PROCEEDINGS

2016 IEEE 35th International Symposium on Reliable Distributed Systems

Workshops

— SRDSW 2016 —

PROCEEDINGS

2016 IEEE 35th International Symposium on Reliable Distributed Systems Workshops

— SRDSW 2016 —

26–29 September 2016
Budapest, Hungary

Sponsored by

IEEE
IEEE Computer Society
Technical Committee on Distributed Processing (TCDP)

Supported by

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Measurement and Information Systems

CPS Conference Publishing Services

Los Alamitos, California
Washington • Tokyo

IEEE computer society

IEEE Computer Society Order Number E5938
BMS Part Number CFP1613R-ART
ISBN 978-1-5090-5259-2

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Individual paper REPRINTS may be ordered at: <reprints@computer.org>

Editorial production by Randy Bilof
Cover art production by Mark Bartosik

IEEE Computer Society
***Conference Publishing Services* (CPS)**
http://www.computer.org/cps

2016 IEEE 35th Symposium on Reliable Distributed Systems Workshops

SRDSW 2016

Table of Contents

Workshop on Mobility and Cloud Security and Privacy (WMCSP 2016)

Access Control and Intrusion Detection

Biometric Authentication

Security and Safety

The 4th Workshop on Planetary-Scale Distributed Systems (W-PSDS 2016)

Planetary-Scale Distributed Systems

Message from the WMCSP 2016 Workshop Co-Chairs

The Workshop on Mobility and Cloud Security & Privacy brings together researchers and practitioners from security, mobile, and cloud computing communities to present the state of the art, discuss emerging challenges and trends, and propose novel techniques, implementations, and deployments of security, privacy, and reliability solutions for cloud and mobile services.

We have received a set of high-quality submissions, from which 8 high-quality papers will be presented at the workshop. This selection will certainly offer an exciting one-day workshop, with fruitful discussions.

We thank all authors of submitted papers for their relevant contributions to the workshop. We are also grateful to the SRDS 2016 Workshop Co-Chairs, Miguel Correia and Sanjay Madria, and the General Co-Chairs, András Pataricza and Istvan Majzik, for their help during the preparation of the workshop. We are also grateful to the Publication Chair, Andrea Ceccarelli, for his help in the production of the workshop proceedings. We wish to thank all the members of the technical program committee for providing detailed reviews on a short notice. Thanks to their efforts, every paper had between 3 and 5 reviews that will help the authors improve their work. We also thank the external reviewers who provided expert advice on selected topics and improved the overall selection process.

We look forward to seeing you at SRDS 2016 and at the Workshop on Mobility and Cloud Security & Privacy.

Hugues Mercier, Université de Neuchâtel, Switzerland
João Nuno Silva, INESC-ID, IST, Portugal
Luís Rodrigues, INESC-ID, IST, Portugal
WMCSP 2016 Workshop Co-Chairs

SRDS 2016 Symposium Organizers

Symposium Co-Chairs
András Pataricza, Budapest University of Technology and Economics, Hungary
Istvan Majzik, Budapest University of Technology and Economics, Hungary

Technical Program Committee Co-Chairs
Matti A. Hiltunen, AT&T Labs Research, Bedminster, NJ, USA
Sébastien Tixeuil, Université Pierre et Marie Curie, Paris, France

Workshop Co-Chairs
Miguel Correia, INESC-ID, Instituto Superior Tecnico, Portugal
Sanjay Madria, Missouri University of Science and Technology, USA

PhD Forum Co-Chairs
Jiannong Cao, Hong Kong Polytechnic University
Antonio Casimiro, University of Lisbon, Portugal

***Prof. C.V. Ramamoorthy Best Paper Award* Committee**
Andrea Bondavalli, University of Florence, Italy
Felicita di Giandomenico, ISTI-CNR, Italy
Takahiro Hara, Osaka University, Japan
Sanjay Madria, Missouri University of Science and Technology, USA

Publication Chair
Andrea Ceccarelli, University of Florence, Italy

Publicity Co-Chairs
Leszek T. Lilien, Western Michigan University, USA
Gilles Tredan, LAAS-CNRS, France

Local Arrangements Chair
Gábor Huszerl, Budapest University of Technology and Economics, Hungary

Steering Committee Liaison
Andrea Bondavalli, University of Florence, Italy

WMCSP 2016 Workshop Organizers

Workshop Co-Chairs

Hugues Mercier, Université de Neuchâtel, Switzerland
João Nuno Silva, INESC-ID, IST, Portugal
Luís Rodrigues, INESC-ID, IST, Portugal

Program Committee

Driss Aboulkassimi, Commissariat à l'Énergie Atomique et aux Énergies Alternatives, France
Gonzalo Bailador del Pozo, Universidade Politecnica de Madrid, Spain
Manuel Barbosa, Universidade do Porto, Portugal
Georg Carle, TU Munchen, Germany
Pascal Felber, Université de Neuchâtel, Switzerland
Kévin Huguenin, LAAS, France
Ruediger Kapitza, TU Braunschweig, Germany
Diogo Mónica, Docker, USA
Rui Oliveira, Universidade do Minho, Portugal
Emanuel Onica, UAIC, Romania
Asmund Skomedal, Norsk Regnesentral, Norway
Paulo Sousa, Maxdata, Portugal
Ferucio Tiplea, UAIC, Romania

External Reviewers

Workshop on Mobility and Cloud Security & Privacy
WMCSP 2016

Miguel Correia
Heiko Niedermayer
Bernardo Portela
Nuno Santos

Fourth Workshop on Planetary-Scale Distributed Systems
W-PSDS 2016

Albert van der Linde

2016 IEEE 35th Symposium on Reliable Distributed Systems Workshops

sKnock: Port-Knocking for Masses

Daniel Sel, Sree Harsha Totakura, Georg Carle
Chair of Network Architectures and Services
Technical University of Munich
Email: {sel, totakura, carle}@in.tum.de

***Abstract*—Port-knocking is the concept of hiding remote services behind a firewall which allows access to the services' listening ports only after the client has successfully authenticated to the firewall. This helps in preventing scanners from learning what services are currently available on a host and also serves as a defense against zero-day attacks. Existing port-knocking implementations are not scalable in service provider deployments due to their usage of shared secrets. In this paper we introduce an implementation of port-knocking based on x509 certificates aimed towards being highly scalable.**

***Index Terms*—Port-knocking, dynamic firewall, x509**

I. Introduction

Port-knocking is the concept of hiding remote services behind a firewall which drops all incoming connections to the services by default, but allows them only after the client has authenticated to the firewall. This can be seen as an additional layer of security which provides protection from port scans and exploits on the services from unauthorized clients.

There exists many implementations of port-knocking as of today and most of them employ authentication based on shared secrets. In these implementations the firewall is configured to authenticate a client based on its corresponding secret. While this approach is simple and efficient for a small number of clients, it quickly becomes unmanageable for a service provider with a large and dynamically changing client base.

Another problem with authentication based on secrets is that it is common for service providers to offer multiple servers to a common pool of clients for reasons of providing redundancy and load-balancing. In such a setup, the servers or the firewalls protecting the services have to synchronise the port-knocking secrets shared with each client. Furthermore, when a service provider uses a cloud provider for service delivery, the secrets have to be configured in the cloud provider's infrastructure which may give away the size of the client base of the service provider to the cloud provider.

To our knowledge there exists no implementation of port-knocking which demonstrates scalability on-par with what is required for a service provider using cloud computing infrastructure. To overcome the scalability barrier, we propose *sKnock* (named after 'scalable Knock'), our approach towards port-knocking using public-key cryptography to address scalability. Here, we choose to use the X509 [1] certificate standard due to its popularity. The motivation behind using certificates is that a certificate's validity can be checked by having the certificate of the signer; while adhering to x509 allows us to encode authentication information as x509 extensions and be able to use renowned libraries such as OpenSSL to parse the certificates. This, albeit a lookup in the revocation database which is usually small, aids in improving the scalability of authentication. Moreover, this approach is not subjected to the problem of synchronising the secrets among different servers and mitigates the privacy problem of knowing the size of client base as the firewall now only requires the certificate of the certification authority (CA) to authenticate the clients.

Our contributions to this paper are a one-way communication protocol to authenticate clients to the port-knocked firewall, an implementation of this protocol in a client and a server component. The client component provides a C library which allows applications to integrate port-knocking functionality. The server component integrates with the firewall and dynamically configures it to allow authenticated clients to communicate with the services behind the firewall.

The text in this paper is organised as following: the next section introduces an attacker model. We use it to evaluate some of the existing port-knocking approaches and their authentication process in detail to highlight the differences with our approach in Section III. Section IV introduces sKnock describing the authentication protocol and the design decisions we took to overcome common pitfalls. Section V describes our evaluations of an implementation of sKnock in Python. Its limitations are then presented in Section VI. Finally, as part of conclusion we describe the pros and cons of sKnock compared to others and suggest some future work in Section VII.

II. Attacker Model

For an attacker interested in attacking the services protected by port-knocked firewall, he has to either pose as a valid client of the service or defeat the port-knocking protection. For public services, an attacker can easily become a valid client. Therefore, we consider an attacker model where the attacker is interested in attacking the port-knocked firewall.

In this model, we consider the following capabilities to model different types of attackers:

- A1 Record any number of IP packets between a given source and destination and replay them from own IP address.
- A2 Modify, and suppress any number of IP packets from a given source and destination.
- A3 Send IP packets from any IP address and receive packets destined to any IP address.

Capability A1 can be acquired by an attacker by snooping anywhere on the network route between source and destination. Additionally, if the attacker can position himself as a

978-1-5090-5259-2/16 $31.00 © 2016 IEEE
DOI 10.1109/SRDSW.2016.11

hop anywhere in the route, he gains capability A2. A3 can be acquired by positioning himself in the link connecting the firewall to the Internet.

In addition to this, we also consider a valid client to be an attacker if the client's validity is revoked due to some reason and the client then tries to exploit the port-knocked firewall. Therefore, we assign the following properties:

- B1 Attacker knows that the firewall uses port-knocking.
- B2 Attacker may have previously port-knocked successfully as a valid client.

Notice that capabilities B1 and B2 are easier to acquire than A1, A2, and A3. B1 is even easier as it is possible to learn the existence of port-knocked firewall if that information is public (if it is advertised by the service provider).

In the rest of the text, we refer to attackers with a combination of these capabilities using the set notation: an attacker with capabilities A1 and B1 is termed as {A1, B1} attacker. If an attacker has a single capability, we will ignore the braces, *i.e.* A1, B1 attackers mean two type of attackers each with a capability, but not both capabilities.

Finally we assume that attackers cannot decrypt encrypted content and cannot forge signatures for arbitrary parties without the knowledge of their keys. This can be ensured in practice by proper usage of cryptography.

III. Related Work

Port-knocking is historically done by sending packets to different ports in a predefined static sequence. The sequence is kept secret and is shared with authorised clients. The firewall monitors the incoming packets and opens the corresponding port for a remote service for a client if the destination port numbers of a sequence of packets coming from that client match the its corresponding predefined sequence. An A1 attacker can observe this sequence and later replay it to defeat port-knocking.

Variants of port-knocking which use multiple packets to convey authenticating information to the firewall are termed under *Hybrid Port-Knocking*. To defend against A1 attackers, the sequence can be made dynamic with the usage of cryptography, *e.g.* by deriving the sequence from a time based one-time password (TOTP) [2]. However it is vulnerable to {A1, A2} attackers because the attacker can suppress a suspected sequence of packets from reaching the firewall and replay it from his host.

