI. The PMI is an extended "Conventional Interpreter" that has an additional permutation unit to deobfuscate A_X. During interpretation of A_X, the PMI retranslates the obfuscated application based on the current obfuscated instruction (e.g. addition) and the current Au_X value (e.g. 2). The one-to-many mapping of the PMI can be described by:

Map(obfuscated Instruction, Value of Au_X) $\rightarrow$Deobfuscated Instruction

This mapping guarantees that the same obfuscated instruction is mapped into different deobfuscated instructions depending on the current value of Au_X, e.g.:

Map(addition, 1)	$\rightarrow$	*subtraction*
Map(addition, 2)	$\rightarrow$	*division*
Map(addition, 3)	$\rightarrow$	*multiplication*
Map(subtraction, 1)	$\rightarrow$	*division*
Further mappings	$\rightarrow$	...

The PMI requires that the auxiliary input Au_X and the mapping rules are not available (i.e. they need to be protected) to an attacker. If an attacker knows Au_X and mapping rules, then the attacker could apply the same deobfuscation rules as done by the PMI and illegitimate execute the application. However, Au_X and mapping rules are required in the deobfuscation process if a genuine client wants to use the application. Thus, both elements must be stored on the phone and be accessible by the PMI. To protect the Au_X, it is encrypted with a unique key that is different in all PMIs and stored inside the PMI. This unique key further eliminates that an attacker can use knowledge gained in an attack to compromise other PMIs, because all PMIs use different keys to protect the Au_X. To “protect” the mapping rules and the Au_X encryption key, the PMI needs to be hidden as much as possible, for example by integrating the PMI source code inside the Java VM source code [124].

4.3.2 Conclusion on Software-based Protection of Authentication Process

Utilising the resources already available on the phone (e.g. processor, memory) in a software-based approach to protection the authentication process has two advantages:

1) No further costs for development and integration of supporting hardware (e.g. SIM card or custom chip) will arise.
2) Powerful processing resources are already available on today’s phones and can be consequently used straightaway. This reduces the time to deploy new versions of software-based solutions and simplifies the maintenance process.

mCommerce applications requiring secure authentication benefit from software protection. Software protection ensures the integrity and correct execution of the application on the phone. Protecting and verifying the software integrity during the authentication process execution:

1) eliminates attacks in which an attacker changes the application behaviour illegitimately to the attacker’s advantage to get authenticated (cp. section 2.3.1),
2) assures the authenticator that the authentication data is correctly collected and processed. This increases the authenticator’s trust in the received authentication data transmitted by the client and leads to more reliable authentication decisions.

Although tamper resistance and obfuscation techniques cannot fully protect an application against all types of attacks executed by highly skilled attackers [125], these techniques are able to “raise the bar” substantially and make attacks more difficult, time consuming and costly [110]. Layout obfuscation techniques (e.g.

identifier renaming) do not have the potency to withstand manual inspections by an experienced attacker. However, combining these layout techniques with control flow obfuscation techniques as proposed in the oBiometrics scheme [126] (cp. section 5.1) introduces a big challenge even for experienced attackers (cp. section 5.1.3).

The oBiometrics scheme uses opaque variables based on biometric generated keys (cp. section 2.2.2.4) to tightly bind the obfuscated application to the genuine client and thus, make it as difficult as possible for an attacker to bypass or remove the protection. Biometric keys are ideal to work similar to opaque variables, because the value of a biometric key is known after enrolment of the client, but difficult to determine in an attack. In addition to biometric keys, oBiometrics also uses the client's current location to generate location-based keys (cp. section 2.2.3.3) for obfuscation. Combination of biometric-based and location-based keys into an opaque variable for obfuscation tightly binds the correct application execution to the genuine client as well as ensures the current location of the phone.

Obfuscated interpretation implemented in software is used in oBiometrics to dynamically modify the application instructions based on the generated keys. In oBiometrics, the generated keys operate as the auxiliary input Au_X (cp. section 4.3.1.3). Because these keys are always freshly generated from the client's provided biometrics and the phone's current location, oBiometrics does not require Au_X to be stored and protected on the phone. Also, this eliminates that the interpreter and the mapping rules must be hidden in oBiometrics as necessary in the PMI [124]. Hiding the mapping rules and the interpreter inside the VM, as well as the decryption key for Au_X inside the PMI is critical. It is critical, because the Java VM is licensed under the GNU General Public License [127] and thus, the original VM source code is available to everyone. Identifying the PMI inside the VM extracted from the phone is easy, because the extracted VM can be compared to an unmodified VM version compiled from the original sources to identify any differences. Once the PMI instructions are identified inside the VM, these instructions can be de-compiled to locate the mapping rules and the auxiliary input Au_X decryption key. Thus, hiding PMI and the Au_X decryption key inside the VM does not provide enough security and strength against sophisticated attackers. Instead, it must be seen as "security through obscurity", which contradicts the principles of secure crypto systems [128].

A possibility to prevent the in-detail inspection of the Java VM would be to secure (i.e. obfuscate) the complete Java VM.

In oBiometrics, the freshly generated keys eliminate the need to securely store Au_X. The large key-space of the biometric- and location-based keys also removes the requirement to hide the mapping rules. Even if the mapping rules are known by an attacker, the attacker is not able to determine the correct deobfuscated instructions without having the correct key, i.e. the security of oBiometrics does not base upon "security through obscurity". Instead, oBiometrics security bases upon the large key-space and huge amount of possible combinations to deobfuscate the application instructions. This is important because phones are easily lost or stolen and thus, are fully available to an attacker for inspection. I.e. the white box attack model must be assumed as the attacker's capability level (cp. section 4.3.1).
The proposed oBiometrics scheme introduces only run-time overhead (cp. section 5.1.2). Neither additional application instructions are needed to obfuscate the application, nor is more memory required to execute an oBiometrics protected application. This makes oBiometrics a suitable obfuscation technique for phones.

After investigating advantages and disadvantages of SIM cards (cp. section 4.1) and custom chips (cp. section 4.2) to host the authentication process, an evaluation of security, performance and cost impacts of software protection was performed. Based on the presented and discussed results, LOTA concluded that software protection utilising the phone onboard resources outperforms the other two possibilities.
Section 4.3.3 will introduce the Android OS as the technical environment used in the software protection trials and experiments. The proposed obfuscated interpretation schemes to protect mCommerce applications on phones will be described in detail in section 5.1 (oBiometrics) and section 5.2 (LocAuth).

4.3.3 Experimental Environment for Authentication Software Protection

The Android OS was selected as the environment to implement the proposed schemes and to perform the practical trails and experiments. Android is a software stack especially designed for phones and invented by Android Inc. Since 2005, the development of Android is driven by Google Inc. and the Open Handset Alliance. Android was selected for the following two reasons:

1) Android is the Smartphone OS market leader. Android has a total market share of more than 50% and it is expected that this market share continues to rise [129].
2) The complete Android software stack, including OS, middleware and important system and user applications are available as Open Source. Android is distributed under the Apache Software License [130] that allows the use, modification and further distribution of the licensed software without charges as long as the new product is distributed under the same license, too. Because implementation of the proposed oBiometrics and LocAuth schemes (cp. sections 5.1 and 5.2) requires modifications of the OS (i.e. the Dalvik VM (DVM)), it is essential that the OS source code is freely available to perform practical trials.

4.3.3.1 Android System Architecture

The Android OS bases upon a modified Linux kernel as shown in Figure 22 [131]. Of particular interest for the implementation of the proposed schemes are the "Android Runtime" environment including the DVM, the user application layer ("Applications") and the "Application Framework" layer.

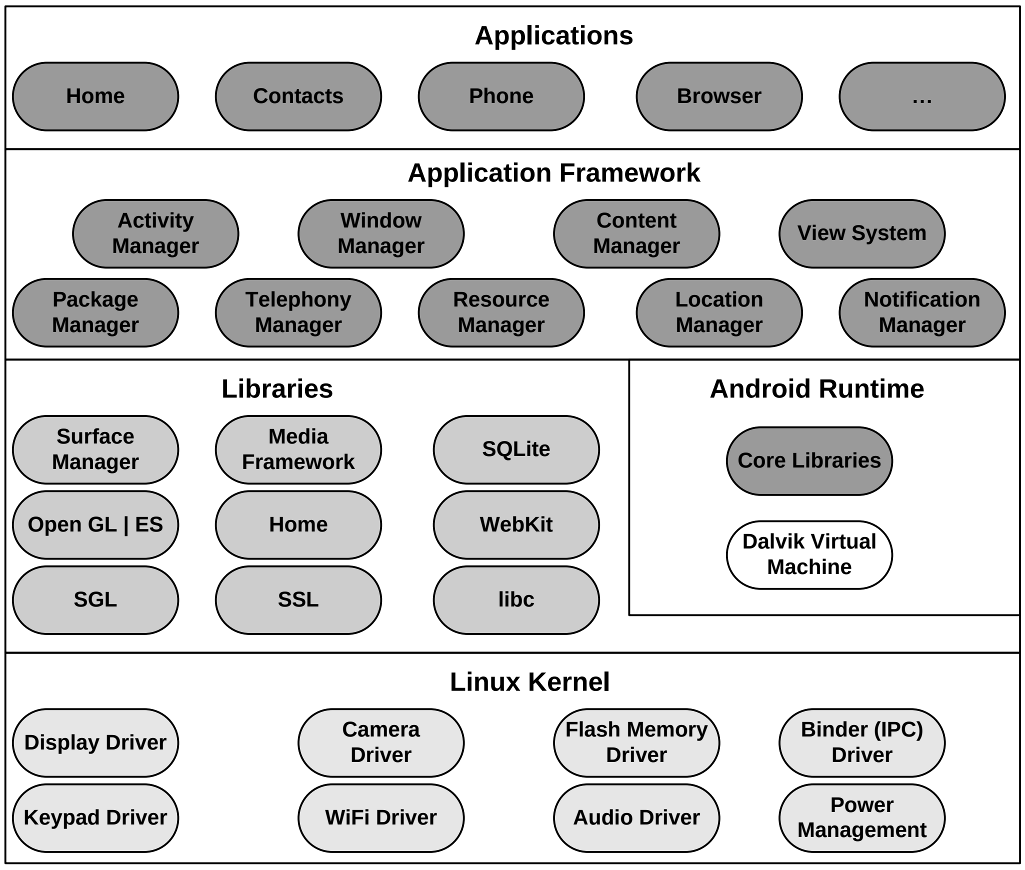


Figure 22: Android system architecture

All system and client applications from the applications layer are interpreted by the DVM. The developed MORE-BAILS, oBiometrics and LocAuth applications are also client applications and thus, elements of the applications layer after the applications have been installed on the phone. To execute these biometric- respectively location-obfuscated applications correctly, the functionality of the DVM was adapted. The modified DVM source code was then re-compiled and combined with the other parts of the Android OS (e.g. Linux Kernel, Libraries) to create a complete new system image to be used on the Android emulator and phones for the trials. All DVM modifications and application trials were performed on the Android Version 2.2 (Codename: Froyo). This was the state-of-the-art Android version when the proposed schemes were implemented for the trials.

The Application Framework layer provides an API to various sensors and receivers available on the phone, e.g. GPS receiver, camera, or microphone. LOTA uses the Application Framework:

1) To capture the required biometric data (e.g. the client's face) via camera and Hardware Manager.
2) To determine the current phone location via onboard GPS receiver and Location Manager.
3) To access the used mobile network communication properties (e.g. currently used cell-ID) via the Telephony Manager.

4.3.3.2 Android Application Development

Android applications are written in Java, which enables Android application developers to use the wide range of available Java software development environments. It is also possible to include code blocks or libraries written in native code (e.g. C or C++) for enhancing the performance of time critical code segments. Developed applications are then, in a first step, compiled using standard Java compilers to Java byte-code [132]. On the resultant Java byte-code, obfuscations techniques like control flow obfuscation [120] can be applied (cp. section 4.3.1.2). However, it is not possible to execute this Java byte-code directly inside the Android system, because Android uses a specially developed VM to interpret the byte-code, the DVM (cp. section 4.3.3.3). This DVM requires different byte-code instructions than the Java VM. To generate the required Dalvik instructions from the Java instructions, the Dalvik cross compiler "dx" is used. This "dx" tool translates Java byte-code instructions into the correct Dalvik format and creates a number of "dex-files", which are the basis for the oBiometrics framework (cp. section 5.1.2).

Android applications are distributed to the clients as Android packages (apk). An apk-file contains the generated byte-code dex-files as well as all other required resources (e.g. images, sounds, translation files, etc) of the developed application. The proposed MORE-BAILS, oBiometrics and LocAuth applications will be distributed to the enrolled clients as apk-files, too. Once a client receives one of these application packages, s/he has to install it on his/her Android phone using the normal installation procedure and the application is immediately ready to use.

4.3.3.3 Dalvik Virtual Machine

The DVM is a register-based VM, which interprets the Dalvik byte-code instructions, i.e. the DVM executes the client's application as shown in Figure 23.

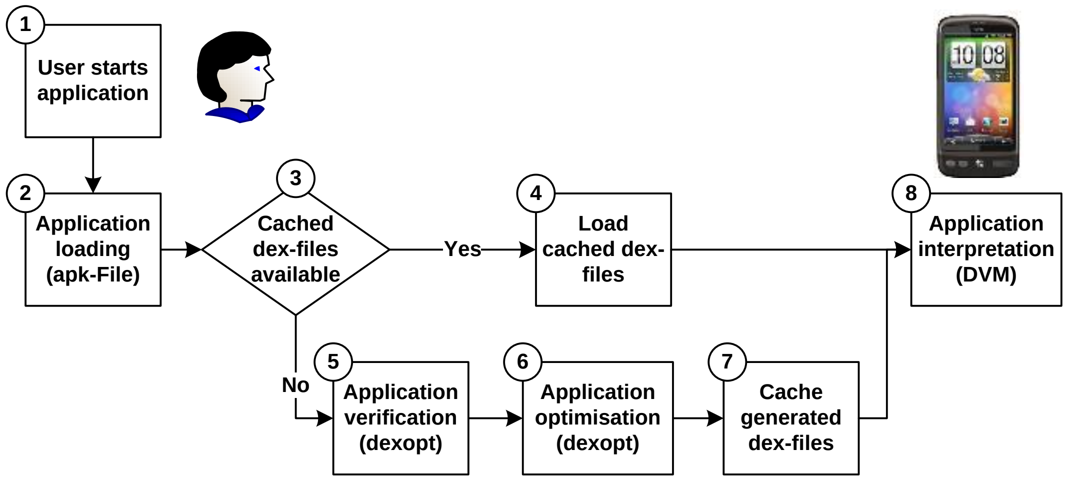

1
User starts application
2
Application loading (apk-File)
3
Cached dex-files available
Yes
No
4
Load cached dex-files
5
Application verification (dexopt)
6
Application optimisation (dexopt)
7
Cache generated dex-files
8
Application interpretation (DVM)
10 08

Figure 23: DVM steps performed during application execution

Before the byte-code is interpreted, the DVM performs a number of steps to load the application, as well as verifies the correctness and optimises the byte-code for faster interpretation.

1) A client starts any application on his/her Android phone.
2) The corresponding apk-file is loaded.
3) The DVM checks, if cached dex-files exist for the loaded application.
4) If these cached dex-files exist, then they are loaded and interpreted by the DVM.

5) If no cached dex-files exist, then the dex-files are extracted from the loaded application package and the “dexopt” tool verifies that no faulty byte-code sequences are present in any of the extracted dex-files. If a corrupted dex-file is identified, then the DVM immediately stops processing this application. The client is notified by the Android OS that this application cannot be executed. The benefit of this verification step is that no time-consuming verification is necessary during the actual application interpretation. This makes the application execute faster, because verified dex-files are cached. Thus, only one verification task is required for multiple executions of the same application.
6) The verified dex-files are optimised by the “dexopt” tool. During this optimisation step, various tasks are performed to simplify the byte-code instructions and subsequently increase the application interpretation speed. For example, method calls are replaced with the corresponding index in the method table to achieve faster access to the method implementation.
7) After all dex-files are optimised, they are stored in the OS application cache.
8) Finally, the DVM interprets the byte-code instructions of the dex-files and executes the application.

The byte-code verification carried out by the “dexopt” tool impacts the developed oBiometrics and LocAuth schemes. These schemes translate the application’s byte-code instructions into other instructions during the obfuscation process. To pass the DVM verification stage, certain rules to the obfuscation process must be applied, i.e. it is not possible to replace instructions randomly (cp. section 5.1.2). A similar limitation arises from the optimisation stage. The “dexopt” tool changes certain instructions at the optimisation stage for faster application execution as shown in step 6 of Figure 23. These instructions cannot be used for obfuscation, because they are no longer available after optimisation, i.e. in the interpretation stage where deobfuscation takes place. Performing deobfuscation prior to verification and optimisation to avoid these limitations is not possible due to the caching mechanism. If deobfuscation would take place before verification or optimisation, then the deobfuscated (i.e. unprotected) version of the application would by cached. This would enable an attacker to start and use the application without the need to provide fresh credentials for deobfuscation. For this reason, it is important that deobfuscation takes place in the last DVM step, the application interpretation.

5 OBFUSCATED INTERPRETATION TO ENHANCE AUTHENTICATION SECURITY ON PHONES

Section 4.3.2 concluded that software-based protection is the most viable option to protect the authentication process execution on the client's phone. This chapter further describes the concept and the prototype implementation details of the proposed oBiometrics software protection scheme that will secure the MORE-BAILS (cp. section 3.2) process execution.

The MORE-BAILS application will be obfuscated (i.e. protected) by a client specific oBiometrics key. This ensures that only the genuine client can execute MORE-BAILS on his/her phone correctly and thus, reduces the risk that an attacker can illegitimately use the MORE-BAILS application.

5.1 oBiometrics: Biometric-based Software Obfuscation

oBiometrics combines obfuscated interpretation (cp. section 4.3.1.3) with biometric generated keys (cp. section 2.2.2.4) to tightly bind the correct application execution to the genuine client. This stops an attacker from illegitimately using an application, even if the attacker has full access and control over the phone (white box attack model, cp. section 4.3.1).

In remote authentication, oBiometrics assures the authenticator that:

1) No attacker has tampered with the authentication application on the client's phone.
2) Authentication data collection and authentication process execution on the phone was performed correctly.

5.1.1 *Concept of the oBiometrics Scheme*

The three steps to protect an application with oBiometrics are shown in Figure 24.

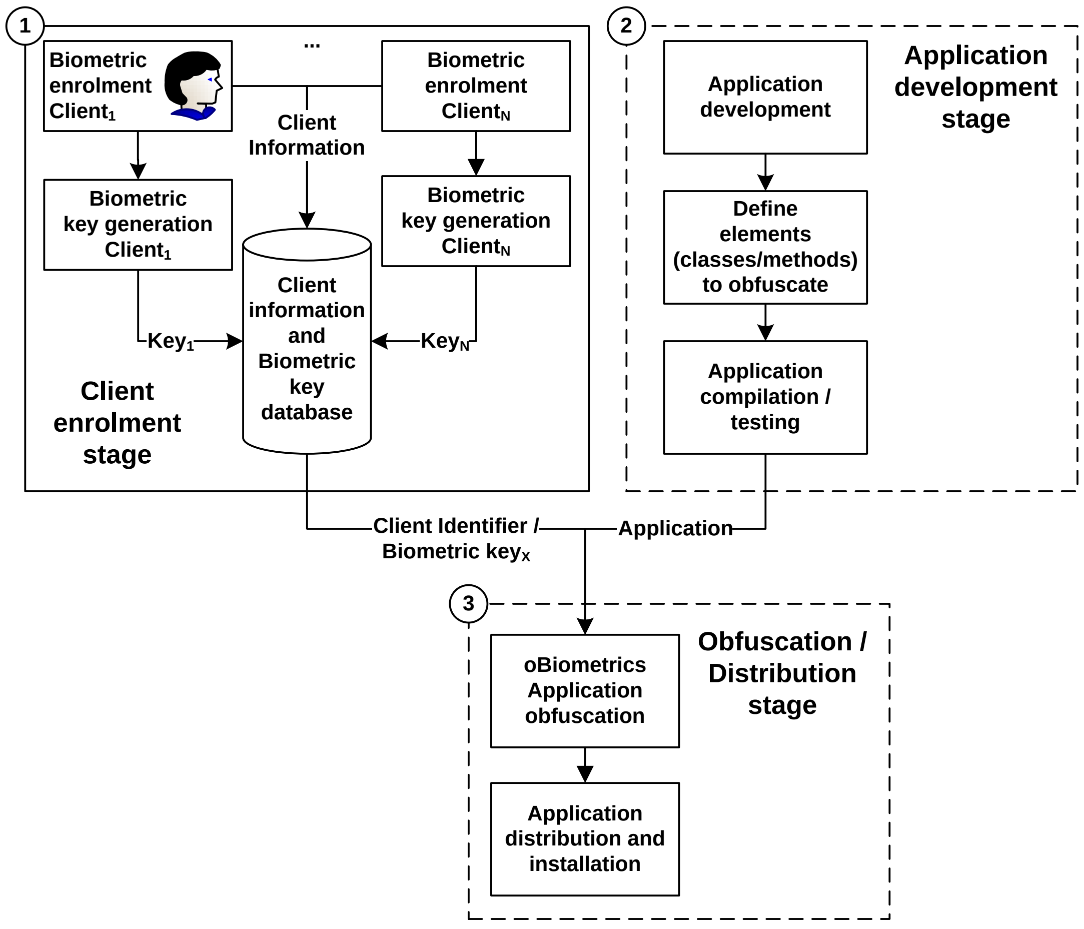

1
Biometric enrolment Client₁
...
Biometric enrolment Client_N
Client Information
Biometric key generation Client₁
Biometric key generation Client_N
Client information and Biometric key database
Key₁
Key_N
Client enrolment stage
2
Application development
Application development stage
Define elements (classes/methods) to obfuscate
Application compilation / testing
Client Identifier / Biometric key_X
Application
3
oBiometrics Application obfuscation
Obfuscation / Distribution stage
Application distribution and installation

Figure 24: Client enrolment, application development and distribution

1) At the client enrolment stage, the authenticator offering the oBiometrics protected application generates a biometrics-based key (cp. section 2.2.2.4) from the freshly captured client's biometrics. This key is then stored together with further personal client information (e.g. client's name and unique identifier) for later use (i.e. application obfuscation) in the authenticator's database. To protect the sensitive biometric data, only cancellable biometric keys (cp. section 0) are used in oBiometrics.
2) In the application development stage, a standard software development cycle (e.g. Java development for Android applications) is executed. The software development is completely independent from the later performed software obfuscation (step 3). Software development does not require specially trained software developers or software developing techniques. Once the application is

developed, elements (e.g. methods, classes, packages) to be obfuscated can be specified. If no elements are specified, then the complete application will be obfuscated. Specifying the elements to be obfuscated (i.e. “partial obfuscation”) can be necessary because:

a) Parts of the application may be used as libraries by other applications, which are not able to handle obfuscated code.
b) Elements are often called during application execution and thus, introduce an unacceptable computational overhead for deobfuscation (cp. section 5.1.3).
c) Partial obfuscation allows protection of nothing but the important parts and algorithms of an application. This speeds-up the interpretation in applications, which do not require complete code protection, because the number of necessary deobfuscation steps decreases.

The developed application is then compiled using standard compilers (e.g. Dalvik Java Cross-Compiler for Android) and tested.

3) In the last step, the application is obfuscated and distributed to the client. The original application instructions are substituted (obfuscated) with instructions selected on the basis of the client's stored biometric key, which makes the obfuscated application uniquely tailored to each client. This client specific, biometric-secured obfuscated application is then distributed to the client for installation on the client’s phone.

The personal client information is used in two ways in oBiometrics:

1) The information links the enrolled client with his/her corresponding biometric key. The authenticator uses this information to re-obfuscate new or updated application versions. Without the client’s biometric key stored in the authenticator’s database, the client has to enrol again for each new application version. This would be expensive and time-consuming for an authenticator offering many applications as well as inconvenient for the client. If the obfuscation is a one-off task, because no application updates will be available, then the biometric keys do not have to be stored. The client’s biometric data can be deleted directly after the obfuscation stage. This is in contrast to other biometric-based authentication systems, which always require the secure storage of the client’s enrolled biometric template to be compared with the freshly

captured biometrics. This storage and comparison of the biometric template is not necessary in oBiometrics, because the obfuscated application "becomes the biometric template". The client does not have to worry about his/her sensitive biometric data. Because it is not stored at any place, it cannot be passed on or be illegitimately used in any way.

2) The information strengthens the oBiometrics obfuscation key (Key$_{oBio}$, cp. section 5.1.2). The Key$_{oBio}$ depends on the biometric generated key as well as the unique client identifiers, i.e. two AFs. This offers more security than depending on only one AF (e.g. biometrics, cp. section 2.2).

Once a client wants to use an application on the phone, the oBiometrics scheme performs the following steps shown in Figure 25.

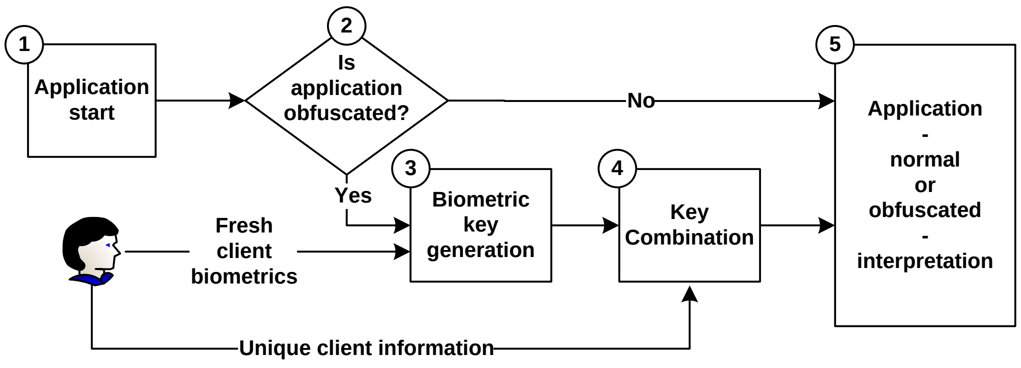

1
Application start
2
Is application obfuscated?
No
Yes
5
Application - normal or obfuscated - interpretation
3
Biometric key generation
4
Key Combination
Fresh client biometrics
Unique client information

Figure 25: oBiometrics application execution on client's phone

1) The client starts the application.
2) oBiometrics checks, if this application is protected and so needs an obfuscated interpretation. The general layout and the instructions of an obfuscated application are not distinguishable from an unprotected application during interpretation. To indicate if the application is obfuscated, a label is inserted during the obfuscation process. If the label is not present in the application, then the application will be normally interpreted.
3) If the label is present in the application, then oBiometrics generates a new biometric key from the freshly captured client biometrics.
4) The biometric key is passed on to the key combination step. This step combines the biometric key with a key generated from the unique client information. Depending on the oBiometrics configuration, the parameters to generate the

second key can be entered directly by the client (knowledge-based AF, e.g. password) or they can be retrieved from a storage medium attached to the phone (object-based AF, e.g. an attached memory card). It is also possible to combine both additional AFs to increase the security even further. In this case, the obfuscation key Key_{oBio} bases upon three AFs. It is not recommended to store the object-based AF directly on the phone. If the phone gets lost or stolen, then the object-based AF is also lost and does not offer any additional security.

5) The key combination step passes the generated key on to the application interpreter, which performs the obfuscated interpretation.

5.1.2 oBiometrics Scheme Implementation

The oBiometrics prototype was implemented and tested on the Android platform (cp. section 4.3.3). In order to enable the Android OS to interpret oBiometrics protected applications correctly, the source code of the DVM was adapted (cp. section 4.3.3.3) to be used for testing and evaluation of the oBiometrics prototype application.

oBiometrics translates the original application byte-code instructions into new instructions depending on the obfuscation key Key_{oBio}. To pass the verification and optimisation steps of the DVM (cp. section 4.3.3.3) successfully, it is not possible to substitute all byte-code instructions during the obfuscation process. For example, the “return-void” instruction cannot be replaced by any other instruction, because this instruction is the only instruction without parameters. If the “return-void” instruction would be substituted with another instruction that expects one or more parameters, the DVM verifier would recognise this error and the application would not be executed any further by the DVM. Similarly, byte-code instructions starting with, for example, “iget” or “invoke*” cannot be obfuscated. These instructions are automatically replaced by the DVM optimiser with other instructions, and thus, are not available for deobfuscation at the DVM interpreter phase. For example, methods referenced by “invoke*” calls are replaced with the actual memory locations of these methods, i.e. the method calls are optimised for static method linking. Table 5 shows exemplarily some of the instructions (third column), which cannot be used for substitution in the obfuscation stage. For clarity, these instructions are divided into groups of similar instructions (columns one and two). The fourth column in Table 5 outlines why the instruction cannot be used for substitution during obfuscation.

Table 5: Instructions not available for oBiometrics obfuscation

Instruction group	Group Description	Associated Instruction Mnemonics	Reason why instruction cannot be substituted
return-void	Return from a method without return value	return-void	Detected by verifier
iget*	Read field instance of kind "*" (e.g. Byte) into register	iget, iget-wide, iget-object, iget-boolean, iget-short, iget-byte, iget-char, iget-short	Replaced by optimiser for static linking
iput*	Store register value into field instance of kind "*" (e.g. Byte)	iput, iput-wide, iput-object, iput-boolean, iput-short, iput-byte, iput-char, iput-short	Replaced by optimiser for static linking
invoke*	Call the specified method	invoke-virtual, invoke-super, invoke-direct, invoke-static	Replaced by optimiser for static linking
invoke*/range	Call the specified method with a range of method parameters	invoke-virtual/range, invoke-super/range, invoke-direct/range, invoke-static/range	Replaced by optimiser for static linking

Another restriction on the available instructions for substitution is that instructions can be only replaced with other instructions, if and only if these instructions have the same general return type as well as the same number and type of parameters. I.e. the instructions must belong to the same instruction group. All other substitutions will not pass the verification phase of the DVM. Table 6 shows exemplarily four of these instruction groups.

Table 6: Selective instruction groups for oBiometrics obfuscation

Instruction Group	Possible byte-code substitutions	Number of instructions in group
3reg-operations	add-int, sub-int, mul-int, div-int, rem-int, and-int, or-int, xor-int, shl-int, shr-int, ushr-int, add-float, sub-float, mul-float, div-float, rem-float, add-long, sub-long, mul-long, div-long, rem-long, and-long, or-long, xor-long, shl-long, shr-long, ushr-long, add-double, sub-double , mul-double, div-double, rem-double, cmpl-float, cmpg-float, cmpl-double, cmpg-double, cmp-long, aget, aget-wide, aget-object, aget-boolean, aget-byte, aget-char, aget-short, aput, aput-wide, aput-object, aput-boolean, aput-byte, aput-char, aput-short	51
2reg-operations	add-int/2addr, sub-int/2addr, mul-int/2addr, div-int/2addr, rem-int/2addr, and-int/2addr, or-int/2addr, xor-int/2addr, shl-int/2addr, shr-int/2addr, ushr-int/2addr, add-float/2addr, sub-float/2addr, mul-float/2addr, div-float/2addr, rem-float/2addr, neg-int, not-int, neg-long, not-long, neg-float, neg-double, int-to-long, int-to-float, int-to-double, long-to-int, long-to-float, long-to-double, float-to-int, float-to-long, float-to-double, double-to-int, double-to-long, double-to-float, int-to-byte, int-to-char, int-to-short	37
2reg-comparison	if-eq, if-ne, if-lt, if-ge, if-gt, if-le	6
1reg-comparison	if-eqz, if-nez, if-ltz, if-gez, if-gtz, if-lez	6

The “add-int” instruction in the first “3reg-operations” group can be substituted with the other 50 byte-code instructions of this group. In the last row, the “if-eqz” instruction can be accordingly translated into all other instructions in the “1reg-comparision” group.

One theoretically possible technique to circumvent these limitations is to deobfuscate the application prior to the DVM verification or optimisation phase. However, this would introduce a tremendous security problem, because the DVM caches successfully verified and optimised applications (cp. section 4.3.3.3).

Android applications are written in Java, compiled by Java language compilers and then converted to Dalvik byte-code by the “dx” tool (cp. section 4.3.3.2). In the oBiometrics prototype, Java Annotations are used to define the Java methods and / or Java classes that should be obfuscated (cp. section 5.1.1). To obfuscate the byte-code instructions in the prototype implementation, the Dalvik source code is first de-compiled by the smali tool [133]. Figure 26 shows an example of a Java method (Figure 26(a)) with an “Obfuscate” annotation and two integer parameters, and the resultant Dalvik byte-code (Figure 26(b)) decompiled by smali.

```
@Obfuscate()
int doIt(int a, int b){
  int c = a + b;
  if (c > 0)
    return c;
  else
    return a;
}
```

(a)

```
.method private doIt(II)I
 .annotation Lbuck/Obf;
   add-int v0, p1, p2
   if-gez v0, :cond_1
   move v0, p1
   :cond_1
   return v0
.end method
```

(b)

Figure 26: Java source code (a) and Dalvik byte-code instructions (b)

During obfuscation, the byte-code instructions (e.g. “add-int” in Figure 26(b)) will be substituted based on the client’s specific Key_{oBio}. Because the number of instructions varies from application to application, a substitution key with a fixed length cannot be used. Instead, a PRNG with Key_{oBio} as seed is used to produce a pseudo-random bit stream of arbitrary length. Figure 27 shows the obfuscation process for the first two byte-code instructions (“add-int”, “if-gez”) of Figure 26(b).