To defend against {A1, A2} attackers the IP address of the client needs to be encoded into the authenticating information in a way that that the firewall can retrieve it to open the port in the firewall for that client. If the client's IP address is in cleartext, then the authenticating information should contain a message authentication code (MAC), *e.g.* through HMAC [3], so that the attacker cannot rewrite it with his IP address.

However, encoding the client IP address in authentication information causes problems for clients behind NAT. Such clients are required to know their NAT's public IP address to be able to successfully knock the firewall. This may, however, lead to *NAT-Knocking* attacks [4] as the NAT's IP address is shared by other clients in the network and one of them could be an attacker. Aycock et. al. [5] proposed an approach based on challenge-response to solve this problem. This approach requires a three-way handshake where the client initiates by sending a request to the firewall. The request contains an identifier and the client's IP address. The firewall then sends a challenge containing the IP address it observed as the request's source IP, the IP address present in the request, and a random nonce, together with a MAC over these fields. The client then responds to this challenge by presenting a MAC over these fields with its pre-shared secret with the firewall. Since the handshake messages are authenticated with MAC, this approach is immune to {A1, A2} attackers.

A practical problem with hybrid port-knocking variants is that they fail when packets are delivered out-of-order to the firewall. To address this, the authenticating information could be sent as a single packet. This variant of port-knocking is termed as *Single Packet Authorisation* and is first documented to be used in *Doorman* [6]. Doorman authenticates clients based on a HMAC derived from the shared secret, port number to open for the client, username, and a random number. The random number is used to provide protection against A1 attackers as the firewall rejects a request with a number already seen. Since it does not include the client IP address, it is susceptible to {A1, A2} attackers.

Furthermore, there are variants which perform stealthy port-knocking by encoding the authenticating information into seemingly random looking fields of known protocols, *e.g.* the initial sequence number field of TCP, and the source port number. The advantage of these variants is that they make it difficult for an attacker to suspect port-knocking mechanisms just by observing the traffic. Among these are *SilentKnock* from Vasserman et. al. [7] and *Knock* from Kirsch et. al. [8].

SilentKnock encodes authentication token into the TCP header fields of the TCP SYN packet sent by the client. The firewall intercepts this packet from the kernel and extracts the token. The server then verifies the token and opens the corresponding port if the token is valid. The token is generated with keyed MAC with counters to prevent replay attacks from A1 attackers.

Similar to SilentKnock, the stealth property in Knock is achieved encoding the authentication token into the TCP header fields. Additionally, Knock allows for the client and the server to derive a session key which is then used to authenticate application data, thus preventing an {A1, A2, A3} attacker to take over the connection after successful port-knocking.

While the concept of stealthy port-knocking can be applied to any operating system, the current state of implementations for SilentKnock and Knock are limited to the Linux kernel. This poses a deployment barrier for service providers as they have to require their consumers to run complying software setup on their hosts.

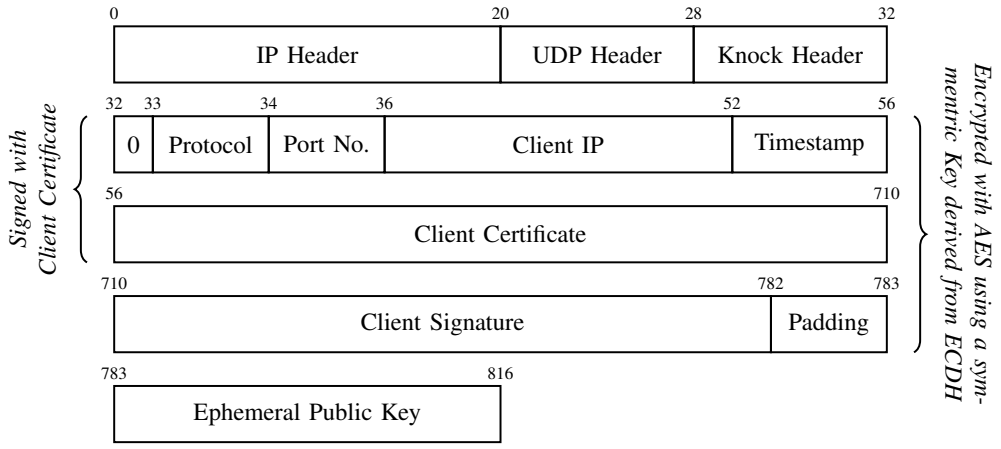


Fig. 1. Packet format of the sKnock authentication packet starting with a IPv4 header. The fields for *Protocol* and *Port No.* provide the port number to be opened in the firewall for the given protocol (TCP or UDP). The *Ephemeral Public Key* contains the public key used by the client and is required for the server to determine the AES key using Elliptic Curve Diffie-Hellman (ECDH).

IV. sKnock

In all of the approaches presented earlier, the client and the firewall depend on a shared secret to authenticate and gain defence against A1, A2 attackers. In the case of B2 attackers, these approaches require the shared secret to be invalidated at the firewall. This brings in the inconvenience and scalability problems discussed in Section I.

sKnock addresses the scalability problem by using certificates: each client gets a certificate which it uses to encrypt and sign the authentication information; the firewall requires the CA certificate to authenticate the client. B2 attackers are defended by limiting the certificate validity to an expiry date and having a certificate revocation list to invalidate certificates before their expiry.

In the reminder of this section we describe sKnock's authentication protocol and give a brief description of its prototype implementation in Python.

A. Protocol

The one-way authentication protocol of sKnock requires the client to send an authentication packet before opening a connection to the remote services behind the firewall. The authentication packet is a UDP packet containing the client's certificate and the port number of the remote service it wants to communicate with on the server. Additionally, it contains the client's IP and timestamp to provide protection against A1, A2 attackers. The format of the packet is shown in Fig. 1. To protect the privacy of the client, this information in the packet is encrypted with an ephemeral key which is derived from the server's public key using Elliptic Curve Diffie-Hellman (ECDH). The ephemeral key is freshly chosen for every authentication packet sent by the client. The client's Diffie-Hellman share required for generating the ephemeral key at the server is also included in the packet.

Since we want to keep the overhead of port-knocking low and also reduce the number of packets involved in the authentication to perform well under packet loss, it is important to fit this payload in one UDP packet. The limiting factors here are the network MTU sizes and the size of the client certificate. A common network MTU of 1500 bytes has eliminated the use of RSA and DSA public keys of lengths 2048 bits and above in the certificates. Fortunately, we were able to use Elliptic Curve Cryptography (ECC) public keys of lengths up to 256 bits offering security equivalent to that of 128 bit AES or 3072 bit RSA keys [9] resulting in a packet size of about 800 bytes. While ECC certificates are not as common as RSA or DSA certificates, this was the only option which gave us the possibility to keep the payload size lower while using x509 certificates.

Reliability against packet loss is achieved by retrying the authentication protocol. For connections to TCP services, the server could be configured to reject connections to closed ports by sending a TCP RST such that a failed authentication protocol will immediately result in TCP connection failure at the client which can then immediately retry. Whereas for UDP, the application requires its own protocol for determining the failure or, alternatively a timeout. The optimal value for the timeout would the sum of round-trip time (RTT) to the firewall, delay for processing the authentication packet and, delay for opening the corresponding port in the firewall.

B. sKnock Certificates

sKnock uses x509v3 certificates with the requirement that the certificates' public and private keys should be 256 bits long and generated using Elliptic Curve Cryptography. The certificates are used only by the clients of a service provider to port-knock the provider's servers. The service provider could either act as a Certificate Authority (CA) for signing these certificates or rely on another party; it is only important to have the same CA certificate configured on the servers.

In addition to authenticating the client, the client certificates also carry authorisation information specifying the protocol,

port pairs the client is authorised to connect to. This information is encoded in the certificates using x509v3 extensions under object identifier for Technical University of Munich, 1.3.6.1.4.1.19518 as *Other Name* in the *Subject Alternative Name* (SAN) extension.

C. sKnock Server

Our prototype implementation of sKnock is developed in Python and works with the Linux *iptables2* firewall. The firewall is dynamically configured to allow traffic from port-knocked connections after the clients are authenticated, while the rest of the traffic is dropped by the firewall, including the sKnock authentication packets. To read the authentication packets we used a raw socket, which in Linux is not subjected to the firewall rules and hence can receive all the traffic reaching the host. As the raw socket receives all the traffic reaching the host, an efficient filtering process is required to filter out valid authentication packets from the rest. This is done by discarding packets which do not meet the following criteria in the order listed:

1) Packet's IPv4 or IPv6 header is valid
2) Packet is UDP and its header is valid
3) Decryption of the packet succeeded
4) Packet has valid sKnock header
   - Byte 32 is 0
   - Timestamp within allowed interval
   - Client IP matches packets source IP
5) Timestamp is within the allowed period
6) Client certificate is valid and not in the revocation list
7) Client certificate authorised for opening requested port
8) Client signature is valid

If the authentication packet passes all of the above checks, the requested port is opened in the firewall for the client sending it.

D. sKnock Client

sKnock client is an implementation of the sKnock protocol for client side applications. The client implementation is available as a library in Python and C. Applications can use the libraries to perform port-knocking at a sKnock firewall. The libraries contain the following functions:

- *knock_new (timeout, retires, verify, server certificate, client certificate, client certificate password)*: creates a new handle with the given certificates. The fields *retries* and *timeout* specify how many times sKnock should retry port-knocking and how long it should wait before determining a failure; applies to TCP connections. The *verify* flag specifies whether the library should test whether the port has been successfully opened or not; applies to TCP connections. *server certificate* and *client certificate* specify the paths to the server and the client certificates. *client_cert_passwd* is the field containing the password for the client certificate. This function returns a handle which is required by the next function.
- *knock_knock (handle, host, port, protocol)*: perform port-knocking by sending the authentication packet to the *host*. *handle* is the value returned from *knock_new()*. *port* and *protocol* specify which port should be opened in the firewall after successful port-knocking.

Applications can port-knock a sKnock firewall by using these two functions before they open a connection to a remote service behind the firewall.

In addition to this, the sKnock client implementation contains a command-line helper program to port-knock the server.

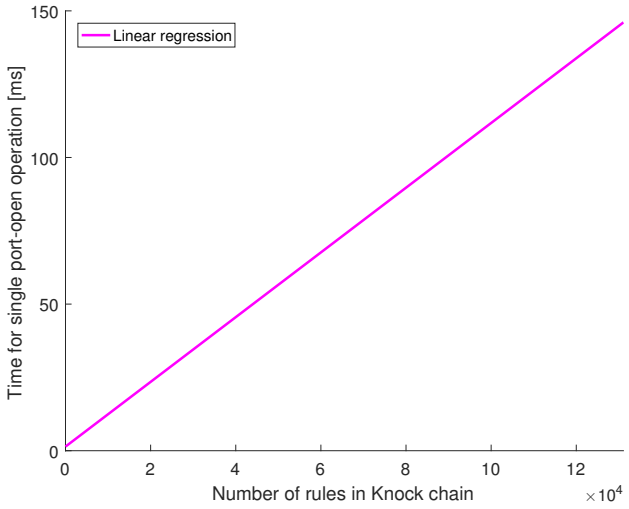


Fig. 2. Delay in adding new rules to the firewall. Rules are added sequentially to open 131070 (65535 TCP + 65535 UDP) for an IPv4 client.

V. Evaluations

Since sKnock is intended as the scalable port-knocking solution, we evaluated its scalability and performance using a variety of tests. As part of this we evaluated the performance of the iptables2 firewall software, our filtering processes for filtering valid authentication packets, and the latency overhead incurred during connection establishment due to port-knocking.