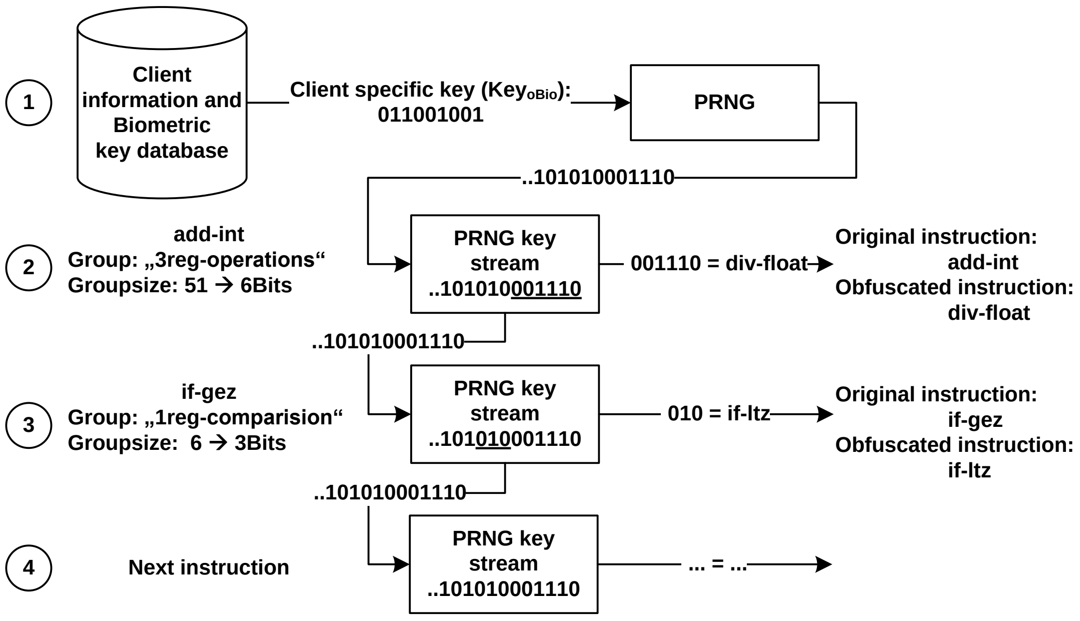

1
Client information and Biometric key database
Client specific key (Key$_{oBio}$): 011001001
PRNG
..101010001110
2
add-int
Group: „3reg-operations“
Groupsize: 51 → 6Bits
PRNG key stream ..101010001110
001110 = div-float
Original instruction: add-int
Obfuscated instruction: div-float
..101010001110
3
if-gez
Group: „1reg-comparision“
Groupsize: 6 → 3Bits
PRNG key stream ..101010001110
010 = if-ltz
Original instruction: if-gez
Obfuscated instruction: if-ltz
..101010001110
4
Next instruction
PRNG key stream ..101010001110
... = ...

Figure 27: Obfuscation process for application byte-code instructions

1) The client specific Key$_{oBio}$ is extracted from the client information and biometric key database and used as a seed to the PRNG.
2) The first instruction to be obfuscated is “add-int” (cp. Figure 26(b)). “add-int” belongs to the “3reg-operations” instruction group (cp. Table 6). The group size of this instruction group “3reg-operations” is 51 elements, i.e. 6 bits are required from the key stream to select the corresponding obfuscated instruction. The first 6 bits from the generated key stream are “001110”. “001110” corresponds to 14 in decimal notation. The “div-float” instruction is at position 14 in the “3reg-operations” group (with counting starting at 0) and will be used as obfuscated instruction to replace the original instruction “add-int”.
3) The next instruction to be obfuscated is “if-gez”. The “1reg-comparision” group of “if-gez” contains 6 elements and thus, requires the next 3 bits (“010”) from the PRNG key stream. The resultant obfuscated instruction is “if-ltz”.
4) This algorithm continues with the following instructions until all application instructions are obfuscated.

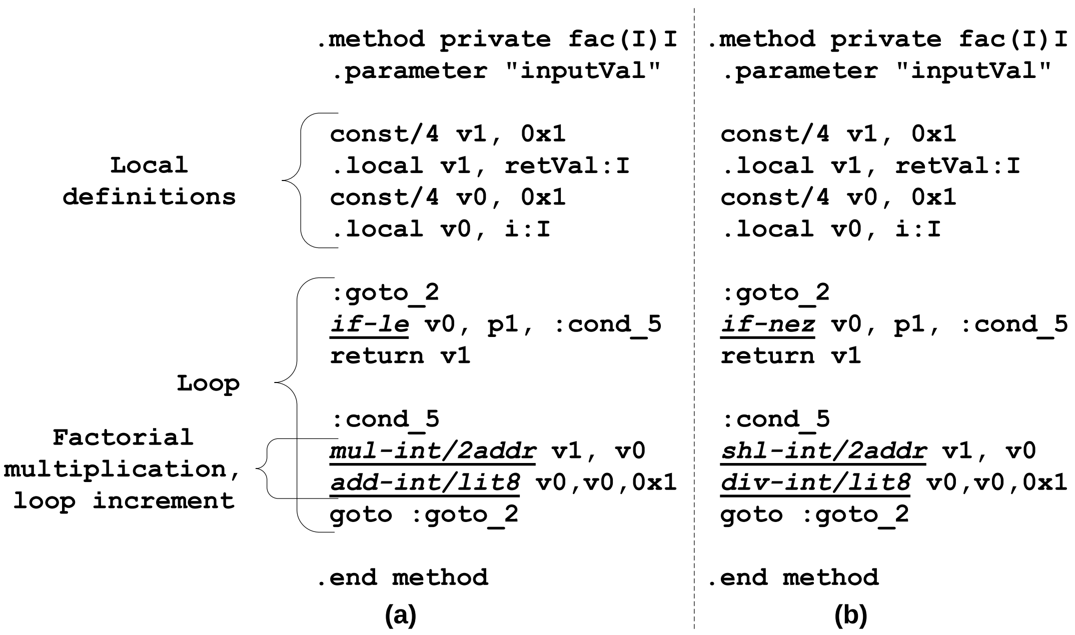
After all application instructions are obfuscated, the obfuscated byte-code is re-assembled by the smali tool and the application is then distributed to the client. Upon receiving the obfuscated application from the authenticator, the client installs the application on the phone. Once the client starts this application, the DVM triggers a process to capture fresh client biometrics. The Key$_{oBio}$ and the pseudo random bit stream are calculated similar to the obfuscation process shown in Figure 27. The resultant PRNG output is then used to deobfuscate the byte-code during interpretation. Based on the PRNG output and the obfuscated instruction, the original instruction is determined by reversing the obfuscation selection rules (cp. Table 6). The deobfuscated byte-code is then executed by the DVM interpreter.

5.1.3 Security / Performance Analysis and Experimental Results

This section analyses the potency, resilience and execution costs (cp. section 4.3.1.2) of the proposed oBiometrics scheme. A single instruction substitution does not create a big challenge for an experienced attacker, who manually analyses the application instructions in a step by step approach. For example, a Java implementation of a "Factorial" function (as shown in Figure 28) results in the byte-code instructions shown in Figure 29.

```java
private int fac(int inputVal) {
  int retVal = 1;
  for (int i = 1; i <= inputVal; ++i) {
    retVal *= i;
  }
  return retVal;
}
```

Figure 28: Factorial function in Java



(a)

```
.method private fac(I)I
 .parameter "inputVal"

 const/4 v1, 0x1
 .local v1, retVal:I
 const/4 v0, 0x1
 .local v0, i:I

 :goto_2
 if-le v0, p1, :cond_5
 return v1

 :cond_5
 mul-int/2addr v1, v0
 add-int/lit8 v0,v0,0x1
 goto :goto_2

.end method
```

(b)

```
.method private fac(I)I
 .parameter "inputVal"

 const/4 v1, 0x1
 .local v1, retVal:I
 const/4 v0, 0x1
 .local v0, i:I

 :goto_2
 if-nez v0, p1, :cond_5
 return v1

 :cond_5
 shl-int/2addr v1, v0
 div-int/lit8 v0,v0,0x1
 goto :goto_2

.end method
```

Figure 29: Byte-code of factorial function: (a) original, (b) obfuscated

With the knowledge of the method and variable names, the individual elements (e.g. variable definitions, loop implementation etc.) of the method can be simply identified. If the attacker faces an obfuscated version as shown in Figure 29(b), the attacker could recognise the substituted instructions and replace them back to the original ones, assuming that the attacker knows the correct implementation of the factorial algorithm. However, if obfuscated interpretation is combined with further obfuscation techniques, then the analysing and deobfuscation task becomes much more complicated even for an experienced attacker. For the example shown in Figure 29, renaming of variables and methods [117] removes the information required to identify the real semantics of the obfuscated byte-code. This increases the potency of oBiometrics and makes a manually performed deobfuscation attempt more complicated, because the byte-code instructions can describe any algorithm.

Potency of oBiometrics also increases with an increasing number of byte-code instructions that can be used for obfuscation, because a manual inspection becomes more difficult and time consuming. Table 7 shows exemplarily the total number of byte-code instructions of six Android applications and how many of these instructions can be theoretically used for obfuscation.

Table 7: Number of instructions available for oBiometrics obfuscation

Application name	Application origin	Total number of instructions	Number of instructions available for obfuscation	Percentage of instruction available for obfuscation
Browser	Android system application	23000	14400	63%
Contacts	Android system application	33800	22100	65%
E-Mail	Android system application	99600	67700	68%
Phone	Android system application	42200	25100	59%
PayPal	Market application: finance	60000	38600	64%
FXCM Mobile	Market application: finance	34300	20900	61%
Average		**48817**	**31467**	**64%**

The first four applications (Browser, Contacts, E-Mail, and Phone) are Android system applications available on all Android phones. The remaining two applications (PayPal and FXCM Mobile TSII (MarketSimplified Inc)) are two of the "top-free in Finance" applications from the Android market. The total number of byte-code instructions varies between 23000 (Browser) and 100000 (E-Mail). Between 14000 and 67000 of these instructions can be theoretically obfuscated. All remaining instructions cannot be used for obfuscation, because substitution of these instructions would be identified during the DVM verification or optimisation phase (cp. section 4.3.3.3). On average, over 60% of the total number of instructions can be used for obfuscation. This large number of instructions makes an attack, in which an attacker analyses the obfuscated byte-code manually step by step, infeasible or at least very time consuming and inefficient.

The large number of available instructions for obfuscation also increases oBiometrics resilience. The total number of possible substitution combinations (T) for an application can be calculated by formula (4):

$$\mathrm{T} = \prod_{i=1}^{N} S^{E} \tag{4}$$

Where:	N : Number of instruction groups.
	S : Size of instruction group i.
	E : Number of elements of group i in the byte-code.

A method containing ten instructions (five instructions from the “3reg-operations”, two instructions from the “2reg-comparision” and three instructions from the “2reg-operations” group, cp. section 5.1.2) results in:

$$T = 51^{5} * 6^{2} * 37^{3} = 6.29*10^{14}$$

possible combinations. Because Android applications have several thousand byte-code instructions (cp. Table 7), it is impossible to try all possible combinations. Even in a sophisticated, brute-force attack with hundreds of computers, such an attack requires years to finish. Furthermore, the obfuscated application and all possible deobfuscated versions are still valid applications (in terms of byte-code verification). This makes a brute-force attack more difficult, because the attacker cannot be sure, if the deobfuscated version is the correct version, i.e. has also the correct semantics. In contrast, it is likely that the application will produce a wrong output or crash at some point during execution.

Resilience of oBiometrics was also tested with de-compilation and reverse-engineering tools for Java and Dalvik byte-code, e.g. “undx” or “Dex2Jar”. In 20%

of the performed tests, these tools were not able to restore any useful application source code from the obfuscated byte-code, because they failed to analyse the changed byte-code instruction structure. In the remaining 80%, the tools returned some de-compiled source code. However, none of the returned source code corresponded to the original semantics of the obfuscated application. I.e. the tools failed in 100% of the tests to restore the real application semantics, because de-compilation tools are not able to understand and verify the semantics of a software application. Instead, these tools simply transfer the obfuscated byte-code back into its corresponding Java form, without being able to judge on the semantic correctness. A human is required to analyse and verify the correctness, which is difficult because of the large number of possible deobfuscated versions.

Analysis of the development and execution costs of oBiometrics is divided into two aspects:

1) Time and resources needed by the application provider to generate an oBiometrics protected application.
2) Time and resources needed to execute the protected application on the client's phone.

Protecting applications with oBiometrics follows the standard software development cycle (cp. section 5.1.1). No additional time or special resources are needed during the application development. The optional specification of the application elements to be obfuscated also does not introduce any costs or time delays in the application development, because it only requires adding Java annotations to the application. The most costly tasks (in terms of required financial resources and time) are client enrolment, biometric key generation and byte-code obfuscation. However, as these tasks are performed only once and they can be performed automatically, the introduced costs can be neglected. Trials showed that obfuscation of thousands of byte-code instructions can be achieved in 3 to 5 seconds depending on the overall program size and structure.

On the client's phone, the required byte-code deobfuscation process adds a further step to the application interpretation inside the DVM (cp. Figure 25), which increases the overall application execution time. However, because oBiometrics does not require that the substitution rules are hidden (cp. section 4.3.2); a substitution table implementation based on a standard vector can be used. This structure has a time complexity of O(1) to access an arbitrary element (i.e. the corresponding deobfuscated instruction) in the vector. I.e. finding and replacing an obfuscated byte-code instruction with the original one is achieved in constant time inside the DVM interpreter. This re-translation must be performed for all obfuscated instructions “n” of the application, i.e. the overall time complexity of oBiometrics is O(n).

oBiometrics also does not add noteworthy memory requirements to the DVM. Additionally memory is only needed to store the substitution table. The DVM supports up to 256 byte-codes, with currently 218 instructions in use. Each of these byte-codes is represented by one byte, thus storing the entire substitution table requires less than one kilo-byte memory.

5.1.4 Conclusion on oBiometrics

oBiometrics offers a lightweight, yet strong software protection mechanism for phones to prevent the illegitimate execution of applications as well as to protect the software against reverse-engineering. Trails on the developed Android prototype showed that the actual run-time overhead introduced by oBiometrics obfuscated interpretation is not noticeable on the client’s phone. The time required to initialise the DVM, to load the apk-file and all associated files (images, layout- and preference-files, etc.), and finally execute the byte-code instructions takes much longer than the time needed for instruction substitutions added by oBiometrics. For time critical applications, partial obfuscation can be used to adjust the introduced run-time overhead for deobfuscation of the obfuscated byte-code instructions. Also, the memory requirement of oBiometrics is insignificant compared to the memory required to execute an application.

The software implemented obfuscated interpretation enhances the flexibility of the application development and supports application updates. New application versions can be instantly obfuscated by the authenticator using the stored client’s obfuscation key Key_{oBio}. Distribution of these application updates is easy, because they can be sent to the enrolled client’s by email for example. Application installation on the client’s phone is also simple, because the application installation routine offered by the phone can be used.

Combination of obfuscated interpretation with biometric-based keys tightly binds the correct application execution to the genuine client. oBiometrics was particularly developed to protect and ensure the correct process execution of mCommerce applications requiring strong client authentication. This shall assure the remote authenticator that the authentication application was correctly executed and that the authentication data was collected and processed accurately on the phone. However, oBiometrics is not limited to this application area. Instead, oBiometrics can be used to protect all kinds of applications on phones. In addition, oBiometrics can also be utilised to protect software applications running on PCs.

Security strength of oBiometrics bases upon the large key space of the client specific biometric-based Key_{oBio} and the large number of instructions available for substitution. In combination with further obfuscation techniques (e.g. layout obfuscation techniques like renaming, cp. section 4.3.1.2), it becomes difficult for an attacker to understand the real semantics of the protected application even if the attacker performs an in-depth application analyses manually, i.e. oBiometrics offers a high potency.

Reverse-engineering tools also fail in the attempt to restore the original semantics of oBiometrics protected applications, because these tools simple re-translate the obfuscated instructions into higher level application statements without considering the application's semantics, i.e. oBiometrics offers a high resilience. Because Key_{oBio} is always freshly generated from the client's provided biometrics, neither the permutation rules nor the obfuscation key must be hidden inside the VM. This simplifies implementation of the obfuscated interpretation framework on phones and eliminates the risk that an attacker can circumvent the protection by identifying the deobfuscation rules or the used key in the application code (cp. section 4.3.2).

5.2 LocAuth: Location-obfuscated Authentication Challenge

MORE-BAILS achieves strong multi-factor client authentication on phones (cp. chapter 3), which correct execution is ensured by oBiometrics (cp. section 5.1). LocAuth was designed to be seamlessly combined with MORE-BAILS and oBiometrics. LocAuth assures the real-time and one-time property of the MORE-

BAILS authentication attempt to the authenticator and thus, prevents distance attacks, e.g. replay or impersonation attacks (cp. section 2.3.1) [134].

LocAuth uses the oBiometrics obfuscated interpretation framework in combination with random nonce(s) and location-based keys to generate a location-obfuscated, real-time authentication-challenge program (called LocProg). Figure 30 shows how an mCommerce application executed on the client's phone can use MORE-BAILS, oBiometrics and LocAuth to achieve robust and secure client authentication.