All evaluations were performed on DELL OptiPlex 9020M machines equipped with a quad-core Intel(R) Core(TM) i5-4590T CPU @ 2.00GHz and 16GB of memory. The operating system running on the machine was Ubuntu Linux 12.04.5 LTS. The underlying cryptographic routines were provided by *OpenSSL-1.0.1* and the firewall was *iptables2-1.4.12*. The machines were connected with a *1 Gbps* Ethernet cable when required to form a network.

A. Firewall

We evaluated the scalability limitations of using iptables2 firewall by measuring the time taken to open 131072 ports (65535 TCP + 65535 UDP) for an IPv4 client. The test adds a rule in the firewall to open a port for the given client and measures the time taken for the firewall to add this rule. This is repeated sequentially for each port until all of them are open for the given client. The evaluation data is shown in Fig. 2.

The results show that the delay in adding new rules is linearly proportional to the number of rules present in the firewall.

B. Packet Processing

Since sKnock uses a raw socket, it has to filter valid authentication packets from the rest of the traffic reaching the host. For this we defined a filtering process in Section IV-C. In this evaluation we measured the performance limitation of this filtering process with a static test by pre-generating some valid authentication packets together with some TCP and UDP packets complete with an Ethernet header and random payload. The pre-generated packets are read from a static list, therefore the delay incurred by raw sockets is not accounted.

In our test environment, we decided to run two different measurements: one synthetic worst-case scenario, which simulates that all incoming packets are valid authentication packets and a realistic scenario with 1% of port-knocking traffic while the rest of the packets are irrelevant to sKnock. Additionally the test data for the second scenario contains 5% of packets bigger than the configured minimum authentication packet size. Among these packets the TCP to UDP ratio is set at 5:1 in order to resemble the Internets' traffic patterns as close as possible [10].

The test-run performed to evaluate the worst-case processing power of our implementation yielded a result of roughly *5300 pps* (packets per second), which translates to a processing time of less than *0.19ms* per packet for valid authentication requests.

Our second test case, which is a closer approximation for real-world operation, yielded even higher processing capabilities with an average throughput of over *6100 pps*. Here our implementation achieved processing times of about *0.16ms* per packet.

C. Connection Overhead

In this evaluation we measured the latency incurred while opening connections when using sKnock's port-knocking. We used a simple protocol based on timestamps to measure this latency: the client sends its timestamp to the server as the first packet after port-knocking; the server then responds with its timestamp to the client. The server observes one-way latency while the client observes round-trip latency. The clocks of the client and the server are kept in sync using Precision Time Protocol (PTP) [11].

Since sKnock server requires some processing time to validate a port-knocking request and to add a new rule in the firewall to open the requested port, attempts to connect to a port may fail if the connection packets immediately follow the port-knocking request. Moreover, out-of-order delivery further increases the risk of such failures. To determine the optimal wait period between the authentication packet and the subsequent connection packet, we developed a calibration script. The script tries to open connections with a wait period derived from a start value and retries successful connections with shorter wait periods until a tiny (configurable) fraction of connection open requests fail.

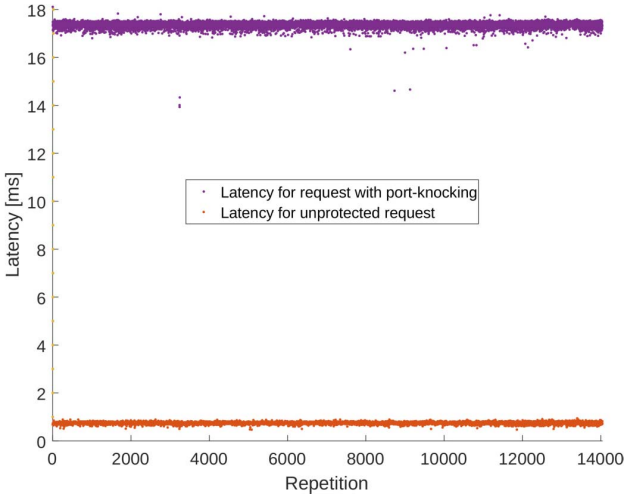


Fig. 3. Latency overhead caused by sKnock (UDP)

TABLE I
LATENCY DUE TO CONNECTION OVERHEAD

	TCP	UDP
Without port-knocking	1.97 ms	0.74 ms
With port-knocking	18.25 ms	17.37 ms
Port-knocking overhead	16.27 ms	16.63 ms

In this evaluation the calibration script yielded an optimal value of *11 ms* for the wait period allowing for a failure rate of *0.2%* with an accuracy of *99%*. To remove noise from the underlying network, we repeated the simple timestamp-based protocol number of times.

The observed latency for UDP run of the simple protocol with an without port-knocking protection can be seen in Fig. 3. The evaluation is also repeated for TCP and the results are summarized in TABLE I.

VI. LIMITATIONS

A major limitation of sKnock lies in the amount of overhead caused by the authentication packet. This is far greater than what is required by other implementations. However, if the connection is long-lived, this overhead will be tiny and since only one packet is used, the delay for port-knocking is kept to a minimum.

The other limitations of sKnock which we are aware of are as following:

A. Incompatibility with NAT

Since the current protocol requires the client to include its IP address into the authentication information, clients using NAT gateways cannot successfully port-knock the firewall.

Deployments seeking compatibility with NAT could ignore the client IP check, but they will then be susceptible to A2 attackers.

B. Vulnerability to Attacks

Any port-knocking scheme employing encryption is further subjected to the *DoS-Knocking* attack [4] where an attacker carries out a DoS attack by sending many legitimately-looking invalid port-knocking requests to keep the firewall busy while legitimate traffic is left in starvation. We agree that this will be also be the case with sKnock, but due to the usage of raw sockets, only new valid legitimate port-knocking requests are denied service as they are to be processed by sKnock which is kept busy with the DoS requests. Since sKnock's current implementation in Python cannot not occupy all of existing processor cores due to the presence of a global interpreter lock any already opened connections are not starved in this case because the firewall is not processing the port-knocking requests and it has already been configured to allow traffic from previously port-knocked clients.

sKnock is susceptible to replay attacks by {A1, A2, A3, B1} attackers. Such an attacker could wait for the port-knocking to succeed and then take over the connection by masquerading as the client. These attackers can be defended by authenticating the connection data which follows port-knocking as done by Knock [8]. This defence is not yet implemented in sKnock.

C. Performance

The current Python implementation of sKnock has limitations in terms of operational efficiency due to Python interpreter being single threaded. While this implicitly gives partial defence towards DoS-Knocking attacks when run on a system with a processor having more than a single core, the throughput could be improved by parallelising request filtering and validation.

Another implementation specific problem with ECC keys is that ECC is relatively new and OpenSSL supports only a limited number of well known curves. Moreover, due to their novel state, there exists no hardware implementation as of this writing to significantly speed up their processing, thus limiting the performance of our implementation.

D. Trust in NIST Curves

sKnock relies on OpenSSL for providing PKI support. As of this writing, OpenSSL does not yet support any of the curve determined as SafeCurves [12] providing 256 bit keys. This led us to use NIST-P 256 curve whose usage may not be secure [12] and hence should be replaced when the required support is available in OpenSSL.

VII. Conclusion & Future Work

When compared to other port-knocking implementations sKnock has high overhead in terms of payload and processing requirements. Furthermore, stealthy port-knocking with sKnock is not possible. As an advantage, sKnock provides easy deployability to service providers as it can be readily integrated into their PKI infrastructure and as it does not require changes to the client's operating system.

The overhead incurred due to using x509 certificates could be reduced by implementing a custom format for encoding the certificates into the authentication information. This will however deny us the usage of well-audited PKI libraries, increase development costs and induce security issues. Alternatively, usage of other formats such as OpenSSH certificates could be explored.

During our evaluations we found that the lack of native parallelism in Python severely limited the performance. Additionally, there was also some overhead involved in converting data structures between the underlying cryptographic libraries and the firewall interface, which were available as C libraries. Therefore, we believe that on a multi-core system the performance of sKnock could be improved by implementing it in a system programming language such as C or Rust.

Another improvement could be made to improve support for UDP data streams. The current prototype requires the client to keep sending authentication packets periodically as long as the application is using the UDP stream to avoid the firewall from timing out the connection and closing the corresponding port. This could be improved by adding application level UDP connection tracking at the server, where the remote service can close the firewall when the connection is no longer required.

Acknowledgment

This work was carried out under the scope of the SafeCloud EU research project funded through EU grant agreement 653884.

References

[1] I. ITU, "Information technology–open systems interconnection–the directory: Publickey and attribute certificate frameworks," *Suíça, Genebra, ago*, 2005.

[2] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "Totp: Time-based one-time password algorithm," Internet Requests for Comments, RFC Editor, RFC 6238, May 2011, http://www.rfc-editor.org/rfc/rfc6238.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6238.txt

[3] H. Krawczyk, M. Bellare, and R. Canetti, "Hmac: Keyed-hashing for message authentication," Internet Requests for Comments, RFC Editor, RFC 2104, February 1997, http://www.rfc-editor.org/rfc/rfc2104.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2104.txt

[4] A. I. Manzanares, J. T. Márquez, J. M. Estevez-Tapiador, and J. C. H. Castro, "Attacks on port knocking authentication mechanism," in *International Conference on Computational Science and Its Applications*. Springer, 2005, pp. 1292–1300.

[5] J. Aycock, M. Jacobson *et al.*, "Improved port knocking with strong authentication," in *21st Annual Computer Security Applications Conference (ACSAC'05)*. IEEE, 2005, pp. 10–pp.

[6] M. Krzywinski, "Port knocking from the inside out," *SysAdmin Magazine*, vol. 12, no. 6, pp. 12–17, 2003.

[7] E. Y. Vasserman, N. Hopper, and J. Tyra, "Silentknock: practical, provably undetectable authentication," *International Journal of Information Security*, vol. 8, no. 2, pp. 121–135, 2009.

[8] J. Kirsch and C. Grothoff, "Knock: Practical and secure stealthy servers," 2014.

[9] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Nist special publication 800-57," *NIST Special Publication*, vol. 800, no. 57, pp. 1–142, 2007.

[10] M. Zhang, M. Dusi, W. John, and C. Chen, "Analysis of udp traffic usage on internet backbone links," in *Applications and the Internet, 2009. SAINT'09. Ninth Annual International Symposium on*. IEEE, 2009, pp. 280–281.

[11] H. Weibel, "High precision clock synchronization according to ieee 1588 implementation and performance issues," *Proc. Embedded World 2005*, 2005.

[12] D. J. Bernstein and T. Lange, "Safecurves: choosing safe curves for elliptic-curve cryptography," *URL: http://safecurves. cr. yp. to*, 2013.

2016 IEEE 35th Symposium on Reliable Distributed Systems Workshops

Toward Access Control Model for Context-aware Services Offloaded to Cloud Computing

Ichiro Satoh
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
ichiro@nii.ac.jp

***Abstract*—Offloading context-aware services with intensive tasks to cloud computing infrastructures is useful. However, we have a problem resulting from differences between context-centric access control models for pervasive computing and subject-based access control models for cloud computing. To solve this problem, this paper proposes a location model for spatially specifying containment relationships of persons, physical entities, spaces, and computers. The model also manages context-centric access control models, and introduces an interface between pervasive computing and cloud computing programs. The interfaces enable context-aware services executed for the latter to access computational resources and information under an access control model for the former. This paper presents the basic notion of the model and its prototype implementation.**

I. Introduction

Context-aware applications on pervasive computing devices can take advantage of the availability of information and have the ability to interact with surrounding devices and embedded digital artifacts. However, the computational resources available from context-aware services in pervasive computing devices tend to be limited. To better cope with the limited resources in pervasive computers, intensive tasks can be offloaded to cloud computing. Since these tasks are executed on servers in cloud computing on behalf of pervasive computers, we need to make sure the tasks that access computational resources from the servers are under appropriate access control. Access control models for context-aware services tend to be context-dependent, but access control models designed for cloud computing are independent of any context.