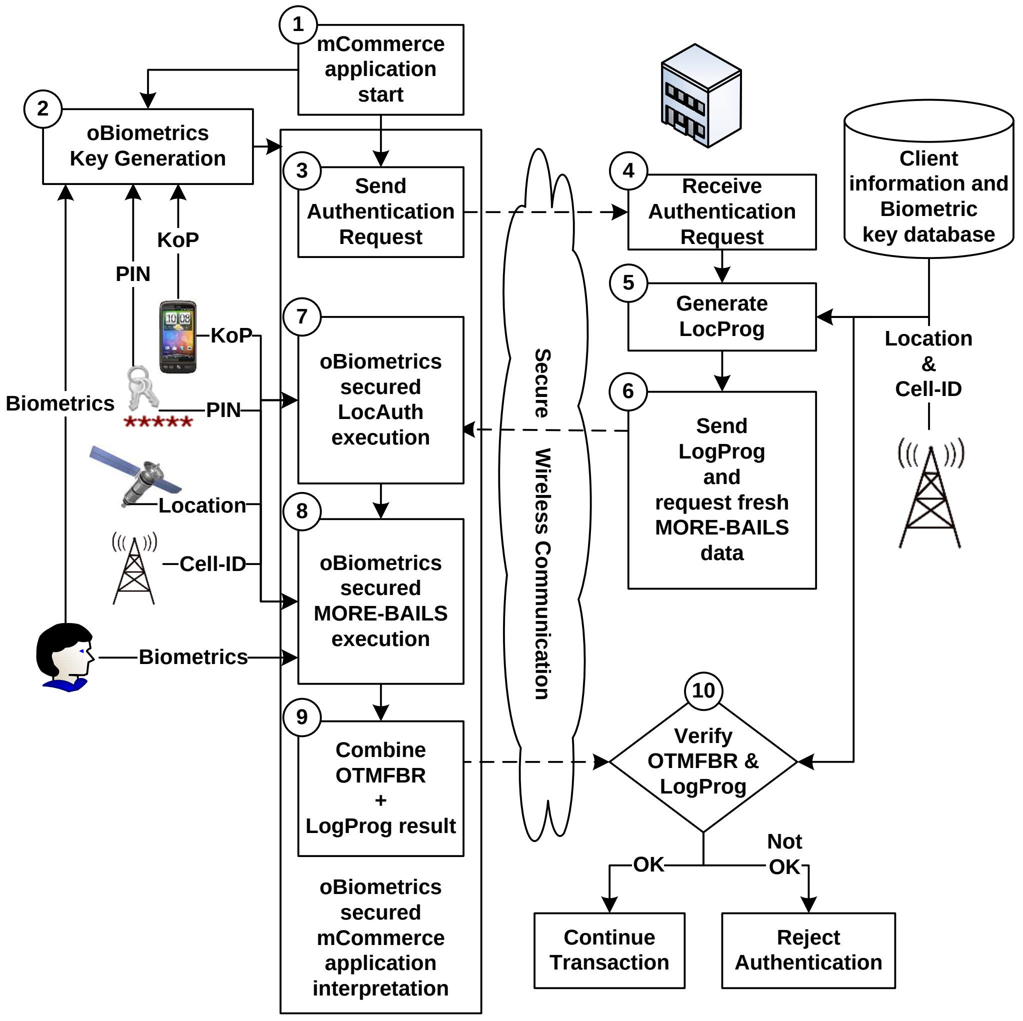

1 mCommerce application start
2 oBiometrics Key Generation
3 Send Authentication Request
4 Receive Authentication Request
5 Generate LocProg
6 Send LogProg and request fresh MORE-BAILS data
7 oBiometrics secured LocAuth execution
8 oBiometrics secured MORE-BAILS execution
9 Combine OTMFBR + LogProg result
10 Verify OTMFBR & LogProg
Client information and Biometric key database
Location & Cell-ID
KoP
PIN
Biometrics
Location
Cell-ID
Secure Wireless Communication
oBiometrics secured mCommerce application interpretation
OK
Not OK
Continue Transaction
Reject Authentication

Figure 30: LocAuth combined with oBiometrics and MORE-BAILS

1) The authentication attempt starts on the client side once the client opens the MORE-BAILS protected mCommerce application on his/her phone.
2) After the mCommerce application has started, a Key$_{\text{oBio}}$ is generated from the freshly taken client's biometrics and the provided personal client data to deobfuscate the oBiometrics secured mCommerce application.

3) If the Key$_{oBio}$ is correct, then oBiometrics interpretation is successful (dotted rectangle in Figure 30) and an authentication request is sent to the authenticator using a secure wireless communication channel.
4) Upon receiving this authentication request, the authenticator requests the client’s current phone location and the cell-ID from the MNO (cp. section 2.2.3.1).
5) A fresh LocProg is then generated and location-obfuscated by a key (Key-Loc$_{Obf}$, cp. section 5.2.1.2) based on the client’s current location and the cell-ID.
6) The generated LocProg together with a request to provide fresh MORE-BAILS data is then sent from the authenticator to the client.
7) The mCommerce application on the client's phone receives the fresh LocProg and dynamically integrates LocProg into the oBiometrics secured LocAuth execution. LocAuth determines the phone’s current location by using the onboard GPS receiver. Simultaneously, the currently used cell-ID is identified as well as PIN and KoP are retrieved. Then, the key to deobfuscate the received LocProg is calculated (Key-Loc$_{DeObf}$, cp. section 5.2.1.2). If the correct Key-Loc$_{DeObf}$ is used, then the client is able to calculate the correct LocProg result.
8) MORE-BAILS is executed and the OTMFBR is generated (cp. section 3.2).
9) The calculated LocProg result is combined with the generated OTMFBR and sent to the authenticator.
10) If verification of the LocProg result and the OTMFBR is successful, then the authenticator continues the transaction, otherwise the client’s authentication attempt is rejected.

LocProg uses random elements such as nonce(s), dynamic changing values known independently to client and authenticator (e.g. cell-ID), personal client data enrolled with the authenticator (PIN, KoP) and location as AFs. These AFs are used in the following way to enhance LocProg security:

1) The random elements are unique to each generated LocProg and thus, they are used to prevent replay attacks as well as to ensure the one-time and real-time (i.e. the response will be accepted by the authenticator within a short time span, e.g. 5 seconds) property of the LocProg challenge.
2) The dynamically changing values (cell-ID), together with the enrolled client data (PIN, KoP), guarantee that only the genuine client with his/her phone can calculate the correct LocProg response, i.e. to prevent impersonation attacks.

3) The KoP acts as a “non-visible” AF. Integration of KoP into LocProg eliminates the risk that an attacker sitting next to the client in the public is able to steal (i.e. to see) all LocProg AFs. Because the KoP will never be exposed to the phone screen, an attacker is not able to get hold of the KoP. The other AFs (PIN, location) can be theoretically “seen” by an attacker. Either by seeing the client entering his/her PIN or by knowing the client’s current location.
4) The independently verified location eliminates that an attacker can pretend to be at a certain place without actually being there, i.e. the client’s actual phone location is assured to the authenticator (cp. section 2.2.3.3).
5) PIN, KoP and cell-ID are used to increase the key-space of the location-based Key-Loc$_{Obf}$ (cp. section 2.2.3.3) and thus, increase the number of different LocProg challenges.

5.2.1 *Implementation and Experimental Results*

The feasibility of LocAuth was tested on the Android platform (cp. section 4.3.3). Table 8 shows some of the collected location measurement for illustration and further explanation of the LocAuth implementation. Entries 1-6 in Table 8 reflect real location measurements. Entries 7-8 (shaded) are intentionally added wrong control locations to reflect an attack on LocAuth. These control entries should be identified and rejected by the authenticator, because the attacker does not use the real location of the client’s phone.

Table 8: Location measurements for LocAuth experiments

	GPS-based Location (L_C)		**MNO-based Location (L_A)**			**Distance between L_C and L_A, metre (L_E)**	**Cell-ID**
	Lat L_C (deg)	**Lon L_C (deg)**	**Lat L_A (deg)**	**Lon L_A (deg)**	**PV (meter)**		
1	51.4977	0.0080	51.4970	0.0070	300	104	53209247
2	51.5142	-0.1486	51.5140	-0.1470	200	113	48627332
3	51.5131	-0.1419	51.5120	-0.1390	500	235	48626504
4	51.5127	-0.1468	51.5140	-0.1470	100	145	48627332
5	51.5138	-0.1439	51.5130	-0.1400	400	284	55171098
6	51.5083	-0.1509	51.5090	-0.1510	100	78	48627465
7	51.5020	-0.0013	51.4980	-0.0100	300	749	53416308
8	51.4862	0.1105	51.4900	0.1200	300	782	54732850

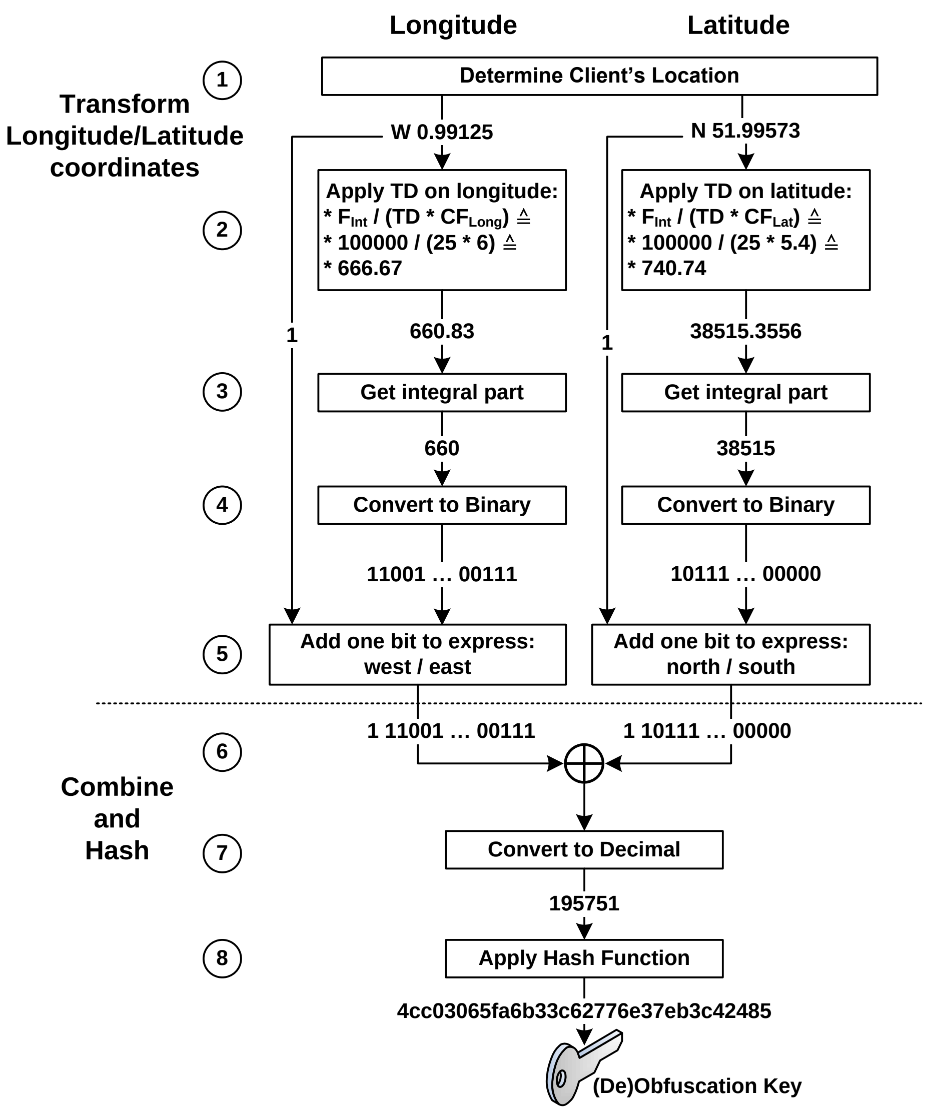
Latitude and longitude values of the client's phone location determined with the onboard GPS receiver are shown in column "L_C" in Table 8. Column "L_A" shows latitude and longitude values determined on the authenticator side via the MNO. Column "L_A" also shows the Proximity Value (PV) provided by MNO to the authenticator. This PV expresses the confidence that the MNO has in its own localisation measurements. I.e. the MNO states that the client's phone is inside a circle around the provided location L_A with a radius equal to PV.

The localisation error (distance between L_C and L_A), resultant from the different accuracy of the two localisation techniques used (cp. section 2.2.3.1), is shown in Column "L_E". This error is between 78 and 284 metres, with an average error of 160 metres for the genuine measurements (i.e. entries 1-6). The location error of the two added control entries is larger (765 metre) and should be identified as attacks. The last column shows the "Cell-ID" currently used in the communication process between client and MNO (cp. section 2.2.3.1).

5.2.1.1 LDEA Location-based Key Generation

Correct interpretation of LocProg on the client's phone requires the corresponding deobfuscation "Key-Loc$_{DeObf}$" to the obfuscation "Key-Loc$_{Obf}$" used by the authenticator.

LOTA analysed, if the LDEA location-based key generation approach [61] (cp. section 2.2.3.3) can be utilised in LocAuth to generate the (de)obfuscation keys. The relevant part of the LDEA location-based key generation process is detailed in Figure 31.



Longitude
Latitude
Transform Longitude/Latitude coordinates
1
Determine Client's Location
W 0.99125
N 51.99573
2
Apply TD on longitude: * F_Int / (TD * CF_Long) ≙ * 100000 / (25 * 6) ≙ * 666.67
Apply TD on latitude: * F_Int / (TD * CF_Lat) ≙ * 100000 / (25 * 5.4) ≙ * 740.74
1
1
660.83
38515.3556
3
Get integral part
Get integral part
660
38515
4
Convert to Binary
Convert to Binary
11001 … 00111
10111 … 00000
5
Add one bit to express: west / east
Add one bit to express: north / south
1 11001 … 00111
1 10111 … 00000
6
Combine and Hash
7
Convert to Decimal
195751
8
Apply Hash Function
4cc03065fa6b33c62776e37eb3c42485
(De)Obfuscation Key

Figure 31: LDEA key generation process

1) The LDEA key generation process starts with the determination of the client's location, i.e. the location in which the client is able to use the deobfuscation key. In this example, the University of Buckingham is used as the client's location with its coordinates West 0.99125, and North 51.99573.
2) The TD is then applied to the determined coordinates. The coordinates are first multiplied with the factor F_{Int} = 100000 to become an Integer and afterwards divided by the tolerate distance TD (i.e. TD = 25 metre for this example).
3) The divisor needs to be corrected by a correction factor (CF_{Long}) of 6 for the longitude respectively a CF_{Lat} of 5.4 for the latitude coordinate for each metre of tolerated distance [61].
4) The integral part of the result is calculated.
5) The integral part is binarised.

6) One bit is added to the binary value to express north / south respectively east / west of the determined coordinates.
7) The two binary values are combined by the XOR-function.
8) The XORed binary value is converted back to a decimal form.
9) The decimal value is hashed (e.g. Secure Hash Algorithm (SHA)) and used as the (de)obfuscation key.

Trials of this LDEA key generation process were conducted by LOTA. Based on the received trial results, LOTA concluded that the LDEA process does not always produce the same key on both sides, even if the receiver is well within the tolerated distance area TD. In the example of Figure 31, the client should be able to calculate a correct deobfuscation key in a circle of 25 metres (i.e. the TD value) around the used location of "W0.99125 / N51.99573". However, if the client is at location "W0.99151 / N51.99573", which is 17 metres away from the intended location, then the integral part of the longitude value is different ("0.99151 * 100000 / (25*6) = 661.0067"). Thus, the generated deobfuscation key is different. This means that the client is not able to calculate the correct deobfuscation key, although the client is well inside the used TD of 25 metres.

The authors of LDEA [61] were contacted by LOTA to discuss this. The LDEA authors confirmed this LDEA aspect and replied that this might be not a problem in a practical scenario, because of the inaccuracy of current GPS receivers. However, this uncertainty of the LDEA method is a knock-out criterion for the use in LocAuth. LocAuth always requires a correct location-based key generation to calculate the correct challenge-response. If the client calculates a wrong key because of the method uncertainty, then a genuine client authentication attempt would be incorrectly rejected by the authenticator.

5.2.1.2 LocAuth Location-based Key Generation

LOTA concluded that the LDEA location-based key generation cannot be used for LocAuth, because of the methods uncertainty to always calculate the correct key (cp. section 5.2.1.1). Instead, LOTA proposes to use a location grid with size S and a unique key (K_1 to K_n) associated with every grid intersection to calculate the required location-based keys independently by client and authenticator as shown in Figure 32.

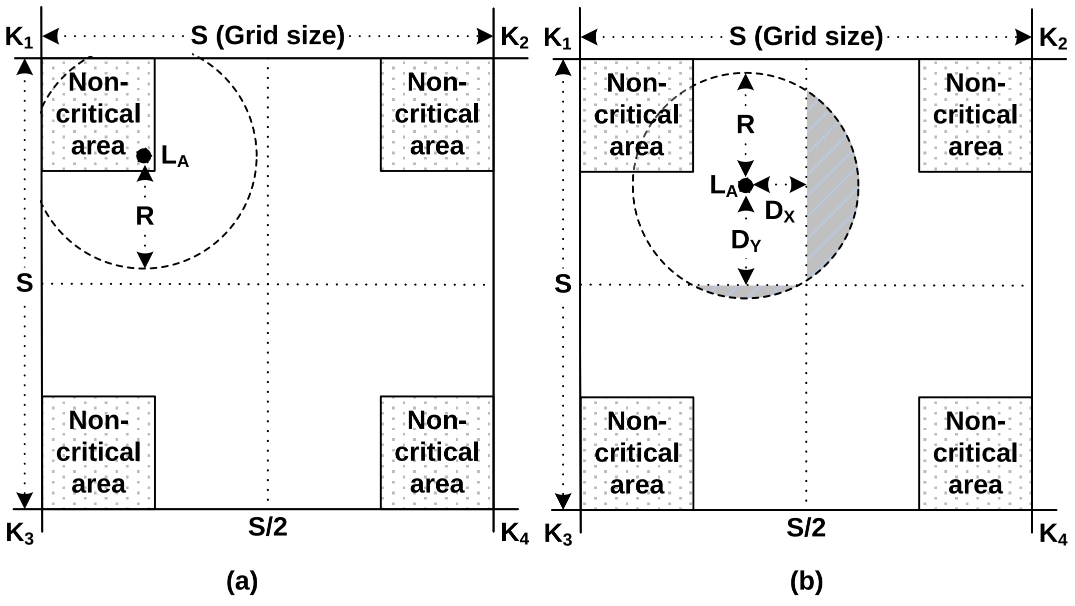


Figure 32: Location grid for key generation

L_A shows the client's phone location determined by the MNO. The authenticator uses the key associated with the nearest grid junction to L_A (i.e. K_1 in Figure 32) to obfuscate LocProg. The client uses the phone's onboard GPS receiver to determine his/her current location (L_C) and the same location grid to calculate the corresponding LocProg deobfuscation key (i.e. the key associated with the nearest grid junction to the GPS-based L_C). Because of the accuracy differences of the two localisation techniques used (cp. section 2.2.3.1), L_A will never be exactly the same as L_C. Therefore, the authenticator will tolerate a certain localisation error and accept an authentication attempt, if the client's location is inside this tolerance area. The dotted circle around L_A with radius R describes this tolerance area. Two cases must be distinguished for the LocProg key calculation:

1) L_A is near a grid intersection (i.e. inside the non-critical area as shown in Figure 32(a)). In this case, the complete tolerance circle R is within one quadrant (upper-right in the example of Figure 32(a)) of the grid. Thus, L_A and L_C will always result in the same key (nearest grid junction), if L_C is inside the tolerance area.
2) L_A is outside the non-critical area as shown in Figure 32(b). In this case, the client might use a different, wrong key (e.g. K_3 or K_4), because L_C is inside the gray shaded areas of the tolerance area and thus, is closer to another grid intersection.

To deal with the problem of wrong key selection in LocAuth, the authenticator will accept LocProg results based on more than one key, if the probability $P_{D_{\{X,Y\}}}$ (calculated by Formula (5)) that the client uses the correct key, is smaller than a pre-defined threshold T. $D_{\{X,Y\}}$ expresses the distance between L_A and the middle of the grid in the vertical (D_X), respectively horizontal (D_Y) direction as shown in Figure 32(b).

$$P_{D_i} = \begin{cases} 1 & \text{if location is in non-critical area} \\ \dfrac{R + D_i}{2R} & \text{, } i \in \{X,Y\} \text{ otherwise} \end{cases} \quad (5)$$

A probability threshold of T=0.95, a grid size of S=500 metres, and a default tolerance distance of R=300 metres is used in the experiments. The 300 metres are chosen to minimise false rejection of genuine clients due to the accuracy difference of the used localisation techniques (cp. section 2.2.3.1) and are supported by the FCC E-911 directive [68].

If the MNO returns a high confidence in the measurement (i.e. the PV is smaller than the default tolerance distance of 300 metres), then the PV is used as the tolerance distance R as shown in Formula (6).

$$R = \begin{cases} \text{proximity} & \text{if proximity} < \text{default tolerance} \\ \text{default tolerance} & \text{otherwise} \end{cases} \quad (6)$$

Calculation of the LocProg (de)obfuscation keys is shown in Figure 33.

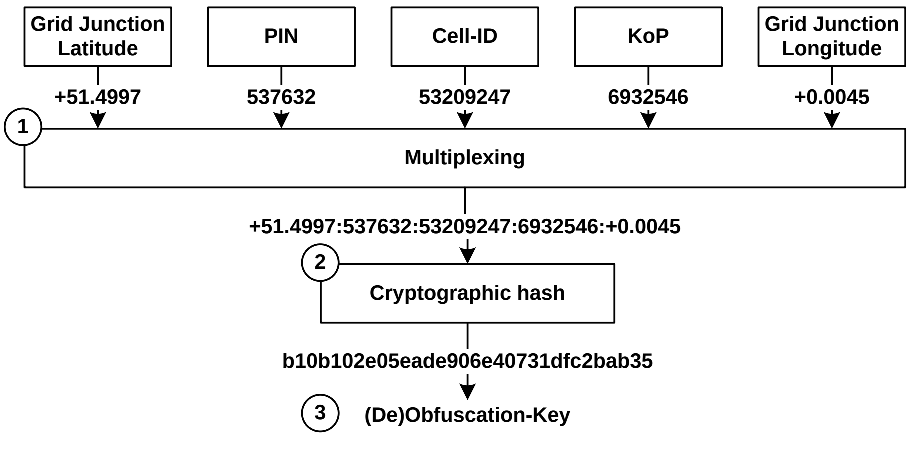


Figure 33: Key generation process for location measurements

1) Latitude and longitude values of the nearest grid intersection are combined (e.g. multiplexed) with PIN, KoP and cell-ID.
2) The combination is cryptographic hashed (e.g. SHA).
3) The resultant key is then used to (de)obfuscate LocProg.

The cell-ID acts as a dynamic changing value and is used together with PIN and KoP to increase the value range of the hash-function input. This increases the LocProg key-space and makes it more difficult for an attacker to guess the correct LocProg result. Table 9 shows the calculated client (Key_C) respectively authenticator keys (Key_{A1} to Key_{A4}) for the eight location measurements of Table 8.

Table 9: Client and authenticator keys

	P_X	P_Y	Key_C	Key_{A1}	Key_{A2}	Key_{A3}	Key_{A4}	Key ok?
1	0.89	0.57	17675 ..	8170a ..	d1cc0 ..	17675 ..	da280 ..	True
2	0.69	0.89	fa008 ..	f66ad ..	fa008 ..	d1ebb ..	b7781 ..	True
3	0.72	0.71	e8a75 ..	44595 ..	e8a75 ..	65a63 ..	79ccb ..	True
4	0.89	1.00	f66ad ..	f66ad ..	fa008 ..			True
5	0.60	0.89	19baa ..	1375b ..	19baa ..	187a5 ..	ce489 ..	True
6	1.00	1.00	d6030 ..	d6030 ..				True
7	0.60	0.61	e6bdf ..	0cb34 ..	198d8 ..	0e007 ..	9599a ..	False
8	0.60	0.79	37f75 ..	c727a ..	31e87 ..	d9827 ..	b376e ..	False

The following observations can be made from Table 9:

1) In row 1, both probability values P_X (0.89) and P_Y (0.57) are below T, i.e. the client is near the grid centre. The authenticator will accept the results based on all four keys of the grid (Key_{A1} to Key_{A4}). The client actually uses the key corresponding to Key_{A3} (shaded cell) for the response.
2) In row 4, probability P_X is below and probability P_Y is above the defined threshold T. In this case, the authenticator accepts both keys above the horizontal middle line of the grid, i.e. Key_{A1} and Key_{A2}. The client uses in this measurement Key_{A1}.
3) In row 6, the client location is determined by the MNO inside a non-critical area (both probabilities are equal to 1) and only one key is calculated by the authenticator. Because the client uses the same key, the client's response will be considered as genuine.

4) In the last two control rows (7, 8); the response is not accepted by the authenticator because the client is too far away from the claimed location. The client's calculate key does not match any of the authenticator's keys.

The current implementation of LocAuth deals with the problem of wrong key selection by accepting more than one key. An alternative approach to calculate the required (de)obfuscation keys is to expend the grid size S until an acceptable FRR (i.e. the genuine client selects the wrong key even though the client's location is close enough to L_A) is reached. The probability that the client uses the correct key depending on grid size S and the tolerated localisation error R (cp. Figure 32(b)) is calculate by Formula (7) as shown in Figure 34.

$$P(S,R) = \frac{1\left(\frac{S}{2}-R\right)+1\left(\frac{S}{2}-R\right)}{S} + \frac{0.75R+0.75R}{S} = \frac{\frac{S}{2}+\frac{S}{2}-2R}{S} + \frac{1.5R}{S} = \frac{S-2R}{S} + \frac{1.5R}{S} = \frac{S-0.5R}{S} = \frac{S}{S} - \frac{R}{2S} = 1 - \frac{R}{2S} \quad (7)$$

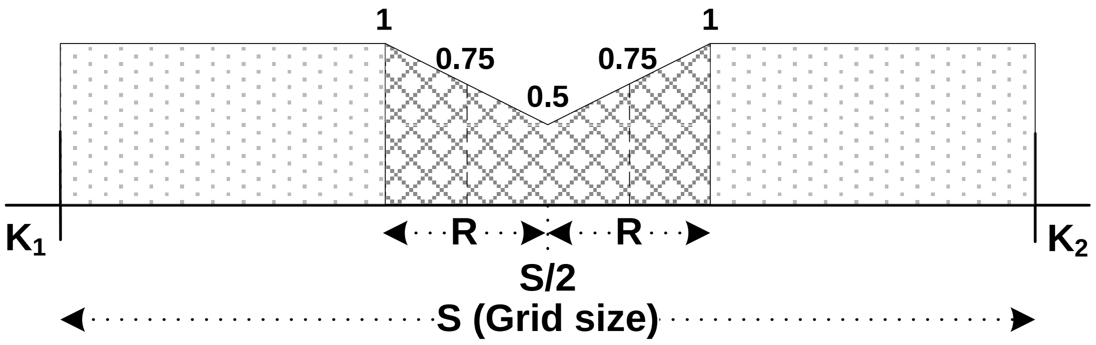


Figure 34: Calculation of key probability

If the client's location is further away than the distance R from the centre of the grid, then the client always selects the correct key, i.e. the probability in these two areas (dotted rectangles in Figure 34) is equal to 1. If the client is within R, then the correct key selection probability varies between 1 (intersection point with dotted rectangles) and 0.5 (centre of the grid, i.e. S/2). On average, the probability for these two (dashed) areas in Figure 34 is 0.75.

Figure 35 shows these probabilities (P-Axis) for grid size values between 500 and 5000 metres (S-Axis) and accepted localisation errors between 200 and 300 metres (R-Axis).

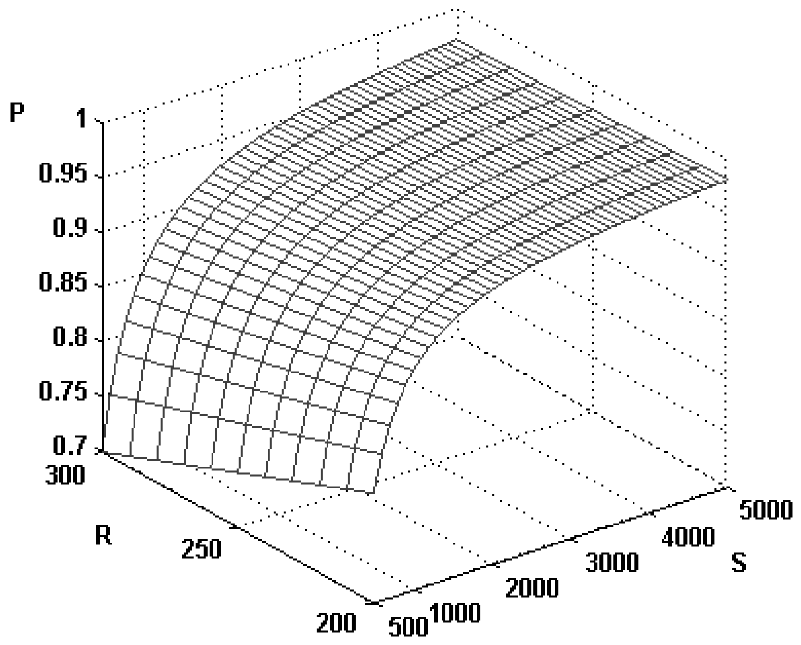


Figure 35: Probability of correct key selection on client side

For example, if an FRR of less than 5% is required in an mCommerce application and the achievable localisation error R is up to 300 metres, then a grid size S=3000 metres is necessary. If the localisation error can be reduced to R=200 metres, a grid size of S=2000 metres is necessary to achieve the same FRR. This results in more available keys, more precise client localisation and therefore higher security. The drawback of this approach is that a genuine client can be falsely rejected and that the required grid size S is larger compared to the approach used in LocProg. A large grid size results in less available grid intersections and thus, less location-based keys. This weakens the key strength and increases the risk that an attacker can correctly guess the key.

Another possible approach for the key selection problem is to add a small indicator to LocProg in the case that the client could select the wrong key. In the case of an ambiguous situation, the indicator can specify the correct key to choose. This approach handles a smaller grid size compared to the other two approaches, because the authenticator needs to calculate and accept only one key. A drawback of this approach is that the key is not completely independently determined on both sides because of the sent indicator. This contradicts the properties of secure and independent location-based key generation as defined by LOTA (cp. section 2.2.3.4).

5.2.1.3 LocAuth Prototype Implementation

The LocAuth prototype was developed and tested on the Android platform (cp. section 4.3.3). LocProg is integrated into LocAuth on the “Applications” layer (cp. section 4.3.3.1) once the client receives the fresh LocProg from the authenticator using the Java code shown in Figure 36.

```
DexClassLoader cl  = new DexClassLoader(Path2RecLocProg, ..);
Class cClass       = cl.loadClass(ChallengeClassName);
Method cMethod     = cClass.getMethod(challengeMethodName, ..);
Response cResponse = cMethod.invoke(cClass.newInstance());
```

Figure 36: Java code to integrate LocProg

One potential security problem of integrating LocProg on-the-fly into LocAuth is that an attacker could try to change LocProg during transmission from the authenticator to the client, i.e. the attacker could perform a code injection attack. To overcome this problem, the LocProg is digitally signed by the authenticator prior to transmission to the client. The LocAuth application verifies the LocProg signature. If the signature is correct, then the received LocProg is integrated into LocAuth, otherwise rejected.

Java annotations as shown in Figure 37 are used to distinguish between oBiometrics byte-code instructions (biometric-obfuscated) and instructions of LocProg (location-obfuscated) inside the DVM.

```
@Obfuscate(type = "biometric")      for oBiometrics
@Obfuscate(type = "location")       for LocAuth
```

Figure 37: Java annotations for oBiometrics and LocAuth

The DVM employs the correct deobfuscation key based on the annotation type during the application interpretation. An example of a LocProg for Android phones based on integer equations is shown in Figure 38.

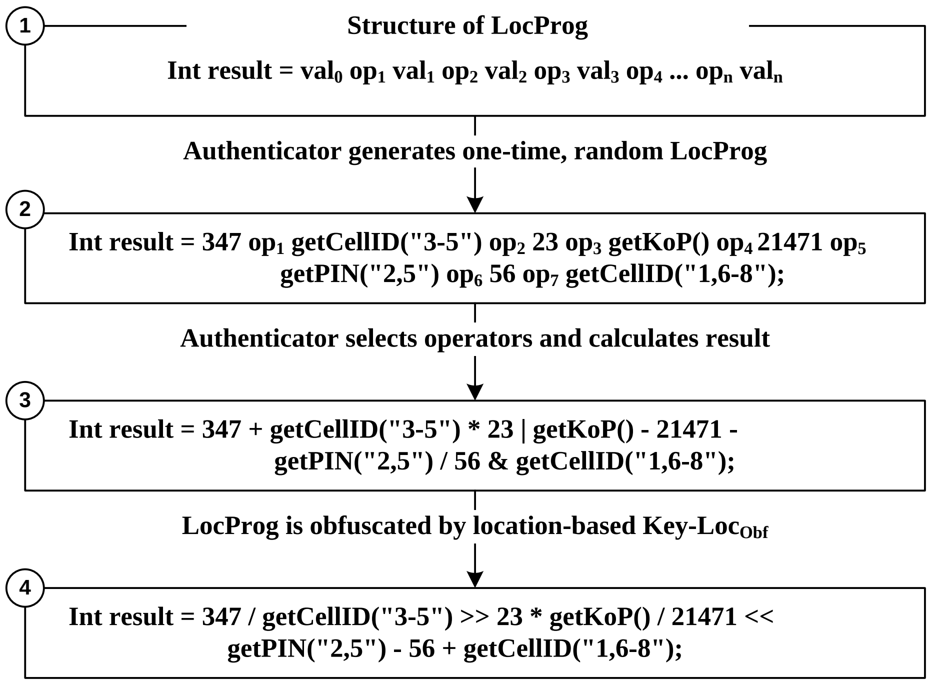

1
Structure of LocProg
Int result = val_0 op_1 val_1 op_2 val_2 op_3 val_3 op_4 ... op_n val_n
Authenticator generates one-time, random LocProg
2
Int result = 347 op_1 getCellID("3-5") op_2 23 op_3 getKoP() op_4 21471 op_5 getPIN("2,5") op_6 56 op_7 getCellID("1,6-8");
Authenticator selects operators and calculates result
3
Int result = 347 + getCellID("3-5") * 23 | getKoP() - 21471 - getPIN("2,5") / 56 & getCellID("1,6-8");
LocProg is obfuscated by location-based Key-Loc_{Obf}
4
Int result = 347 / getCellID("3-5") >> 23 * getKoP() / 21471 << getPIN("2,5") - 56 + getCellID("1,6-8");

Figure 38: Example of LocProg challenge-program for Android phones

1) LocProg contains default integer values (val_0, val_1 etc.) and default operators (op_1, op_2 etc.).
2) During generation of a new LocProg, the authenticator replaces these default values with:
   a) randomly generated numbers,
   b) client specific numbers.