Most existing access control models tend to be subject-centric in the sense that permissions are provided according to the subjects, e.g., the users. However, we need context-centric ones. In such models, *context* is the first-class principle that explicitly guides both policy specifications and the enforcement process, and it is not possible to define a policy without the explicit specifications of the context that makes the policy valid. At a high level, the term context is defined as any information that is useful for characterizing the state, the activity of an entity, or the world in which this entity operates.

We need to select which program objects should be kept on the pervasive computer during an offloading action. Software for context-aware services should utilize the knowledge created by context providers that is accessible through a communication interface between programs running on pervasive computing devices and cloud computing platforms, because software for such services should be reused for a variety of contexts. The proposed interfaces, called *spatial connector*, have several advantages. They were initially designed as a mechanism to reuse software for defining context-aware services for different contexts [16]. Nevertheless, one of the advantages is to enable context-aware services executed for cloud computing to access computational resources and information under an access control model for pervasive computing, where the model is context-centric instead of subject-based.[1] The interfaces can also connect programs running on pervasive computing devices and cloud computing platforms. This paper focuses on offloading services to cloud computing from pervasive computing rather than from mobile computing. Nevertheless, the approach presented in this paper is useful in context-aware services for mobile computing because context-aware services in mobile computing have several common requirements with the ones in pervasive computing.

II. Related Work

This section briefly highlights several existing access models that have influenced our work with access control models for cloud computing and context-aware access control models.

Conventional access control models can be classified into three types: mandatory access control (MAC), discretionary access control (DAC), and role-based access control (RBAC)[5], [7], [12]. RBAC is an alternative to traditional approaches, i.e., DAC, and MAC. In RBAC, users are assigned roles and roles, are assigned permissions. The principle motivation behind RBAC is the ability to specify and enforce enterprise-specific security policies in a way that maps naturally to an organization's structure.

In cloud computing, access control is one of the most important issues. Cloud computing is often characterized by its multi-tenancy and virtualization features. These features have unique security and access privilege challenges

[1]Our previous paper did not describe any contributions to security, including access control.

978-1-5090-5259-2/16 $31.00 © 2016 IEEE
DOI 10.1109/SRDSW.2016.12

due to the sharing of resources among potential untrusted tenants. RBAC is a key technology for cloud computing platforms and is well-suited for multi-domain architecture; it is applicable in cloud systems that deal with health records, stock trading and pairing, and social networking. The RBAC model is used by many cloud computing platforms, e.g., Microsoft Azure and OpenStack. To achieve multi-tenancy in a single data storage system in cloud computing, several researchers proposed approaches to encrypt data before uploading the data to the cloud by using some cryptographic algorithms so that the data were protected from other tenants. Since access control models in cloud computing platforms are often imposed by the platforms, it is difficult to introduce other models into the platforms.

Many researchers have made several attempts to extend RBAC with the notion of context-awareness. By using the uniform notion of a role to capture both user and environmental attributes, our model enables the definition of context-aware security policies. Roles can also make it easy to define and understand complex security policies; adding environment roles to the model was necessary to support the advanced access control requirements that we are faced with in pervasive computing. However, RBAC approaches assume that permissions are first associated with roles, and subsequently subjects are assigned to roles. In context-aware services, permissions should first be associated with contexts, and subsequently subjects are associated with the contexts they are currently operating in. To solve these problems, Covington et al. [4] allow administrators to specify the environmental context through a new type of role called environmental role to generalize traditional RBAC. Their approach aimed to overcome the inherent subject-centric nature of RBAC. Georgiadis et al. [6] proposed a context-based term based on control by integrating RBAC and team-based access control (TMAC) [18].

Nevertheless, conventional security solutions, including RBAC, seem inadequate to control accesses to resources and interoperation among entities in cases of frequent context changes as we discussed in the first section. In fact, conventional subject-based access control systems exploit user identity or role information to determine the set of user permissions. Permissions are tightly coupled to the identity or role of the subject requesting a resource access, whereas context information can only further limit the applicability of the available permissions. However, context-aware services are often required to be provided in appropriate contexts. For example, electric lights in a room should be controlled by the people in the room instead of by anyone outside the room, even when the people are not registered. Our approach considers context as the primary basis rather than the subjects, e.g., the users. Therefore, access control models in pervasive computing tend to be different. Nevertheless, to offload intensive tasks from pervasive computing to cloud computing, we need a bridge between the two models. As far as we know, there has been no study on combining them.

III. Approach

This section outlines our approach to bridge access control models in pervasive and cloud computing.

A. Example Scenario

There are electric lights in a room. When a user has his/her smart phone and enters the room, he/she wants to turn the lights on from his/her smart phone.

1) Only while he/she is in the room, his/her smart phone should have the capability to control the electric appliances in the room. People who are not in the room should not have the capability to control them.
2) However, if there is an administrator responsible for managing the house containing the room, the administrator should have the capability.

The first scenario needs a context-centric model for access controls, and the second is a typical example of subject-based models. Furthermore, we need to support pervasive devices whose resources tend to be limited.

B. Context-centric access control model

Conventional approaches for access controlling are often based on subject-based access control systems in the sense that they exploit user identity or role information to determine the set of user permissions. Permissions for access controls are tightly coupled to the identity or role of the subject requesting a resource access, whereas contextual information can only further limit the applicability of the available permissions. Therefore, they require administrators to know all contexts and users. However, in pervasive computing environments, context-aware services should be selected and configured according to contextual changes, e.g., the users, their goals, and their locations. If conventional approaches for access control are used in such environments, they may lead to a combinatorial explosion of the number of policies. Furthermore, when offloading task intensive services to cloud computing, such services may be managed in an access control model for cloud computing independently of any context.

To support the context-centric access control model, we introduce an interface, called *spatial connector*, between pervasive computing environments and cloud computing platforms. It is used to bind front-services pervasive computers executed on pervasive computers and backend-services executed at cloud computing platforms. It also bridges context-centric access control and subject-based access control models. We also introduce a symbolic location model to represent context like other existing location models for pervasive computing [1], [10], [13], [14]. Each connector is maintained in such a model according to its contextual existence. The model introduces a containment relationship between spaces, because physical spaces are often organized

in a containment relationship. For example, each floor is contained within at most one building, and each room is contained within at most one floor (Fig. 1). The model spatially binds the positions of entities and spaces with the locations of their virtual counterparts, and when they move in the physical world, it deploys their counterparts at proper locations within it. The model supports a context-centric access control model as a containment relationship between devices and spaces (Fig. 2).

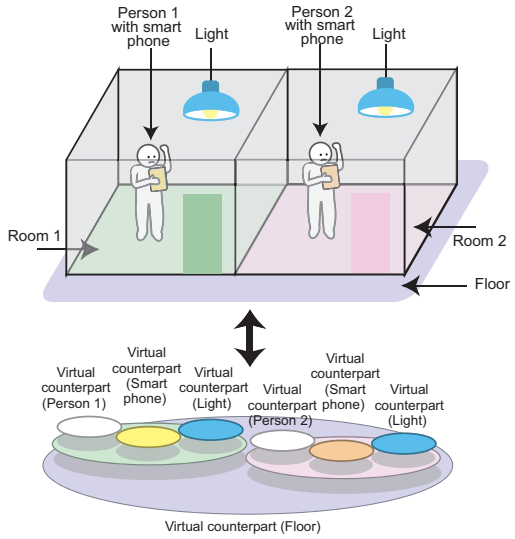


Figure 1. Location model for contexts and access controls

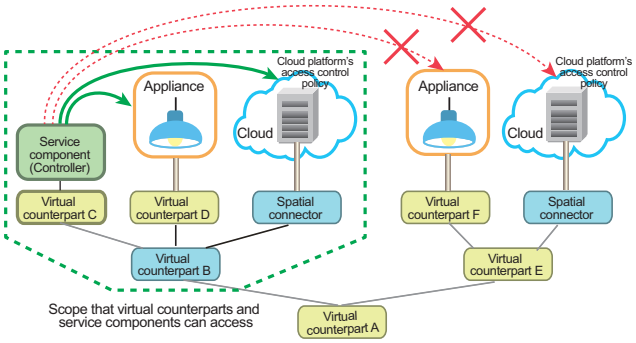


Figure 2. Context-centric access control model

C. Design principles

The approach presented in this paper introduces the following three elements:

- The *location model* is constructed as an acyclic-tree structure of virtual counterparts corresponding to spaces, entities, and computing devices according to their containment relationships. The model is maintained by using location-sensing systems.
- The *virtual counterpart* is a reference to its target, e.g., a user, physical entity, or computing device and contains profiles about the target, e.g., the name of the user, the attributes of the entity and the network address of the device. It is always deployed on pervasive computers close to the current location of its target by using location-sensing systems.
- The *spatial connector* is used to connect at most one virtual counterpart and one or more services, which are provided from pervasive computers or cloud computing platforms. It bridges between access models in pervasive computing and cloud computing. It also enables services, which may be executed on pervasive computers and cloud computing platforms to connect other services provided in the virtual counterparts that contain its virtual counterpart.

For example, when a user enters a room with his/her smart phone, his/her virtual counterpart is migrated to the counterpart corresponding to the room. The spatial connector that is connected to the virtual counterpart corresponding to his/her smart phone enables services that may be executed at pervasive computers and cloud computing platforms to access services provided in the virtual counterparts corresponding to the room. Therefore, the smart phone can use the services via its connector, but devices that are not in the room cannot use the services in the room. Therefore, our model can be used as an access control model in addition to a location model. Our connectors do not distinguish between services executed on pervasive computers and cloud computing platforms. However, access controls for services are context-aware because virtual counterparts corresponding to a target, e.g., the users, physical entities, and computing devices, enable their services to access other services within the spaces that spatially contain their targets.

IV. Design and Implementation

To evaluate the model described in the previous section, we implemented a prototype system that builds on the model. The model itself is independent of any programming languages, but the current implementation uses Java (J2SE or later versions) as an implementation language for components. Our approach maintains a location model for context-aware services and access controls. The model is managed as an acyclic-tree structure of virtual counterparts corresponding to spaces, entities, and computing devices according to their containment relationships in the real world. In addition to the model, the approach has four subsystems: 1) *context manager*, 2) *service components*, 3) *service runtime systems*, and 4) *spatial connectors*. The first is responsible for reflecting changes in the real world and in the locations of users, entities, and pervasive computers. The second is software components for defining services. The third runs on pervasive computers or cloud computing platforms and executes context-aware services. The fourth is responsible for supporting a context-centric access model and connecting virtual counterparts and services.

A. Location model for access controls

The model is organized as an acyclic-tree structure of virtual counterparts and service components, like Unix's file directory, where each counterpart is as a digital representation of a user, physical entity, or computing device. Each counterpart can contain other counterparts or service components, but service components cannot. When a counterpart contains other counterparts, we call the former a *parent* and the latter *children*.

When physical entities, spaces, and computing devices move from location to location in the physical world, the model detects their movements through location-sensing systems and changes the containment relationships between counterparts corresponding to moving entities, their sources, and destinations. Counterpart and service migrations in a hierarchy are performed merely as a transformation of the tree structure of the hierarchy. When a counterpart is moved to another counterpart, a subtree consisting of it and its descendent counterparts and components is moved to a subtree representing the destination (Fig. 3). To support scalability, our model can be managed on multiple computers. When a component is transferred over a network, the runtime system stores the state and the code of the component, including the components embedded within it, into a bit-stream formed in Java's JAR file format that can support digital signatures for authentication.