   The randomly generated numbers (i.e. nonce) ensure the one-time property of this LocProg. The client specific numbers ensure that only the genuine client can calculate the correct response on his/her phone. For the example shown in Figure 38, the second and fifth digit of the PIN (getPIN(2, 5)), the KoP token (getKoP()), and the digits 1, 6, 7 and 8 of the cell-ID (getCellID(1, 6-8)) are used to replace the default values.

3) The authenticator replaces the default operators with randomly selected operators (e.g. add, sub) and calculates the result of this generated LocProg. The result is then stored in the authenticator database to be compared with the result received from the client during LocAuth verification.
4) The authenticator obfuscates the selected operators, i.e. the operators are substituted with other operators based on Key-Loc_{Obf}. The resultant obfuscated LocProg is then sent to the client.

Operator substitutions during obfuscation must be within the same byte-code instruction group (e.g. 3reg-operations, cp. Table 6) to pass the verification phase of the DVM (cp. section 5.1.2). Because LocProg bases upon integer arithmetic, some instructions (e.g. aget, aput) from Table 6 are not used within LocProg as these instructions do not represent an arithmetic operation. Another restriction on the possible operator substitutions occurs from the fact that LocProg should not "crash" during obfuscated interpretation because of an invalid instruction (cp. section 5.2.2). To circumvent that LocProg crashes during interpretation, the "3reg-operations" is further divided into two subgroups: "3reg-operations-single" and "3reg-operations-double" instructions as shown in Table 10. Within these defined subgroups, the instructions for substitution can be selected as described in section 5.1.2.

Table 10: Instruction groups for LocProg obfuscation

Instruction Group	**Possible byte-code substitutions**	**Number of instructions in the group**
3reg-operations-single	add-int, sub-int, mul-int, div-int, rem-int, and-int, or-int, xor-int, shl-int, shr-int, ushr-int, add-float, sub-float, mul-float, div-float, rem-float, cmpl-float, cmpg-float	18
3reg-operations-double	add-long, sub-long, mul-long, div-long, rem-long, and-long, or-long, xor-long, shl-long, shr-long, ushr-long, add-double, sub-double , mul-double, div-double, rem-double, cmpl-double, cmpg-double, cmp-long	19

The LocProg prototype uses Integer arithmetic, i.e. instruction from the "3reg-operations-single" group shown in Table 10. The possible instruction combinations of a LocProg using this "3reg-operations-single" group are 18^n. 18 is the total number of operators in the "3reg-operations-single" group and "n" corresponds to the number of operations in LocProg.

For example, a LocProg as shown in Figure 38 containing seven operations leads to $18^7 = 6.12*10^8$ possible combinations for the operator obfuscation.

The byte-code for some parts of an obfuscated LocProg is shown in Figure 39.

```
    .method private LocProg-challenge()I
(1)   .annotation build Luk/ac/buckingham/Obfuscate; type="location"
      const-string v1, "3-5"
(3)   invoke-direct {p0, v1}, Luk/ac/buckingham/LocObf;
(2)                              ->getCellID(Ljava/lang/String;)I

      move-result v1
(4)   div-int/2addr v0, v1
      …
(3)   invoke-direct {p0, v1}, Luk/ac/buckingham/LocObf;
(2)                              ->getPIN(Ljava/lang/String;)I
      ...
(4)   shl-int/2addr v0, v1
(5)   return v0
    .end method
```

Figure 39: Byte-code of obfuscated LocProg

1) The LocProg-challenge() method in Figure 39 begins with an obfuscate-annotation of type "location", which is required by the DVM interpreter to select the correct deobfuscation key.
2) Cell-ID and PIN are then requested from the Android system ("getCellID()") respectively client ("getPIN()") via direct function calls.
3) Determined cell-ID and entered PIN are integrated as integers into the DVM interpretation flow (invoke-direct with return type integer "I").
4) The result of this LocProg is calculated by executing the arithmetic operations.
5) The result of the deobfuscated integer calculation is returned by the LocProg-challenge() method ("return v0"). This result is then used as the LocAuth challenge-response and sent from the client's phone to the authenticator for verification.

5.2.2 Security and Performance Analysis

LocAuth bases upon six AFs, which are used to generate and obfuscate a LocProg:

1) Client specific PIN to ensure that only the genuine client can generate the correct response, i.e. the PIN acts as a "something you know" AF.
2) Phone specific KoP to ensure that the genuine client's phone must be used i.e. the phone acts as a "something you have" AF.
3) Biometrics due to the tight combination of LocAuth with oBiometrics, i.e. execution of LocAuth is oBiometrics protected. Biometrics ensures that only the genuine client can execute LocAuth, i.e. the biometrics act as a "something you are" AF.

4) Time due to the integration of randomly generated nonces and the time limit within the client has to respond to the LocAuth challenge. Time ensures the one-time and real-time property of the authentication attempt and thus, eliminates replay attacks.
5) Current cell-ID used in the communication between phone and cellular network. The Cell-ID acts as a dynamically changing secret that is known only to the authenticator (via MNO request) and the client.
6) Current location of the client’s phone to ensure that the client is where s/he claims to be. Location eliminates distance attacks, because an attacker is not able to pretend to be at a different location.

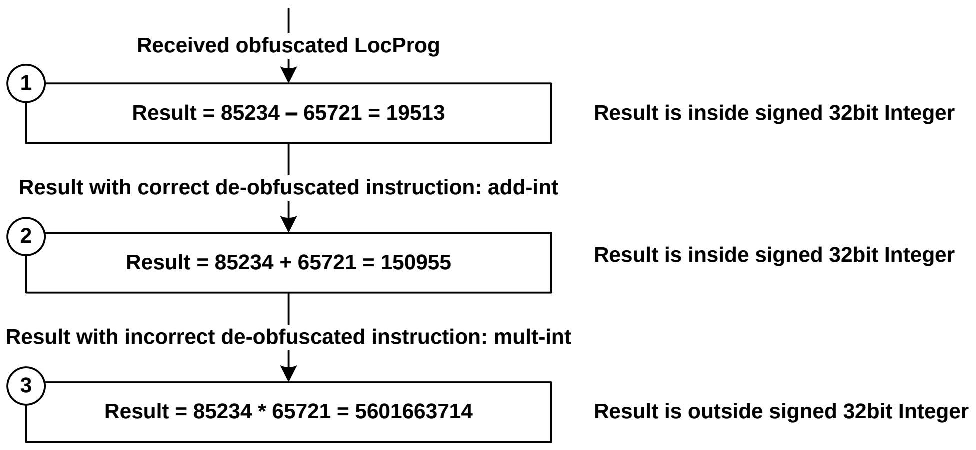
In order to calculate the correct LocProg result, all six AFs must be correctly used. If one AF is incorrect, then the wrong LocProg result will be calculated, which will be identified by the authenticator during verification. I.e. an attacker needs to breach all six AFs simultaneously to get authenticated.

PIN, KoP, time and cell-ID are used to generate a unique, one-time and client specific LocProg. The client’s current location is then used to obfuscate this generated LocProg. Because location is not a completely secret AF and thus, cannot be considered as 100% secure on its own (cp. section 2.2.3.4); PIN, KoP and cell-ID are also integrated into the location-based obfuscation key. This integration serves two purposes:

1) PIN, KoP and cell-ID are more secret AFs than location and thus, it becomes more difficult for an attacker to determine all these AFs at the same time to successfully deobfuscate LocProg.
2) PIN, KoP, and cell-ID increase the key-space of the obfuscation key, which makes a brute-force attack more difficult. The number of location-based keys is limited because of the earth surface and the accuracy of the localisation techniques for phones (cp. section 2.2.3). Combination of the client’s location with PIN, KoP and cell-ID eliminates this limitation of location-based keys. The obfuscation key length can be adjusted to the application security needs by changing the length (e.g. a longer KoP) and properties (e.g. integration of special characters beside numbers in the PIN) of KoP and PIN.

LocProg is designed to terminate the interpretation process correctly at all times. If the wrong location-based deobfuscation key is used, then the LocProg interpretation will not "crash", but will instead calculate a wrong result. The "no-crash" property of LocAuth is important to eliminate locally performed "try-and-error" attacks. Without this "no-crash" property, the attacker can conclude on its own (i.e. without sending the result to the authenticator) that the wrong key was used and try other keys until LocProg terminates correctly. This reduces the number of challenge-responses and thus, makes a "guessing and trying" attack to find the correct LocProg result more likely.

The "no-crash" property is implemented in LocProg by using a specially designed, Integer-based arithmetic calculation. Integers are defined in Android systems as signed 32bit values. Integer arithmetic was selected for the LocProg prototype, because these arithmetic operations do not raise an exception once a data-type overflow occurs. Instead, application interpretation continuous normally without an application crash.
A data-type overflow can occur, if an incorrect deobfuscation key is used as shown in Figure 40.



Received obfuscated LocProg
1
Result = 85234 – 65721 = 19513
Result is inside signed 32bit Integer
Result with correct de-obfuscated instruction: add-int
2
Result = 85234 + 65721 = 150955
Result is inside signed 32bit Integer
Result with incorrect de-obfuscated instruction: mult-int
3
Result = 85234 * 65721 = 5601663714
Result is outside signed 32bit Integer

Figure 40: Example of LocProg Integer overflow

1) The received obfuscated LocProg contains a subtraction operation, i.e. a "sub-int" instruction. The calculated result of this LocProg is 19513, which is inside the value range of signed 32bit Integers.

2) If the correct deobfuscation key is used, then the “sub-int” instruction will be replaced with an “add-int” instruction. The result of 150955 is also within the value range of signed 32bit Integers.
3) The “sub-int” instruction is deobfuscated to a “mult-int” instruction, because the wrong deobfuscation key was used. The result would be 5601663714. This value is outside the range of signed 32bit Integers (i.e. 2147483647).

On Android phones, no crash happens in the case of a data-type overflow and the application interpretation continuous normally. In the example of Figure 40, the overflowed value of “5601663714” is converted to “-1306696418”.

The use of signed 32bit Integers in LocProg produces $4.29*10^{9}$ return values. This number of LocProg results is considered by LOTA as sufficient and secure, because an attacker cannot determine locally, if the calculated result is correct. The attacker always needs to send the result for verification to the authenticator. If the authenticator receives a wrong result, the authenticator will immediately reject this authentication attempt. This means that the attack has only one chance to guess the correct result, i.e. the probability to guess the correct result is $2.33*10^{-8}$%.
To increase the number of LocProg results even further and thus, make it more difficult for an attacker to guess the correct result, future LocProg versions could use:

1) Arithmetic equations based on signed 64 bit “Long” values, i.e. $1.84*10^{19}$ return values.
2) Combine the results of more than one equation into an array structure.

Uniform distribution of the LocProg results (i.e. all results should occur with the same probability) is important for the LocAuth security. The results need to be uniformly distributed to avoid that an attacker concentrates on results, which occur with higher probability. This would simplify a brute-force attack or make guessing the correct result easier. LocProg avoids certain combinations of operators and values to achieve a uniform result distribution. For example, LocProg avoids the following operator and value combinations:

1) Several multiplication operations with one value equal to 0. This combination always leads to the value of 0 for the multiplications, regardless of the other values, i.e. the value 0 has a higher probability to occur as a LocProg result.
2) “shift-right” and “shift-left” operations are never followed by a large value. This leads to a value of 0 (“shift-right”), respectively the minimum / maximum Integer (“shift-left”) regardless of the value on the left hand side of the shift-operator.

Execution of LocAuth introduces computational overhead during execution of the obfuscated application. Each obfuscated instruction requires additional computing cycles to be deobfuscated and then interpreted (cp. section 5.1.3). However, the computational overhead introduced by LocAuth is negligible, because the number of obfuscated LocProg instructions (i.e. several dozen) is much smaller compared to the thousands of byte-code instructions of the mCommerce application. In the trials, and in comparison with the time required to acquire and process the other AFs (e.g. biometrics), no performance degradation was noticeable once LocAuth was enabled in the prototype.

5.2.3 Conclusion on LocAuth

LocAuth provides real-time client authentication for mCommerce applications on phones. LocAuth challenges remote clients with a randomly generated, one-time, location-obfuscated challenge / response program that includes six AFs for robust and reliable client authentication. I.e. an attacker needs to breach all six AFs simultaneously to get illegitimately authenticated. Furthermore, the use of two independent sources to generate the (de)obfuscation key adds strength to the location AF of LocAuth. Thus, LocAuth offers qualified level of security tailored to mCommerce applications for phones, yet remains practical with minimum overhead.

On the authenticator side, the client’s personal data (e.g. PIN), the client’s current phone location and serving cell-ID are integrated into the LocAuth challenge. Client’s personal data is directly available from the authenticator’s own database after client’s enrolment. Phone location and cell-ID are accessible from the MNO, who is constantly determining the phone’s location during normal operation anyway. Performed trials showed that generation of a fresh LocProg by the authenticator takes about 0.8 seconds. On the client side, the current location and serving cell-ID are accessible without overhead on the phone using the onboard GPS receiver. Client's personal data is retrieved from the phone storage (KoP) and entered by the client (PIN). Trials showed furthermore that interpretation of LocProg on the client's phone does not add a noticeable delay to the authentication process.

LocAuth is practical and user-friendly, because the client has to enter only the PIN. I.e. the other AFs (e.g. KoP, cell-ID, location) are automatically calculated by LocAuth and do not require any additional effort from the client. For the PIN, two LocAuth configurations are possible to decrease the required client's effort further. These configurations are:

1) The same PIN is used for oBiometrics and LocAuth. This configuration introduces minimum inconvenience to the client, because the client needs to remember and enter only one single PIN.
2) Different PINs are used for oBiometrics and LocAuth. This configuration is recommended, because an attacker needs to know two pieces of information from the client to break the authentication system.

LocAuth requires that the client is static or moves only slowly during the authentication process. If the client moves too fast (e.g. in a car on a motorway or a train), then the client's location changes too much between obfuscation and deobfuscation of LocProg. I.e. the location-based (de)obfuscation keys used by the authenticator and client are different. This results in an invalid LocProg response, which will not be accepted by the authenticator. Trials showed that the time between creation of LocProg by the authenticator and calculation of the LocProg result on the phone are about 30 seconds. This means that the client can move at a maximum speed of 36 km/h (22 miles per hour), because the LocAuth key generation technique is able to handle a location difference of up to 300 metres (cp. section 5.2.1). To minimise any delay during execution of the mCommerce application, LocAuth is executed before MORE-BAILS (cp. Figure 30).

LocAuth can be used to secure all phone applications that require the assurance of client's location as well as real-time. However, availability of the obfuscated interpretation framework on the phone is compulsory to handle the location-obfuscated LocProg correctly.

6 CONCLUSIONS AND FUTURE WORK

LOTA proposes working solutions to enhance security and reliability of client authentication in mCommerce applications that are run on Smartphones. The novelties of LOTA centre on the secure and reliable integration of location and real-time as new authentication factors into established authentication methods based on classic authentication factors like PINs, tokens, and biometrics.

In particular, LOTA proposes unique and novel approaches to enhance the security of remote client authentication solutions. These approaches are:

1) Provide solutions that attempt to bring back the face-to-face characteristics (i.e. clearly defined "who, where, when and how" of an authentication attempt) of office-based authentication into remote authentication performed on the phone.
2) Provide a method to protect the host environment to securely and correctly execute the authentication process on the phone.

These two solutions, when implemented inside any mCommerce application on phones will:

1) Reduce the risk of attacks (e.g. client's impersonation, message replay or distance attacks, in which an attacker claims to be at a certain location to deceive the authentication system about the attacker's real whereabouts) on the application due to the integration of multi-factor authentication.
2) Increase the authenticator's (i.e. service provider like financial institutes or online shop merchants) trust into the received client's data due to the correct and verified execution of the authentication process on the client's phone.

Consequently, the proposed solutions of LOTA shall help to convince more cloud service providers to offer their services via phone applications to their clients.

All proposed novelties and developed schemes were successfully tested and evaluated on state-of-the-art Android-based Smartphones. The performed trials clearly showed the commercial viability and practicality of the implemented scheme algorithms.