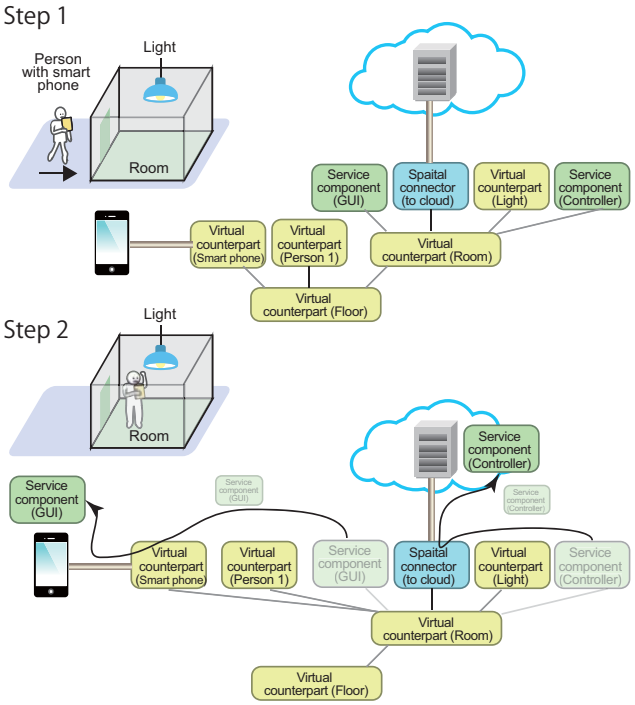


Figure 3. Containment and deployment of Virtual counterparts and service components when person enter in room

1) Virtual Containers: Each counterpart is attached to at most one target, e.g., a user, physical entity, or space. In the current prototype of the approach, each counterpart keeps the identifier of its target or RFID tag attached to the target. Our approach monitors location-sensing systems and the spatial binding of more than one counterpart to each user or physical entity. Virtual counterparts for computers can automatically forward their service components to their target computers by using the component migration mechanism when it receives the service components.

2) Spatial connector: Our approach enables users to explicitly assign a *connector* to each container through its management GUI, even while the container and its inner component are in use. Each connector is activated when its target counterpart migrates to a computer in another cell due to the movement of the counterpart's target in the physical world. We will explain how to deploy containers, including their inner components with connectors according to the deployment of the counterparts that are specified in the connectors as their targets. The deployment of each container is specified in its connector and is managed by runtime systems without any centralized management system. When spatial connectors are connected to computers, they can communicate with them. If computers that spatial connectors are connected to can execute service components, the connectors can forward service components when they receive the components.

In many commercial cloud computing platforms, their access control models are subject-based, e.g., RBAC and RBAC's extensions, in the sense that permissions are provided according to the subjects, e.g., the users. Our model is based on contexts in the sense that contexts are used as the first-class principle for access controls for services and resources. There is a gap between access control models in pervasive computing and cloud computing. Spatial connectors have protocols for the authentication used in cloud platforms. For example, the current implementation supports interfaces for AWS multi-factor authentication (MFA) to use Amazon's cloud computing services, including EC2. When a service in our model signs in to an AWS web-site, they will be prompted for their user name and password as well as for an authentication code from their AWS MFA device. Taken together, these multiple factors provide increased security for your AWS account settings and resources.

3) Context-centric access model: To support context-centric access control, the model allows each spatial connector to bind services according to their locations (Fig. 2).

- When services are executed on a pervasive computer, the components for defining them are contained in the counterpart corresponding to the computer. They can access the resources provided from the computer under the computer's access control policy.
- Each counterpart is contained in another counterpart, where the former corresponds to a target, e.g., a person, physical entity, space, or computer, as a spatial containment relationship, and the latter corresponds to

the space that contains the target. The former can access service components contained in the latter.
- Each counterpart or service component can delegate service components to cloud computing when it has a special connector that links it to the components executed on cloud computing.

We here explain the model with an example scenario. Suppose that a television in a room delegates its services to cloud computing platforms. When a user a smart phone enters the room, his/her counterpart is contained in the counterpart corresponding to the room. Our model enables counterparts to bind spatial connectors to the services contained in the room. Therefore, his/her smart phone can connect the services provided from cloud computing platforms while the phone and television are in the same room.

B. Service Component

Each service component is constructed as a collection of Java objects. When the life-cycle state of a service component is changed, the runtime system issues certain events to the service component and the service component's descendent components. Many pervasive computers have a small amount of memory and slower processors. They cannot always support all services. We introduce an approach to dynamically install upgraded software that is immediately required in computing devices that may be running. Service components are deployable software that can travel from a computing device and computing device archived by using mobile agent technology [15]. Each service component can be dynamically deployed at computing devices. Each service component consists of service methods and is defined as a subclass of a built-in abstract class. Most serializable JavaBeans can be used as service components. When a service component migrates to another computer, not only the program code but also its state is transferred to the destination. For example, if a service component is included in a virtual counterpart corresponding to a user, when the user moves to another location, it is migrated with the counterpart to another counterpart corresponding to the location.

C. Service runtime system

Each service runtime system runs on a pervasive computer or cloud server and is responsible for executing and migrating service components with containers to other service runtime systems running on different computers through a TCP channel. It governs all the components inside it and maintains the life-cycle state of each service component via its component. When the life-cycle state of a service component changes, e.g., when it is created, terminates, or migrates to another runtime system, its current runtime system issues specific events to the component via its component, where the component may mask some events or issue other events. The system has a built-in mechanism for transmitting the bit-stream over the network through an extension of the HTTP protocol. The current system basically uses the Java object serialization package for marshaling components. The package does not support the capturing of stack frames of threads. Instead, when a component is serialized, the system propagates certain events within its embedded components to instruct the component to stop its active threads.

V. Current Status

The current prototype implementation was built on the Java virtual machine (Java VM version 1.8 or later), which concealed differences between the platform architectures of computers. Although the current implementation was not constructed for performance, we evaluated the performance of only basic operations in a distributed system where eight computers (Intel Core i5 2.6 GHz with MacOS X 10.9 and J2SE version 8) were connected through a giga-ethernet. The cost of component migration was 38 ms through a giga-ethernet, and the cost of component migration to Amazon EC2 was 240 ms. The prototype implementation also supported two commercial tracking systems to locate persons, physical entities, and computers. The first is the Spider active RFID tag system, which is a typical example of proximity-based tracking. It provides active RF-tags to users. Each tag has a unique identifier that periodically emits an RF-beacon (every second) that conveys an identifier within a range of 1-20 meters. The second system is the Aeroscout positioning system, which consists of four or more readers located in a room. These readers can measure differences in the arrival timings of WiFi-based RF-pulses emitted from tags and can estimate the positions of the tags from multiple measurements of the distance between the readers and tags; these measurement units correspond to about two meters. Although the current implementation is a prototype system, it has several security mechanisms. The JVM that our runtime systems are executed under could explicitly restrict components so that they could only access specified resources to protect computers from malicious components. To protect components from malicious computers, the runtime system supports authentication mechanisms to migrate components so that all runtime systems can only send components to and only receive them from trusted runtime systems.

A. Evaluation

To prove the utility and validation of the proposed model, we implemented an example of the model. By using this example of our model, we could implement a *disaggregated computing* system, which is an approach to dynamically compose devices, e.g., displays, keyboards, and mice that are not attached to the same computer, into a virtual computer among distributed computers in pervasive computing environments [3], [9], [17]. Our system introduced a context-centric access control and offloaded services to cloud com-

puting into a disaggregated computing system. The system consists of three kinds software components that support *model*, *view*, and *control* behaviors based on a model-view-control (MVC) pattern [11]. The *model* component manages and stores drawing data and should be executed on a server equipped with a powerful processor and a lot of memory. The *view* component displays drawing data on the screen of its current host and should be deployed on computers equipped with large screens. The *control* component forwards drawing data from the pointing device, e.g., the mouse, of its current computer to the first behavior.

We executed these components on three computing entities: smart TV, tablet in a room and a server in cloud computing environment. In the model, the counterpart corresponding to the smart TV and the counterpart corresponding to the tablet were contained in the counterpart corresponding to the room. There was also a spatial connector between the their counterparts and the server. We initially deployed the *view* component at the counterpart corresponding to the smart TV and the *control* component on the counterpart corresponding to the tablet. The two counterparts forwarded the *view* component to the smart TV and *control* component to the tablet. The *model* component was deployed at the counterpart corresponding to the room and then was forwarded to the server through the connector. Since the smart TV's counterpart, the tablet's counterpart and the connector to the server were contained in the same counterpart, they could coordinate with one another. We also added another computer outside the room. The counterpart of the computer was not contained for the counterpart corresponding to the room. Therefore, the computer could not access the three components, i.e., the *model*, *view*, *control* components, and the three computers, i.e., the smart TV, tablet, and servers.

VI. Conclusion

We constructed a location model for context-aware access controls in addition to context-aware services, which are executed on pervasive computing side or cloud computing side. The model introduced contexts as the first-class principle for access controls to services and resources. The model was constructed in a symbolic location model for representing spatially containment relationships of persons, physical entities, spaces, and computers. When a user entered a room, the approach enabled him/her to access services provided in the room, but the people outside the room could not access the services. Also, the approach could offload context-aware services with intensive tasks, which need much resources, to cloud computing platforms from pervasive computing devices.

References

[1] M. Beigl, T. Zimmer, C. Decker, A Location Model for Communicating and Processing of Context, Personal and Ubiquitous Computing, vol. 6 Issue 5-6, pp. 341-357, 2002

[2] P. Bellavista, R. Montanari, D. Tibaldi: COSMOS: A Context-centric Access Control Middleware for Mobile Environments, In Proceedings of Mobile Agents for Telecommunication Applications (MATA'2003), LNCS, Vol.2881, pp.77-88, 2003.

[3] B. L. Brumitt, B. Meyers, J. Krumm, A. Kern, S. Shafer: EasyLiving: Technologies for Intelligent Environments, Proceedings of International Symposium on Handheld and Ubiquitous Computing, pp. 12-27, 2000.

[4] M.J. Covington, W. Long, S. Srinivasan, A.K. Dev, M. Ahamad, and G. D. Abowd: Securing context-aware applications using environment roles, In Proceedings of 6th ACM symposium on Access Control Models and Technologies (SACMAT'2001). pp.10-20. 2001.

[5] D.F. Ferraiolo, J.F. Barkley, and D. Richard Kuhn: A role based access control model and reference implementation within a corporate intranet. ACM Transactions on Information Systems Security, Vol.2, No.1, pp.34-64, 1999.

[6] C.K. Georgiadis, I. Mavridis, G. Pangalos, and R.K. Thomas: Flexible team-based access control using contexts, In Proceedings of 6th ACM symposium on Access control models and technologies (SACMAT'01), pp.21-27, 2001.

[7] L. Giuri and P. Iglio: Role templates for content-based access control, In Proceedings of 2nd ACM Workshop on Role Based Access Control (RBAC'97), pp. 153159, 1997.

[8] R. J. Hulsebosch, A. H. Salden, M. S. Bargh, P. W. G. Ebben, and J. Reitsma. 2005: Context sensitive access control, In Proceedings of 10th ACM symposium on Access control models and technologies (SACMAT '05). pp.111-119, 2005.

[9] B. Johanson, G. Hutchins, T. Winograd, M. Stone: PointRight: experience with flexible input redirection in interactive workspaces, in Proceedings of 15th ACM symposium on User interface software and technology, pp.227-234, 2002.