6.1 Achievements and Novelties

The first of LOTA's achievements is a novel MORE-BAILS authentication scheme that combines multiple authentication factors into a novel and secure One-Time Multi-Factor Biometric Representation (OTMFBR) [135]. This proposed OTMFBR representation combines:

1) Classic authenticator factors like PINs, tokens, and biometrics. PIN and biometrics assure the authenticator about the actual client performing this authentication attempt. Tokens ensure that the genuine client's phone is used in the attempt. This reduces the risk of impersonation attacks, because the genuine client and his/her phone are required to be present during authentication.
2) New authentication factors like client's current location, obtained from two independent sources, and real-time to assure the authenticator about the freshness of the received authentication data and the current geographical position of the client.

The integration of independently verified location information [34] and real-time is a second achievement of LOTA and guarantees the one-time property of the OTMFBR and thus, reduces the risk of replay or other distance attacks [67].

The design of the OTMFBR algorithm is arranged in such a way that the overall authentication process completely fails, if any of the authentication factors cannot be verified. This means that an attacker needs to break all authentication factors to be successfully authenticated. Obtained experimental results showed that the MORE-BAILS scheme can achieve a 0% false acceptance rate even if one or more of the authentication factors are successfully compromised by an attacker. This makes the OTMFBR a secure representation of the client's authentication data. Furthermore, the designed OTMFBR representation does not leak any useful information about any of the authentication factors, if an attacker has undermined one or more of these authentication factors. This means that an attacker does not benefit in any way from a partially successful attack (i.e. the attacker has breached some of the authentication factors).

Commercial viability, practicality, usability, and the client's privacy are also important requirements for mCommerce authentication solutions on phones. The proposed MORE-BAILS solution is designed to:

1) Be user-friendly and easy to use for the client during authentication on the phone. MORE-BAILS requires only dedicated client actions to captured the fresh biometrics and entering the PIN. All other authentication factors are automatically determined by MORE-BAILS.
2) Be cost effectively implemented by the authenticator. A third contribution of MORE-BAILS is to use the available phone onboard sensors to capture the client's biometrics, onboard GPS receiver for location, keypad for PIN, and SIM card to securely store the KoP, which are then combined to the OTMFBR [135]. MORE-BAILS does not require any additional hardware components to be integrated into the client's phone.
3) Be easily integrated to any phone application and distributed to the clients. MORE-BAILS can be attached to any mCommerce phone application and then distributed as part of this mCommerce application to the authenticator's clients.
4) Maintain the privacy of the client's location data without loss of security or introducing computational overhead on the authenticator or client side. To preserve the client's location privacy, LOTA proposes as a fourth achievement two novel techniques to transfer the client's actual physical location into another secure location-domain [75]. These unique transformation algorithms maintain the actual distances between client and authenticator determined locations. Hence, this enables the authenticator to verify the claimed location without knowing the actual current location of the client, i.e. to avoid breaching the client's location privacy. Implementation and experimental evaluation of the proposed privacy preserving location transfer techniques showed that the required calculations can be achieved in 0.5 seconds on today's available Smartphones, making these techniques commercial viable and practical.

Three approaches to protect the host environment (i.e. the client's phone) of the authentication process against malicious modifications were identified by LOTA as a fifth achievement [98]. Protecting the host environment is important in remote authentication, because:

1) It assures the authenticator about the correct capturing and processing of the authentication data on the phone.
2) It stops attackers from using the authentication application to get illegitimately authenticated.

The benefits, drawbacks and strengths of:

1) using SIM cards to host the authentication process,
2) using custom chips for authentication,
3) using software-based protection of the authentication process,

in terms of security, implied costs, flexibility and viability to deploy on today's phones were investigated. LOTA has concluded as a sixth achievement that software-based protection techniques are the most viable option to protect the host environment of the authentication process [98].

A seventh new contribution of LOTA is the proposed software protection framework "oBiometrics" algorithm [126]. This framework binds the correct execution of the authentication application to the genuine client through biometric-based obfuscated interpretation. The oBiometrics algorithm is implemented so that correct interpretation of the application on the phone will fail, if the wrong biometric-based key is used. This stops any unauthorised use of the application and prevents that an attacker is able to use the application.
Experiments and theoretical analyses of oBiometrics proved that it is difficult for an attacker to bypass the oBiometrics protection. I.e. it is not possible to understand the real semantics of the obfuscated application without having the correct deobfuscation key. Even if an attacker has full access to the phone and to the application code, the attacker cannot make sense of the application code.

Even though oBiometrics was developed to protect the authentication process of mCommerce applications, oBiometrics can be similarly used to protect all kinds of other applications on phones. For example, publishers can use oBiometrics to ensure that only their subscribers can use the application to access the latest newspaper on their phones.

LocAuth, an eight new contribution of LOTA, ensures the real-time and one-time property of an authentication attempt by using a location-obfuscated challenge / response program [134]. LocAuth can be seamlessly combined with oBiometrics and will be used by the authenticator to check in real-time if a genuine client is present at the phone. If this is not the case, then the authenticator can immediately restrict access to the provided service via the provided application.

LocAuth requires establishing the same location-based key independently on the client and authenticator side for successful (de)obfuscation of the challenge/response program. LOTA proved as a ninth contribution that this is not always possible, as was claimed by other researchers [61]. To overcome this problem of independent location-based key generation, three new approaches to independently generate location-based keys have been proposed. Tests and trials proved that these proposed approaches minimise the risk of false rejection of genuine clients, yet remain the independence of the key generation.

To ensure viability and commerciality of oBiometrics and LocAuth (oB&L), oB&L prototypes were tested on state-of-the-art Android-based Smartphones. Experiments and trials showed that the introduced overhead by oB&L is insignificant compared to the required computational time of the mCommerce application.

The oB&L prototypes and the adapted code of the Android virtual machine will be submitted to the Google Android development software repository to be integrated into future versions of the Android operating system. This allows the use of oB&L by the general public.

oB&L prototypes can be straightforward adapted to work with phones using a Java virtual machine to execute client applications by redefining the instruction substitution groups. This adaption allows, for example, the use of oB&L on BlackBerry Smartphones. However, the new Windows 8 operating system, for example used on Nokia Smartphones, uses the Windows Run-Time environment to execute client applications. To use oB&L on these phones, the obfuscated interpretation needs to be executed outside a virtual machine. This aspect has not been considered in LOTA.

6.2 Future Perspective

Proposed future work arising from the achievements of LOTA addresses topics in the area of biometrics, in particular age recognition and continuous biometric-based authentication on phones, as well as software protection.

1) An in-depth investigation and experiments need to be conducted to establish how the strength of oBiometrics can be further enhanced by the incorporation of other obfuscation techniques (e.g. opaque variables or control flow obfuscations) and how this combination will influence the performance of the oBiometrics obfuscation system. It is expected, that these techniques can be used together, because the other obfuscation techniques will be applied before the instruction substitution of oBiometrics takes place, i.e. the already obfuscated application code appears as ordinary code to the oBiometrics code translator. The combination should not affect the obfuscated interpretation security, because the obfuscated application will be oBiometrics protected. In contrast, the use of further obfuscation techniques should enhance the security of oBiometrics, because the added obfuscation techniques build a first line of defence against application code reverse-engineering.

   In some cases, the number of application instructions increases, because of the additional obfuscation technique. The security of oBiometrics benefits from this increase, because more instructions are available for substitution. However, the oBiometrics performance decreases, because of the computational overhead added by the obfuscation technique. It needs to be investigated, how the added security compares to the introduced overhead in the oBiometrics scheme.
2) Despite the fact that HC/HD SIM cards still have not reached the end-consumer market, an observation of the further developments in this technological area are worthwhile. Once these advanced SIM cards are available for testing, it is interesting to investigate, if oBiometrics can be utilised directly on these advanced SIM cards. Tight combination of oBiometrics and HC/HD SIM cards further raises the bar for an attacker to break the authentication system, because:
   a) oBiometrics is completely executed in a secure environment,
   b) authentication factors like the Key-On-Phone remain fully inside the secure SIM card memory and do not need to be transferred into the potential hostile processing environment of the phone.

3) The oBiometrics and MORE-BAILS prototypes use the biometric face modality to securely and reliably identify the genuine client. Integration of continuous biometric-based authentication factors can strengthen the real-time and freshness guarantee of oBiometrics and MORE-BAILS further.

   Inertial sensors such as accelerometer and gyroscope, which are now available on nearly all Smartphones, offer interesting opportunities for the integration of continuous authentication. Researchers already proposed techniques to use accelerometers on phones for gait recognition [14] or for measurements of pre-defined and then explicitly performed gestures to identify clients [136]. However, using pre-defined gestures that are not directly related to the mCommerce authentication process introduce an additional burden to the client, because the client has to perform extra actions.

   “Visual” observations of, for example, the holding angle or the typical movement of the phone during usage of applications or making phone calls clearly indicated differences between clients. These typical movements can be used as a continuous authentication factor in oBiometrics and MORE-BAILS to support the currently employed authentication factors. This continuous authentication is interesting from a usability point of view in particular, because it “comes for free” and does not introduce any further burden to the client.

4) Age recognition aims at estimating the actual age of a person based on, for example, his/her recorded voice or captured face. Combination of age recognition with oBiometrics could lead to a novel solution to prevent certain software to be used by clients of certain age groups.

   Many young children have nowadays their own Smartphone and are also able to install any application from the various marketplaces available. However, it is often not desired that these young children are able to use all of these available applications because of youth endangering contents. An age recognising adaption of oBiometrics can be used to ensure that only clients above a certain age group are able to use these applications. Verifying the age directly prior to execution of the application has a main advantage over verification prior to application installation. Children cannot use an application installed on a different (e.g. their parents’) phone. This would be possible, if the age is checked only once during application installation.

References

[1] BROADWAN - Broadband services for everyone over fixed wireless access networks. [Online]. http://www.broadwan.org.

[2] SecurePhone Project. [Online]. http://www.secure-phone.info.

[3] COST ic0803 - RF/Microwave Communication Subsystems for Emerging Wireless Technologies. [Online]. http://www.cost-ic0803.org.

[4] C. Vielhauer, *Biometric User Authentication for IT Security: From Fundamentals to Handwriting*.: Springer-Verlag New York Inc, 2005.

[5] B. Schneier, "Two-factor authentication: too little, too late," *Communications of the ACM*, vol. 48, no. 4, p. 136, 2005.

[6] S.Z. Li and A.K. Jain, *Encyclopedia of Biometrics*.: Springer US, 2009.

[7] D. Florencio and C. Herley, "Where do security policies come from?," in *Proceedings of the Sixth Symposium on Usable Privacy and Security*, New York, NY, USA, 2010, pp. 10:1--10:14.

[8] A. Adams and M.A. Sasse, "Users are not the enemy," *Communications of the ACM*, vol. 42, no. 12, pp. 40-46, 1999.

[9] M.A. Sasse, S. Brostoff, and D. Weirich, "Transforming the 'weakest link' - a human/computer interaction approach to usable and effective security," *BT Technology Journal*, vol. 19, no. 3, pp. 122-131, 2001.

[10] N.L. Clarke and S.M. Furnell, "Authentication of users on mobile telephones-a survey of attitudes and practices," *Computers & Security*, vol. 24, no. 7, pp. 519-527, 2005.

[11] A.K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 14, no. 1, pp. 4-20, 2004.

[12] P. Abeni, P. Baltatu, and R. D’Alessandro, "A face recognition system for mobile phones," *Information Security Solutions Europe*, pp. 211-217, 2006.

[13] W. Shi, J. Yang, Y. Jiang, F. Yang, and Y. Xiong, "SenGuard: Passive user identification on smartphones using multiple sensors," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE*

7th International Conference on, *IEEE*, 2011, pp. 141-148.

[14] M.O. Derawi, C. Nickel, P. Bours, and C. Busch, "Unobtrusive user-authentication on mobile phones using biometric gait recognition," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*, *IEEE*, 2010, pp. 306-311.

[15] K. Delac and M. Grgic, "A survey of biometric recognition methods," in *Electronics in Marine, 2004. Proceedings Elmar 2004. 46th International Symposium*, *IEEE*, 2004, pp. 184-193.

[16] H. Sellahewa and S.A. Jassim, "Illumination and expression invariant face recognition: Toward sample quality-based adaptive fusion," in *Biometrics: Theory, Applications and Systems, 2008. BTAS 2008. 2nd IEEE International Conference on*, *IEEE*, 2008, pp. 1-6.

[17] Anil K. Jain, Karthik Nandakumar, and Abhishek Nagar, "Biometric template security," *EURASIP Journal on Advances in Signal Processing. Special Issue on Biometrics*, vol. 2008, pp. 1-17, 2008.

[18] Nalini K. Ratha, Jonathan H. Connell, and Ruud M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Systems Journal*, vol. 40, no. 3, pp. 614-634, 2001.

[19] M. Faundez-Zanuy, "On the vulnerability of biometric security systems," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 19, no. 6, pp. 3-8, Jun 2004.

[20] Harin Sellahewa, "Wavelet-based automatic face recognition for constrained devices," Applied Computing Department, The University of Buckingham, UK, Ph.D. dissertation 2006.

[21] Naseer Al-Jawad, "Exploiting Statistical Properties of Wavelet Coefficients for Image/Video Processing and Analysis Tasks," Applied Computing Department, The University of Buckingham, UK, Ph.D. dissertation May 2009.

[22] Hisham Al-Assam, Harin Sellahewa, and Sabah Jassim, "A Lightweight approach for biometric template protection," in *Proceedings of SPIE*, vol. 7351, March 2009, pp. 73510P.1-73510P.12.

[23] S.A. Jassim and H. Sellahewa, "A wavelet-based approach to face

verification/recognition," in *Proc. SPIE Symposium on Unmanned / Unattended Sensors and Sensor Networks II*, vol. 5986, 2005, pp. 77-86.

[24] Harin Sellahewa and Sabah A. Jassim, "Wavelet-based face verification for constrained platforms," , vol. 5779, 2005, pp. 173-183.

[25] A.B.J. Teoh, Y.W. Kuan, and S. Lee, "Cancellable biometrics and annotations on BioHash," *Pattern recognition*, vol. 41, no. 6, pp. 2034-2044, 2008.

[26] R.W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147-160, 1950.

[27] Hisham Al-Assam and Sabah Jassim, "Security evaluation of biometric keys," *Computers & Security*, vol. 31, no. 2, pp. 151-163, 2012.

[28] S.B. Wicker and V.K. Bhargava, *Reed-Solomon codes and their applications*.: Wiley-IEEE Press, 1999.

[29] R.C. Bose and D.K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and control*, vol. 3, no. 1, pp. 68-79, 1960.

[30] H. Liu, H. Ma, M. El Zarki, and S. Gupta, "Error control schemes for networks: An overview," *Mobile Networks and Applications*, vol. 2, no. 2, pp. 167-182, 1997.

[31] A. Stoianov, T. Kevenaar, and M. Van der Veen, "Security issues of biometric encryption," in *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference, Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference*, 2009, pp. 34-39.

[32] F. Hao, R. Anderson, and J. Daugman, "Combining cryptography with biometrics effectively," *IEEE Transactions on Computers*, pp. 1081-1088, 2006.

[33] K. Nandakumar, A.K. Jain, and S. Pankanti, "Fingerprint-based fuzzy vault: Implementation and performance," *Information Forensics and Security, IEEE Transactions on*, vol. 2, no. 4, pp. 744-757, 2007.

[34] Torben Kuseler and Ihsan Alshahib Lami, "Using Geographical Location as an Authentication Factor to enhance mCommerce Applications on Smartphones," *International Journal of Computer Science and Security*

(IJCSS), vol. 6, no. 4, p. 10, 2012, Under review.

[35] G. Sun, J. Chen, W. Guo, and K.J.R. Liu, "Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs," *Signal Processing Magazine, IEEE*, vol. 22, no. 4, pp. 12-23, 2005.

[36] U.S. Government. Official U.S. Government information about the Global Positioning System (GPS) and related topics. [Online]. http://www.gps.gov/

[37] NMEA 0183 - Standard for interfacing marine electronic devices, Version 4.00, 2008.

[38] Paul A. Zandbergen, "Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning," *Transactions in GIS*, vol. 13, no. s1, pp. 5-25, 2009.

[39] Skyhook. [Online]. http://www.skyhookwireless.com

[40] Axel Kuepper, *Location-Based Services: Fundamentals and Operation*.: Wiley Online Library, Oct 2005.

[41] TruePosition, U-TDOA: Enabling New Location-Based Safety and Security Solutions, Oct 2008.

[42] T. Kindberg, K. Zhang, and N. Shankar, "Context authentication using constrained channels," in *Mobile Computing Systems and Applications, 2002. Proceedings Fourth IEEE Workshop on, IEEE*, 2002, pp. 14-21.

[43] L. Scott and D.E. Denning, "Location Based Encryption & Its Role In Digital Cinema Distribution," Tech. rep. 2003.

[44] A.I.G.T. Ferreres, B.R. Alvarez, and A.R. Garnacho, "Guaranteeing the authenticity of location information," *IEEE Pervasive Computing*, pp. 72-80, 2008.

[45] D. Denning and P. MacDoran, "Location-Based Authentication: Grounding Cyperspace for Better Security," *Computer Fraud and Security Bulletin*, Feb 1996.

[46] S. Lo, D.S. De Lorenzo, P.K. Enge, D. Akos, and P. Bradley, "Signal authentication-a secure civil gnss for today," *inside GNSS*, vol. 4, no. 5, pp. 30-39, 2009.

[47] G. Becker, S. Lo, D. De Lorenzo, P. Enge, and C. Paar, "Secure Location Verification," *Data and Applications Security and Privacy XXIV*, pp. 366-

373, 2010.

[48] A. Haeberlen et al., "Practical robust localization over large-scale 802.11 wireless networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking, ACM,* 2004, pp. 70-84.

[49] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, New York, NY, USA, 2009, pp. 3:1--3:6.

[50] W. Luo and U. Hengartner, "VeriPlace: a privacy-aware location proof architecture," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM,* 2010, pp. 23-32.

[51] Z. Zhu and G. Cao, "Applaus: A privacy-preserving location proof updating system for location-based services," in *INFOCOM, 2011 Proceedings IEEE, IEEE,* 2011, pp. 1889-1897.

[52] V. Lenders, E. Koukoumidis, P. Zhang, and M. Martonosi, "Location-based trust for mobile user-generated content: applications, challenges and implementations," in *Proceedings of the 9th workshop on Mobile computing systems and applications, ACM*, 2008, pp. 60-64.

[53] T. Weigold, T. Kramp, and M. Baentsch, "Remote client authentication," *Security & Privacy, IEEE,* vol. 6, no. 4, pp. 36-43, 2008.

[54] F. Aloul, S. Zahidi, and W. El-Hajj, "Two factor authentication using mobile phones," in *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on, IEEE,* 2009, pp. 641-644.

[55] D. De Figueiredo, "The Case for Mobile Two-Factor Authentication," *Security & Privacy, IEEE,* vol. 9, no. 5, pp. 81-85, 2011.

[56] W.B. Hsieh and J.S. Leu, "Design of a time and location based One-Time Password authentication scheme," in *Wireless Communications and Mobile Computing Conference (IWCMC), 7th International, IEEE,* 2011, pp. 201-206.

[57] L. Scott and D.E. Denning, "A location based encryption technique and some of its applications," in *ION National Technical Meeting*, vol. 2003, 2003, pp. 730-740.

[58] A. Al-Fuqaha and O. Al-Ibrahim, "Geo-encryption protocol for mobile networks," *Computer Communications*, vol. 30, no. 11-12, pp. 2510-2517, 2007.

[59] G. Yan and S. Olariu, "An efficient geographic location-based security mechanism for vehicular adhoc networks," in *IEEE 6th International Conference on Mobile Adhoc and Sensor Systems, MASS'09, IEEE*, 2009, pp. 804-809.

[60] G. Yan, J. Lin, D.B. Rawat, and W. Yang, "A Geographic Location-Based Security Mechanism for Intelligent Vehicular Networks," *Intelligent Computing and Information Science*, pp. 693-698, 2011.

[61] H.C. Liao and Y.H. Chao, "A new data encryption algorithm based on the location of mobile users," *Information Technology Journal*, vol. 7, no. 1, pp. 63-69, 2008.

[62] X. Fang and J. Zhan, "Online Banking Authentication Using Mobile Phones," in *Future Information Technology (FutureTech), 5th International Conference on, IEEE*, 2010, pp. 1-5.

[63] J.M. McCune, A. Perrig, and M.K. Reiter, "Seeing-is-believing: using camera phones for human-verifiable authentication," in *Security and Privacy, 2005 IEEE Symposium on*, May 2005, pp. 110-124.

[64] K. Hickman and T. Elgamal, "The SSL protocol," Internet Draft RFC, Tech. rep. 1995.

[65] S. Kowalski and M. Goldstein, "Consumers' Awareness of, Attitudes Towards and Adoption of Mobile Phone Security," *Human Factors in Telecommunication (HFT)*, vol. 6, 2006.

[66] S. Furnell, N. Clarke, and S. Karatzouni, "Beyond the pin: Enhancing user authentication for mobile devices," *Computer Fraud & Security*, vol. 2008, no. 8, pp. 12-17, 2008.

[67] Ihsan A. Lami, Torben Kuseler, Hisham Al-Assam, and Sabah Jassim, "LocBiometrics: Mobile phone based multifactor biometric authentication with time and location assurance," in *Proc. 18th Telecommunications Forum (IEEE TELFOR 2010), IEEE Telfor*, Nov 2010.

[68] U.S. Federal Communications. 9-1-1 Service. [Online].

http://www.fcc.gov/pshs/services/911-services/.

[69] FollowUs. [Online]. http://www.followus.info.

[70] Marian Mohr, Christopher Edwards, and Ben McCarthy, "A study of LBS accuracy in the UK and a novel approach to inferring the positioning technology employed," *Comput. Commun.*, vol. 31, no. 6, pp. 1148-1159, 2008.

[71] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman, "From few to many: Generative models for recognition under variable pose and illumination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 643-660, 2001.

[72] Google Inc. Google Mobile Maps. [Online]. http://www.google.co.uk/mobile/maps/.

[73] W. Gellert, S. Gottwald, M. Hellwich, H. Kastner, and H. Kustner, *The VNR concise encyclopedia of mathematics*.: Van Nostrand Reinhold, 1989.

[74] K.S. Hasan et al., "Cost Effective GPS-GPRS Based Object Tracking System," *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, pp. 18-20, 2009.

[75] Torben Kuseler, Hisham Al-Assam, Sabah Jassim, and Ihsan A. Lami, "Privacy preserving, real-time and location secured biometrics for mCommerce authentication," in *Mobile Multimedia/Image Processing, Security, and Applications 2011*, *SPIE*, vol. 8063, SPIE, Bellingham, WA, Apr 2011.

[76] Sabah A. Jassim, Hisham Al-Assam, and Harin Sellahewa, "Improving performance and security of biometrics using efficient and stable random projection techniques," in *Proceedings of the 6th International Symposium on Image and Signal Processing and Analysis (ISPA)*, 2009, pp. 556-561.

[77] Hisham Al-Assam, Harin Sellahewa, and Sabah A. Jassim, "Multi-Factor Biometrics for Authentication: A false sense of security," in *Proceedings of the 12th ACM Workshop on Multimedia and Security*, 2010, pp. 81-88.

[78] Andreas Constantinou, White Paper: High Capacity SIMs, 2006, Company: Informa Telecoms & Media.

[79] Keith E. Mayes and Konstantinos Markantonakis, "On the potential of high

density smart cards," *Information Security Technical Report*, vol. 11, pp. 147-153, 2006.

[80] K. Rannenberg, "Identity management in mobile cellular networks and related applications," *Information Security Technical Report*, vol. 9, no. 1, pp. 77-85, 2004.

[81] V. Khu-Smith and C.J. Mitchell, "Enhancing E-commerce Security Using GSM Authentication," , *E-commerce and web technologies: 4th international conference, EC-Web,* Prague, Czech Repbulic, Sep 2003, pp. 72-83.

[82] T. Mantoro, A. Milisic, and M.A. Ayu, "Online payment procedure involving mobile phone network infrastructure and devices," in *Multimedia Computing and Systems (ICMCS), 2011 International Conference on*, Apr 2011, pp. 1-6.

[83] M. Witteman, "Advances in smartcard security," *Information Security Bulletin*, vol. 7, pp. 11-22, 2002.

[84] Lars Schnake, Carsten Rust, and Rebekka Neumann, SIM Card Based Security and Trust Management in Mobile Services, Jun 2009.

[85] Q. Tang, J. Zou, C. Fan, and X. Zhang, "A mobile identity authentication scheme of e-commerce based on Java-SIM card," in *Information Networking and Automation (ICINA), 2010 International Conference on*, *IEEE,* vol. 2, Oct 2010, pp. 114-118.

[86] A. De Santis et al., "An Extensible Framework for Efficient Secure SMS," in *Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on*, *IEEE*, 2010, pp. 843-850.

[87] ISO/IEC 7816: Identification cards -- Integrated circuit cards, 2011.

[88] J.R. Rao, P. Rohatgi, H. Scherzer, and S. Tinguely, "Partitioning attacks: or how to rapidly clone some GSM cards," in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, *IEEE*, 2002, pp. 31-41.

[89] S. Willassen, "Forensics and the GSM mobile telephone system," *International Journal of Digital Evidence*, vol. 2, no. 1, pp. 1-17, 2003.

[90] M. Becher et al., "Mobile security catching up? Revealing the nuts and bolts of the security of mobile devices," in *Security and Privacy (SP), 2011 IEEE Symposium on*, *IEEE*, 2011, pp. 96-111.

[91] K.E. Mayes and K. Markantonakis, *Smart cards, tokens, security and*

applications.: Springer-Verlag New York Inc, 2008.

[92] Clive Maxfield, *The Design Warrior's Guide to FPGAs*. Orlando, FL, USA: Academic Press, Inc., 2004.

[93] V. Betz, "FPGA challenges and opportunities at 40nm and beyond," in *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, *IEEE*, 2009, pp. 4-4.

[94] Celia Lopez-Ongil et al., "FPGA Implementation of Biometric Authentication System Based on Hand Geometry," in *FPL*, 2004, pp. 43-53.

[95] D.D. Hwang and I. Verbauwhede, "Design of portable biometric authenticators-energy, performance, and security tradeoffs," *Consumer Electronics, IEEE Transactions on*, vol. 50, no. 4, pp. 1222-1231, 2004.

[96] S. Drimer, "Volatile FPGA design security - a survey," *IEEE Computer Society Annual Volume*, pp. 292-297, 2008.

[97] N. Selmane, S. Bhasin, S. Guilley, and J.L. Danger, "Security evaluation of application-specific integrated circuits and field programmable gate arrays against setup time violation attacks," *Information Security, IET*, vol. 5, no. 4, pp. 181-190, 2011.

[98] Torben Kuseler, Ihsan Lami, Sabah Jassim, and Harin Sellahewa, "eBiometrics: an enhanced multi-biometrics authentication technique for real-time remote applications on mobile devices," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 7708, Apr 2010, pp. 77080E.1-77080E.9.

[99] S. Kent, "Protecting Externally Supplied Software in Small Computers," Cambridge, MA, USA, Tech. rep. Sep 1980.

[100] Jan M. Memon, Asma Khan, Amber Baig, and Asadullah Shah, "A Study of Software Protection Techniques," in *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering*, 2007, pp. 249-253.

[101] S. Chow, P. Eisen, H. Johnson, and P. C. Van, "White-Box Cryptography and an AES Implementation," in *Proceedings of the Ninth Workshop on Selected Areas in Cryptography (SAC 2002)*, 2002, pp. 250-270.

[102] C. Collberg and J. Nagra, *Surreptitious software: obfuscation, watermarking, and tamperproofing for software protection*.: Addison-Wesley Professional,

2009.

[103] David Aucsmith, "Tamper Resistant Software: An Implementation," in *Information Hiding*, 1996, pp. 317-333.

[104] P. Wang, S. Kang, and K. Kim, "Tamper resistant software through dynamic integrity checking," in *Proceedings of the 2005 Symposium on Cryptography and Information Security*., 2005.

[105] Matias Madou et al., "On the Effectiveness of Source Code Transformations for Binary Obfuscation," in *Software Engineering Research and Practice*, 2006, pp. 527-533.

[106] Xuesong Zhang, Fengling He, and Wanli Zuo, "A Java Program Tamper-Proofing Method," in *SECTECH '08: Proceedings of the 2008 International Conference on Security Technology*, Washington, DC, USA, 2008, pp. 71-74.

[107] E.J. Chikofsky and J.H. Cross, "Reverse engineering and design recovery: A taxonomy," *Software, IEEE*, vol. 7, no. 1, pp. 13-17, 1990.

[108] Boaz Barak et al., "On the (im)possibility of obfuscating programs," in *Lecture Notes in Computer Science*, 2001, pp. 1-18.

[109] Andrew W. Appel, Deobfuscation is in NP, Aug 2002.

[110] M. Ceccato et al., "The effectiveness of source code obfuscation: an experimental assessment," in *Program Comprehension, 2009. ICPC'09. IEEE 17th International Conference on*, *IEEE*, 2009, pp. 178-187.

[111] Christian Collberg, Clark Thomborson, and Douglas Low, "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs," in *Conference Record Of The ACM Symposium On Principles of Programming Languages 1998, POPL'98*, *Association for Computing Machinery Inc.*, vol. 25, San Diego, CA, Jan 1998, pp. 184-196.

[112] Christian Collberg, Clark Thomborson, and Douglas Low, "A taxonomy of obfuscating transformations," Tech. rep. Jul 1997.

[113] M. Karnick, J. MacBride, S. McGinnis, Y. Tang, and R. Ramachandran, "A Qualitative Analysis of Java Obfuscation," in *Proceedings of 10th IASTED International Conference on Software Engineering and Applications, Dallas TX, USA*, *Citeseer*, 2006.

[114] Cataldo Basile, Stefano Di Carlo, Thomas Herlea, and Jasvir Nagra, Towards

a Formal Model for Software Tamper Resistance, 2009.

[115] Paul C. van Oorschot, "Revisiting Software Protection," in *ISC*, vol. 2851, 2003, pp. 1-13.

[116] J.M. Memon, A. Mughal, F. Memon, and others, "Preventing Reverse Engineering Threat in Java Using Byte Code Obfuscation Techniques," in *Emerging Technologies, 2006. ICET'06. International Conference on*, *IEEE*, 2006, pp. 689-694.

[117] Z. Tang, X. Chen, D. Fang, and F. Chen, "Research on Java Software Protection with the Obfuscation in Identifier Renaming," in *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, *IEEE*, 2009, pp. 1067-1071.

[118] Praveen Sivadasan, P. Sojan Lal, and Naveen Sivadasan, "JDATATRANS for Array Obfuscation in Java Source Code to Defeat Reverse Engineering from Decompiled Codes," *CoRR*, vol. abs/0809.3503, 2008.

[119] Michael Batchelder and Laurie J. Hendren, "Obfuscating Java: The Most Pain for the Least Gain," in *Compiler Construction*, vol. 4420, 2007, pp. 96-110.

[120] T.W. Hou, H.Y. Chen, and M.H. Tsai, "Three control flow obfuscation methods for Java software," *IEE Proceedings-Software*, vol. 153, no. 2, p. 80, 2006.

[121] D. Dolz and G. Parra, "Using exception handling to build opaque predicates in intermediate code obfuscation techniques," *Journal of Computer Science & Technology*, vol. 8, no. 2, 2008.

[122] M. Madou et al., "Software protection through dynamic code mutation," *Information Security Applications*, pp. 194-206, 2006.

[123] Akito Monden, Antoine Monsifrot, and Clark Thomborson, "A framework for obfuscated interpretation," in *ACSW Frontiers '04: Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation*, Darlinghurst, Australia, 2004, pp. 7-16.

[124] Xuesong Zhang, Fengling He, and Wanli Zuo, "A Framework for Mobile Phone Java Software Protection," in *ICCIT '08: Proceedings of the 2008*

Third International Conference on Convergence and Hybrid Information Technology, Washington, DC, USA, 2008, pp. 527-532.

[125] Bruce Schneier, Crypto-Gram Newsletter, Feb 1999.

[126] Torben Kuseler, Ihsan A. Lami, and Hisham Al-Assam, "oBiometrics: A Software protection scheme using biometric-based obfuscation," in *2011 African Conference on Software Engineering and Applied Computing (ACSEAC)*, Cape Town, South Africa, Sep 2011.

[127] GNU Project. (2007, June) GNU General Public License V3.0. [Online]. http://www.gnu.org/licenses/gpl.html.

[128] Auguste Kerckhoffs, "La cryptographie militaire," *Journal des sciences militaires*, vol. IX, pp. 5-83, Jan 1883.

[129] Gartner Inc., Gartner Says Sales of Mobile Devices Grew 5.6 Percent in Third Quarter of 2011; Smartphone Sales Increased 42 Percent, Nov 2011.

[130] The Apache Software, Apache License, Version 2.0, Jan 2004.

[131] Android Developers. Android System Architecture. [Online]. http://www.android.com.

[132] Shane Conder and Lauren Darcey, *Android Wireless Application Development*, 2nd ed. Upper Saddle River, NJ: Addison-Wesley Professional, 2010.

[133] An assembler/disassembler for Android's dex format. [Online]. http://code.google.com/p/smali/.

[134] Torben Kuseler and Ihsan Alshahib Lami, "dLocAuth: a dynamic multifactor authentication scheme for mCommerce applications using independent location-based obfuscation," in *Mobile Multimedia/Image Processing, Security, and Applications 2012, SPIE*, SPIE, Bellingham, WA, Apr 2012.

[135] Torben Kuseler, Hisham Al-Assam, and Ihsan Alshahib Lami, "One Time Multi Factor Biometric Representation (OTMFBR) for remote client authentication," Patent Application Number 1109832.4, 2011.

[136] R. Mayrhofer and H. Gellersen, "Shake well before use: Intuitive and secure pairing of mobile devices," *Mobile Computing, IEEE Transactions on*, vol. 8, no. 6, pp. 792-806, 2009.