[10] U. Leonhardt, and J. Magee, Towards a General Location Service for Mobile Environments, Proceedings of IEEE Workshop on Services in Distributed and Networked Environments, pp. 43-50, IEEE Computer Society, 1996.

[11] G. E. Krasner and S. T. Pope, A Cookbook for Using the Model-View-Controller User Interface Paradigma in Smalltalk-80, Journal of Object Oriented Programming, vol.1 No.3, pp. 26-49, 1988.

[12] R.S. Sandhu, et al.: Role-Based Access Control Models, IEEE Computer, Vol. 29, No.2, 1996

[13] I. Satoh: A Location Model for Pervasive Computing Environments, Proceedings of IEEE 3rd International Conference on Pervasive Computing and Communications (PerCom'05), pp,215-224, IEEE Computer Society, March 2005.

[14] Ichiro Satoh: A location model for smart environments Pervasive and Mobile Computing, Vol.3, No.2, pp.158-179, Elsevier, 2007.

[15] I. Satoh: Mobile Agents, in Handbook of Ambient Intelligence and Smart Environments, pp.771-791, Springer (2010).

[16] Ichiro Satoh: Spatial Connector - Loosely Binding Contextual Changes and Non-Context-Aware Services, Proceedings of 8th International Conference on Software Technologies (ICSOFT'2013), pp.50–57, 2013.

[17] P. Tandler: The BEACH application model and software framework for synchronous collaboration in ubiquitous computing environments, Journal of Systems and Software - Special issue: Ubiquitous computing archive Vol.69, No.3, pp.267-296, 2004.

[18] R.K. Thomas: Team-Based Access Control (TMAC): A Primitive for Applying Role-Based Access Controls in Collaborative Environments, in Proceedings of 3nd ACM workshop on Role-based Access Control, pp.13-19, 1997.

Comfort and Security Perception of Biometrics in Mobile Phones with Widespread Sensors

Javier Guerra-Casanova
Technical University of Madrid
Campus de Montegancedo
28223 Pozuelo de Alarcón,
Madrid, Spain
Email: javier.guerra@upm.es

Belén Ríos-Sánchez
and Miguel Viana-Matesanz
and Gonzalo Bailador
and Carmen Sánchez-Ávila
Group of Biometrics, Biosignals and Security
Research Centre for Smart Buildings
and Energy Efficiency
Technical University of Madrid
Campus de Montegancedo
28223 Pozuelo de Alarcón,
Madrid, Spain
Email: {brios, mviana, gbailador, csanchez}@cedint.upm.es

María José Melcón de Giles
Technical University of Madrid
Campus de Montegancedo
28223 Pozuelo de Alarcón,
Madrid, Spain
Email: mariajose.melcon@upm.es

***Abstract*—Comfort and security perception are two key factors to provide an adequate biometric solution. This article presents the results of an online survey about these characteristics in four different biometric modes implemented in mobile phones with widespread sensors. Additionally, it presents the main concerns that the use of these biometric modes generates in people, which provides a roadmap of additional issues that should be improved to create satisfactory biometric techniques.**

***Keywords*—Comfort, security, biometrics, perception, mobile phones, face, hand, voice, in-air signature.**

I. Introduction

With the increasing functionalities and services that are accessible via mobile telephones, there is a strong argument that the user authentication level on mobile devices should be extended beyond the Personal Identification Number (PIN) that has traditionally been used. One of the principal alternatives where the industry and research institutions have focused their attention on is the usage of biometric techniques on mobile phones as a method to verify the identity of a person accessing a service.

Usually, biometric research intends to improve the recognition algorithms to increase the performance of the biometric authentication. However, the user perception in terms of security and acceptability is at least as important as the performance in order to increase the use of this technology. To the knowledge of the authors, the first study on biometric acceptability perception was published in 1995 [1]. Contrary to expectations, it was found that behavioural-based biometric systems were perceived as less acceptable than those related to physiological traits. A deeper study with 391 participants was carried out in 2007 [2]. In this study, 25% of participants had not heard about biometrics. Their main concerns were cleanliness, safety and the storage of their biometric templates. More recently, a study of users acceptance and satisfaction of biometric systems was presented in [3], concluding that the perceived security does not coincide with the performance. Furthermore, in [4] the user preferences and graded security on mobile phones were assessed for different biometric modes. In addition, [5] presents a study of usability in terms of time and memory recall for voice, face and 2D gesture biometrics. Finally, [6] presents a study about biometric authentication on iPhone and Android smartphones in terms of usability, perceptions and influences on adoption.

Nowadays, one of the most known mobile biometric modes is fingerprint. However, this technology requires a specific mobile phone with a fingerprint sensor incorporated. In this study, we will focus on the acceptability of biometric techniques that can be used with widespread sensors in mobile devices (without specific hardware). In this way, we will evaluate the acceptability in terms of comfort and security perception of the following biometric modes: Face [7], Hand [8], Voice [9] and Gesture Recognition (also named In-Air Signature) [10]. All these biometric modes use widespread sensors such as cameras, microphones or accelerometers that most of the current mobile phones do integrate.

This article is composed by the following sections: Section II describes the survey proposed to assess the security and comfort perception as well as the main concerns about the evaluated biometric modes evaluated. Next, Section III shows the answer results grouped by biometric mode and by the expertise in biometrics of the participant. Finally, the main conclusions of the study are presented in Section IV.

II. Survey description

The main objective of this online survey is to study two different aspects of the candidate biometric modes:

- Comfort perception: If users feel comfortable using this technology.
- Security perception: If users feel that the security of this biometric mode is strong enough.

978-1-5090-5259-2/16 $31.00 © 2016 IEEE
DOI 10.1109/SRDSW.2016.13

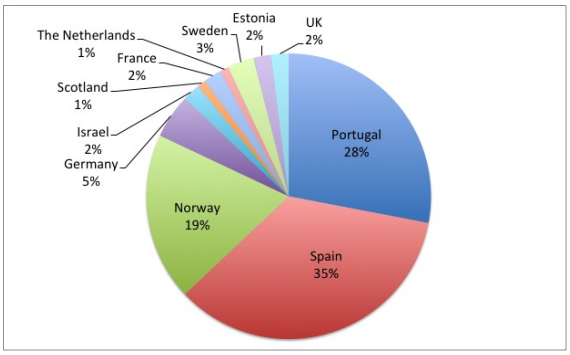


Fig. 1. Participant Country distribution

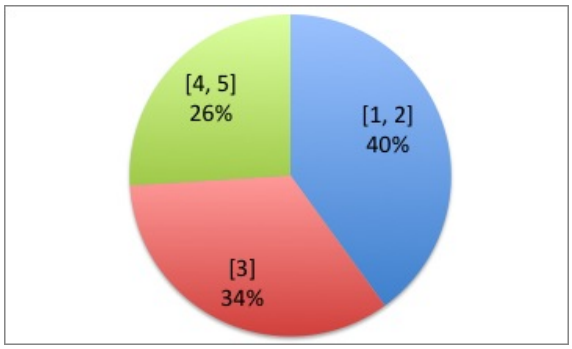


Fig. 2. Participant Biometric Experience distribution from Very High [5] to Very Low [1]

Both aspects are assessed for each biometric mode between a mark of 1(very low) and 5 (very high). Additionally, an open text field was created for each biometric technique asking participants to write their main concerns about using each specific technique. At the beginning a short video of a person using this biometric mode is displayed, and after that the questions about comfort, security and concerns are presented.

Some effort was put into providing little but clear information about each biometric mode, looking specifically for security and comfort perception. We did not include long explanations of the algorithms nor the authentication procedure. This decision was taken in order to make the survey more understandable and to be sure that the information was clear and short.

All the answers were stored anonymously together with some personal information in order to be able to distinguish between specific groups of people. In particular, the following information was asked: age, gender, country, technological experience and biometric expertise.

This survey has been answered by 100 participants from different countries in Europe. There are mainly three countries that have contributed with a significant number of answers: Spain, Norway and Portugal but people from 11 countries have participated in this study. The distribution of the participants by country is shown in Figure 1.

Regarding the distribution of gender and age, half of the answers come from a population aged between 31 and 50 years old, 14% over 50 years old and 36% from people between 18 and 30 years old. In addition, the percentage of male answers double the ones by females (67-33%).

Finally, the survey incorporated a question to separate target groups of the population in terms of their biometric experience. Figure 2 refers to the proportion of the answers where the participant declared a very high or high ([5-4]), normal ([3]) or low or very low ([2-1]) biometric knowledge. Since a significant number of answers to this survey come from technological people and researchers, we have decided to make the distinction between the answers of these very technological people with experience in security in biometrics and the rest of population. Of course, the opinion of security experts is very valuable and must be considered, but it should be treated differently than the opinion of other people that are not so much in contact with this technology. Indeed, it can be observed from their answers that they have different concerns about each biometric mode.

Actually, the survey did not offer any information about the algorithms deployed nor the approaches. There was only one video showing the process of an authentication procedure and the questions were oriented to perception of comfort and security. In spite of this, it is quite noticeable that a significant proportion of technological people automatically thought about specific issues on the algorithms instead of the questions raised in the survey.

III. Results

Below the results of each technique are presented separately. Comfort and security results are provided graphically and then analysed. In both figures, the cumulative sum of answers from very high to very low has been presented in order to be able to check the proportion of people that provides at least certain value to each question. In addition, those concerns which appeared most frequently are provided.

A. *Face recognition survey results*

Regarding face recognition, the distribution of the answers to the questions about comfort and security is presented in Figures 3 and 4.

In terms of comfort perception, around 70% of people considers that face recognition is at least comfortable. In this case, we can not discern much difference between the responses of the different groups of biometric expertise.

Regarding security perception, around 65% of the people assumes that face recognition is secure enough. In this case, we can distinguish a lot of differences between the people who have expertise in biometrics and those who do not. The security experts know the limitations of this technology and without a deeper explanation of the algorithm and the approach deployed, they have many concerns about the biometric performance in terms of security. As the main scope of this study was to analyse the perception of the common users, in our opinion it is much more significant for this purpose the opinion of non-experts.

However, most of the concerns that security experts provide a significant roadmap of the approaches to be considered in order to increase the security of this biometric mode.

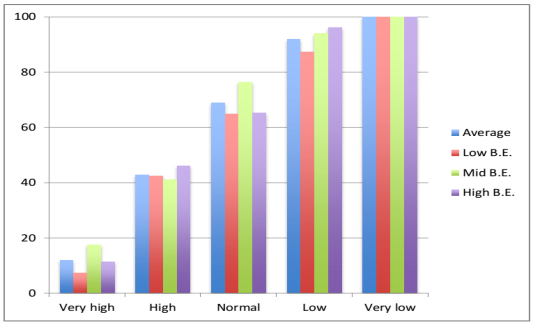


Fig. 3. Face comfort perception

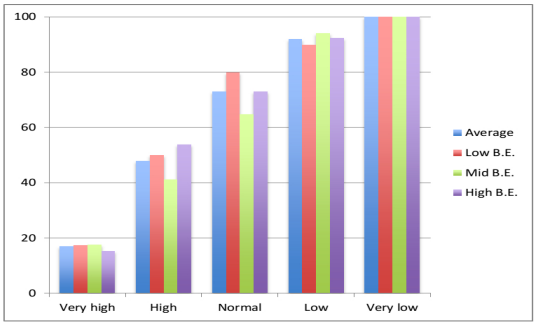


Fig. 5. Hand comfort perception

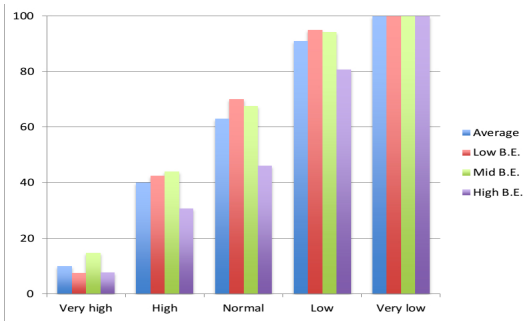


Fig. 4. Face security perception

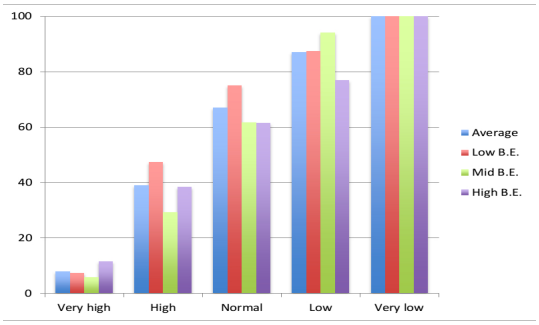


Fig. 6. Hand security perception

The top 5 most commented concerns regarding face recognition were:

- Performance in different conditions: 32% of the participants declared some concerns about the performance of face recognition in different lighting conditions and also whether the algorithm would work correctly when the face changes along the time.
- Forging the system with photographs or videos: 23% of the participants suggested that the system could be easily forged with the photograph of the face of someone else or directly that a liveness detection method should be incorporated in order to avoid the use of fake images.
- None: 16% of the participants did not indicate any concerns about face recognition biometrics.
- Strange use: 15% of the participants insinuated that the use of the frontal camera could be awkward or not appropriate in some conditions. Most of this people associated face recognition with "taking a selfie".
- Privacy: 10% of the participants were concerned about the use of photographs of their faces and about the fact that their face could be stored anywhere out of their control.

B. Hand recognition survey results

In terms of hand recognition, the comfort and security perception distribution are presented in Figures 5 and 6. It is noticeable that no information about the algorithm nor the approach was presented, only the video of the person making a photo of their hand. Accordingly, these results are suitable for the geometry and the palm techniques, although the approach followed was one of the main concerns of the people with high biometric expertise.

In terms of comfort perception, more than 75% of participants have considered hand recognition as, at least, comfortable.

In terms of security perception, around the 70% of the participants assessed hand biometric as normal security. In this case, it is noticeable that people without biometric expertise considered hand biometrics as much more secure that the rest of target groups.

Furthermore, the participants expressed their main concerns about hand recognition. The top 5 most commented concerns are presented below:

- Performance in different conditions: 27% of the participants declared some concerns about the performance of hand recognition in dark conditions and carrying gloves.
- None: 19% of the participants did not state any concerns about hand recognition biometrics.
- Strange use: 18% of the participants found using hand recognition not very natural. Some of them also declared that the requirement of using both hands could make this biometric mode uncomfortable.
- Forging the system with photographs or videos: 14% of the participants argued that the system could be

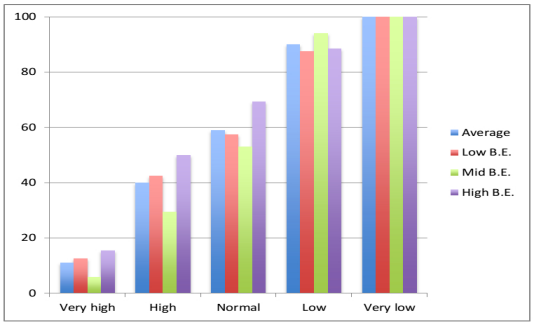


Fig. 7. Voice comfort perception

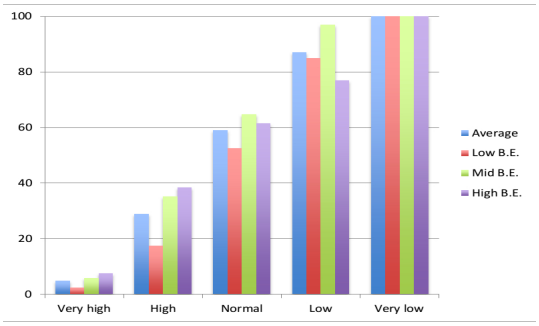


Fig. 8. Voice security perception

forged with a photograph or a video of the hand of the user and indicated the need of a liveness detection algorithm to avoid this.

- Privacy: 4% of the participants were concerned about the privacy requirements and how and where the hand images were stored.

In terms of the strange use, it is also possible that the video presented to the participants was quite overacted in order to be more understandable (even more than in face recognition). Accordingly, the concerns on this idea might be avoided by just using the hand recognition more naturally.

C. Voice recognition survey results

Regarding voice biometrics, Figure 7 and 8 present the results of voice recognition questions. The sentence "my word is my password" was selected to represent the biometric access using a short sentence pronounced aloud.

According to Figure 7, around 60% of the participants declared a normal comfort perception of voice recognition. It is noticeable that the biometric expert group declared a much more comfort grade than the rest of the groups. Probably, they have some knowledge about any of the many initiatives trying to use voice recognition as a biometric mode when an authentication is required during a phone call and they have assumed this particular use case where voice biometrics is very interesting and comfortable, since it works at the simultaneously and transparently to the user while making the phone call. However, the rest of population thought about a use case where they had just to pronounce a short sentence in order to grant access to something in the phone. In this case, voice recognition is not so comfortable since there are many situations where the authentication should be granted in silence.

Regarding security perception, the average result of a good security perception is around 60%. In this question, we notice a similar behaviour as in the previous one, where people with a low biometric expertise do not think that voice biometrics can be good enough. Participants with high biometric expertise considered the phone-call use case where voice biometrics provides an extra and useful evidence.

On the other hand, the participants stated their main concerns about voice recognition. In this case, it is noticeable that many people declared several concerns when wondering about the use of voice biometrics. Only the 4 most significant are presented:

- Problems while using: 41% of the participants found using voice biometrics very inappropriate in many common situations where it is not possible to speak aloud.
- Performance in different conditions: 40% of the participants declared concerns about the performance of voice recognition, specially in noisy environments and when people suffer illnesses that affect the voice.
- Forging the system with recordings: 29% of the participants insinuated that it was quite simple to perform a reply attack on voice biometrics, since anyone can record you saying aloud your password and using directly to attack the system.
- None: Only 8% of the participants did not declare any concerns about voice recognition biometrics.

D. In-Air Signature biometrics

Finally, the last biometric technique assessed is the one based on gesture recognition. As explained before, this technique authenticates the user by creating a personal gesture, as a signature in the air. Of course, this technique is not so known as the others. Figures 9 and 10 display the results of the answers of the questions about comfort and security perception of the technique.

In this case, more than 60% of the people declared that In-Air Signature biometrics is comfortable. However, almost 80% of biometric experts position this technique with a 80% of normal comfort grade. This result is quite surprising and can be derived from the fact that many of the biometric experts that answered the survey have had some contact with this technology and have tested it. However, in our opinion, the answers from other groups are more significant, since they present the opinion of general users who faced for the first time the possibility of using gesture recognition for authentication.

In terms of security perception, the results are quite scattered. People with low biometric expertise does not feel that this technique can be very secure. However, biometric experts and people with some experience in this field considered that it has better results, probably because they are aware of some of the research made in this area.

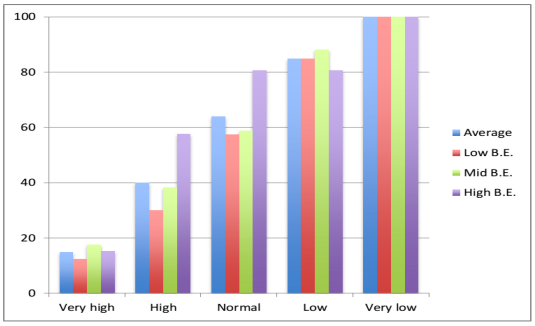


Fig. 9. In-Air Signature comfort perception

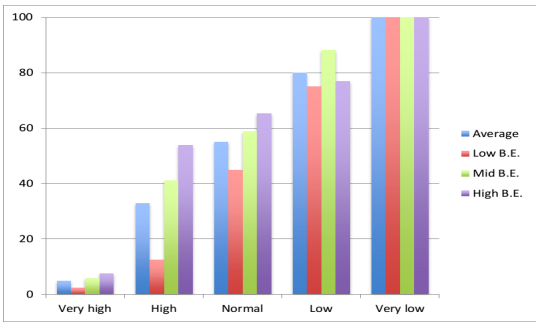


Fig. 10. In-Air Signature security perception

Furthermore, the Top 5 main concerns about gesture recognition are:

- Imitation: 25% of the participants were concerned about the possibility that a person watching the gesture performance of someone else could imitate it successfully to forge the system.
- Strange use: 22% of the participants found some concerns about the use of gesture recognition. Some of them declared that they would feel strange making gestures and others found specific crowded situations where gestures could not be done.
- Performance: 18% of the participants have doubts about the performance of the biometric mode, since some of them felt difficult to repeat their gesture accurately enough.
- None: 16% of the participants did not state any concerns about gesture recognition biometrics.
- Remember: 6% of the participants were concerned about being able to remember their gesture in the future.

E. Comparison of results

A summary of the previous results is presented below.

The most comfortable biometric mode in the survey is the hand recognition, followed by the face. The gesture and voice are the least comfortable and quite similar. Analyzing the results for the different biometric expertise groups separately, it is noticeable that hand recognition comfort perception outstands for the group of non-biometric experts. Additionally, gesture recognition is presented as the most comfortable technique for the biometric experts. This result should be considered carefully since there is a not insignificant proportion of people in this group which have direct contact with one research group that has deployed this technology.

On the other hand, the average security perception results show that the face and hand recognition stand out from voice and gesture recognition.

Additionally, the results of security perception separated by biometric expertise groups state the following conclusions:

- The high-expertise participants perceive face recognition as the one with lowest security, mainly because of the need of a liveness detection system in order to avoid other people using photographs to forge the system. This behaviour is not followed by the rest of groups that categorize face recognition as one of the best biometric modes of the survey.
- Hand and voice present quite similar results in the high-expertise group, but the less biometric experience the most security perception of hand recognition in relation to voice.
- Gesture recognition shows good security perception for the biometric experts. However people without biometric experience do not have the same perception.

For the correct interpretation of the previous results it should be taken into account the fact that people may understand differently what is very secure and/or very comfortable. Accordingly, there could be people that have answered these questions comparing the security required to different options, like the DNA authentication (which is the most precise one) or a very light authentication. Additionally, there could be people that do not feel that any biometrics is secure or comfortable enough.

IV. Conclusion

The analysis of the security perception for the different biometric expertise groups leads to the following statements:

- Most participants with high and medium biometric expertise consider hand recognition as secure as face recognition, whereas the participants in the low biometric expertise group do not provide the same score to both techniques but provide very close marks.
- There is a significant part of high biometric experts that assess voice biometric with a lower security perception than face biometrics.
- In terms of gesture recognition, there is a significant part of high biometric experts that perceive gesture recognition at least as secure as face recognition. However, most people without expertise in biometrics perceived gesture recognition less secure than face.

For this analysis, it can be concluded that face and hand recognition are much more comfortable and show a higher security perception than voice and gesture recognition.

However, as stated in the analysis of each technique, there are many concerns that must be taken into consideration in order to increase the security and comfort of both biometric modes. The main concerns to be focused are:

- Liveness detection (Hand and Face biometrics)
- Performance in different environments (Hand and Face biometrics)
- Performance and resistance to imitations (Gesture biometrics)

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement Nº 610713 (Personalised Centralized Authentication System (PCAS) project).

REFERENCES

[1] F. Deane, K. Barrelle, R. Henderson, and D. Mahar, "Perceived acceptability of biometric security systems," *Computers & Security*, vol. 14, no. 3, pp. 225 – 231, 1995.

[2] S. Elliott, S. Massie, and M. Sutton, "The perception of biometric technology: A survey," in *Automatic Identification Advanced Technologies, 2007 IEEE Workshop on*, June 2007, pp. 259–264.

[3] M. El-Abed, R. Giot, B. Hemery, and C. Rosenberger, "A study of users' acceptance and satisfaction of biometric systems," in *Security Technology (ICCST), 2010 IEEE International Carnahan Conference on*, Oct 2010, pp. 170–178.

[4] H. Sieger, N. Kirschnick, and S. Möller, "User preferences for biometric authentication methods and graded security on mobile phones," in *Symposium on Usability, Privacy, and Security (SOUPS)*, 2010.

[5] S. Trewin, C. Swart, L. Koved, J. Martino, K. Singh, and S. Ben-David, "Biometric authentication on a mobile device: A study of user effort, error and task disruption," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ser. ACSAC '12. New York, NY, USA: ACM, 2012, pp. 159–168. [Online]. Available: http://doi.acm.org/10.1145/2420950.2420976

[6] R. Bhagavatula, B. Ur, K. Iacovino, S. M. Kywe, L. F. Cranor, and M. Savvides, "Biometric authentication on iphone and android: Usability, perceptions, and influences on adoption," in *In Proc. USEC*, 2015.

[7] G. Dave, X. Chao, and K. Sriadibhatla, "Face recognition in mobile phones," *Department of Electrical Engineering Stanford University, USA*, 2010.

[8] D. de Santos-Sierra, M. Arriaga-Gomez, G. Bailador, and C. Sanchez-Avila, "Low computational cost multilayer graph-based segmentation algorithms for hand recognition on mobile phones," in *Security Technology (ICCST), 2014 International Carnahan Conference on*, Oct 2014, pp. 1–5.

[9] H. Aronowitz, R. Hoory, J. W. Pelecanos, and D. Nahamoo, "New developments in voice biometrics for user authentication." in *INTERSPEECH*, 2011, pp. 17–20.

[10] J. Guerra-Casanova, C. Sánchez-Ávila, G. Bailador, and A. de Santos Sierra, "Authentication in mobile devices through hand gesture recognition," *International Journal of Information Security*, vol. 11, no. 2, pp. 65–83, 2012.

2016 IEEE 35th Symposium on Reliable Distributed Systems Workshops

A configurable multibiometric system for authentication at different security levels using mobile devices

Belén Ríos-Sánchez
and Miguel Viana-Matesanz
and Carmen Sánchez-Ávila
Group of Biometrics, Biosignals and Security
Research Centre for Smart Buildings
and Energy Efficiency (CeDInt)
Technical University of Madrid
Campus de Montegancedo,
28223 Pozuelo de Alarcón, Madrid, Spain
Email: brios, mviana, csanchez@cedint.upm.es

María José Melcón de Giles
Technical University of Madrid
Campus de Montegancedo
28223 Pozuelo de Alarcón,
Madrid, Spain
Email: mariajose.melcon@upm.es

***Abstract*—This work presents a configurable multibiometric system oriented to mobile devices which combines face, hand and in-air signature biometrics to provide three different levels of security. The number of traits involved in the authentication increases with the security strength, allowing the balance between comfort and accuracy according to the security requirements of the final application. In addition, the security of the system is enhanced by incorporating anti-coercion and aliveness detection mechanisms. To decide which biometric mode should be requested at each security level, an evaluation of the biometrics has been performed in terms of performance and users acceptability.**

Keywords*—*Multibiometrics, Security Levels, Hand, Face, In-Air Signature, User Acceptability, Coercion, Aliveness.

I. Introduction

The expansion of mobile devices has derived into a great number of applications and services which requires to verify the identity of the users, causing a growing trend toward touchless biometrics. In the case of image-based biometrics, in addition to typical trials derived from the properties of each modality, new challenges arise by contact-less images have to be addresed. These challenges include blurriness or intra-class variations due to changes in pose and lighting conditions. It has been demonstrated that the fusion of biometric information coming from different traits permits overcoming those drawbacks intrinsic to each biometric mode as well as those coming from the capturing process and the environmental conditions, increasing the overall accuracy, improving the performance and reducing the vulnerability of the biometric system [1].

As a result of the tendency toward contactless biometrics together with the raising of biometrics fusion, some authors have started working on multimodal biometrics for mobile devices. In [2] Vildjiounaite et al. proposed an unobtrusive method of user authentication for mobile devices which fuses walking style (gait) and voice recognition. The system presented by Kim et al. [4] integrates face, teeth and voice biometrics in mobile devices. Kang and Park proposed a multiunit iris authentication based on quality assessment and score level fusion for mobile phones [5]. McCool et al. [3] presented the implementation of a face and speaker recognition system on a Nokia N900 mobile phone. Trewin et al. examined voice, face and gesture recognition as well as face-voice and gesture-voice combinations to explore the relative demands on user time, effort, error and task disruption [6].

Despite the accuracy of the biometric recognition increases with the number of involved traits, it also requires more collaboration on the part of the user and takes more time, decreasing the comfort of the authentication process. Nevertheless, given the great variety of services available through mobile devices it has been observed that they do not have the same necessities regarding security. Accordingly, in this work a configurable multibiometric system oriented to mobile devices is presented, which permits the authentication of the users at three different security levels.

The proposed system integrates physical and behavioural information coming from face, hand and in-air signature biometrics. As the security level becomes stronger, the number of traits involved in the authentication process increase, allowing the balance between comfort and accuracy according to the security requirements of each application. In addition, the security of the system is reinforced by anti-coercion and aliveness detection mechanisms. The decision of which biometric are associated to each security level is based on an evaluation of the biometrics in terms of performance and users acceptability. To evaluate the performance of each biometric mode and the different fusion combinations (two-by-two and as a whole) a multimodal dataset was used, which was recorded under realistic conditions.

This paper is organized as follows. First, a description of the system is presented in Section II. Then, the evaluation of the system in terms of performance and user acceptability is provided in Section III together with the optimum configura-

978-1-5090-5259-2/16 $31.00 © 2016 IEEE
DOI 10.1109/SRDSW.2016.14

tion of the system. Finally, conclusions are presented in Section IV.

II. System Description

A. Multimodal Biometrics

The proposed system integrates face, hand and in-air gesture biometrics to increase the robustness of the authentication. The information is fused at score level using the weighted minimum rule in a process that could involve all the biometrics or two of them.

1) Face Biometrics: The face recognition algorithm is based on the *eigenfaces* scheme [7] over a holistic facial pattern. This way, during the feature extraction procedure the face images are projected into a feature space which is created from those combinations of image properties which allow the best discrimination among different people. To select these features Principal Component Analysis (PCA) is performed over a collection of face images coming from three public face databases: The ORL Database of Faces[8], the Face Recognition Data from the University of Essex[1] and The Color FERET Database [9]. This collection contains frontal facial images from 1036 different individuals with variations in gender, age, race and background. Although the images don't present severe shadows, rotations, occlusions, poses nor defined expressions, they have been captured in different environmental conditions. For this reason, the images have been previously preprocessed using a three-step-method in which illumination defects and shadows are corrected, the face is located within the image and the face region is centred, aligned, scaled and cropped to a standard size so that all the face images are 92×112 and display the center of the eyes on the same position. The dissimilarity between feature vectors is obtained by means of a measure based on dissimilarity measure coefficients.

2) Hand Biometric Module: Taking advantage of the multibiometric nature of hand, features related to two different hand-related traits are extracted: hand geometry and palmprint. To this end, it is necessary to pre-process the images in such a way that the hand is separated from the rest of the image and the fingertips and inter-finger valleys are detected. This way, the image is normalized and converted to the LAB color space. The B channel which contains the bluish hue missing in the skin is binarized by thresholding, smoothed by a median gaussian filter and added to the L channel. Assuming that hand is centred, a flooding segmentation is performed from the center of the image. Then, fingers are isolated by means of morphological operations on the binary image and identified according two classification criteria: the aspect ratio and the area of the region. Given the five regions corresponding to the fingers, tips are obtained as the furthest points of each region to the hand centroid. Finally, valleys are detected by considering the boundary hand contour region between two finger pairs and selecting the closest point to hand centroid. Once the images have been preprocessed, the biometric features are extracted.

- Biometric information contained within the hand geometry is reduced to few features constituting the hand geometry feature vector. For each finger, its length is calculated from the tip to the center of the line formed by the two corresponding valleys lying at both sides of the given fingers. In addition, the widths of each finger are calculated at four different heights and normalized by the finger length. Finally, some distances related to the palm are obtained to complete the biometric template as shown in Fig. 1.

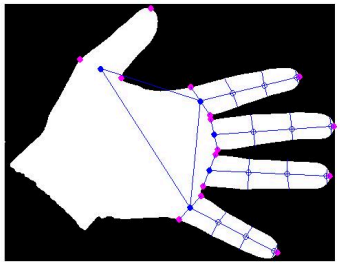
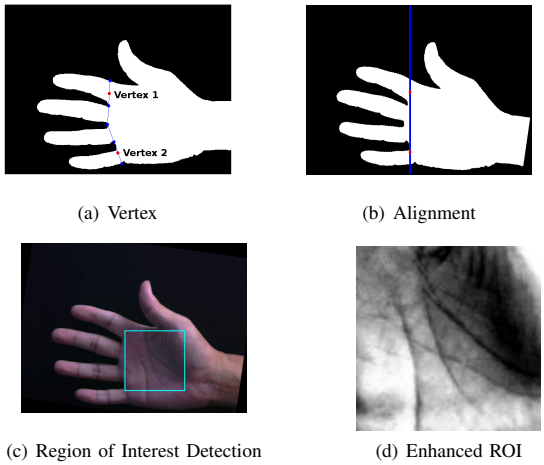


Fig. 1. Hand geometry features.



(a) Vertex (b) Alignment

(c) Region of Interest Detection (d) Enhanced ROI

Fig. 2. Extraction, alignment and enhancement of the region of interest containing the biometric features of the palmprint.

- To obtain the palmprint features it is necessary to locate the Region of Interest (ROI) which contains relevant information related to the palm. Also it is required to align it into a coordinate system common to all the images to avoid problems related to rotation and translation derived from the contactless nature of the images. The ROI is extracted using a similar method to that proposed in [10]. Using valleys as a reference, two vertex of the squared region are calculated (Figure 2(a)), and aligned in the vertical axis as depicted in Figure 2(b). Finally, the squared region is extracted as in Figure 2(c), enhanced by means of histogram equalization and resized to 128×128 px Figure 2(d). To obtain the characteristics that are representative of the palm (principal lines and wrinkles), the texture descriptor Uniform Local Binary Patterns (LBP) [11] is applied using a 3×3 neighborhood in a similar way as [12]. Palmprint features are represented by a

[1] http://cswww.essex.ac.uk/mv/allfaces/index.html

histogram in which a different label is given to each uniform binary pattern and another one is assigned to the rest of binary patterns. Although this histogram could be directly used as the feature vector corresponding to the user of the biometric system, aimed to add some global information, the image is divided into several subregions and a histogram is calculated for each sub-image. Finally, histograms are concatenated to compose the final biometric feature vector which describes the palm. Different subregions sizes between 8×8 and 32×32 pixels were empirically tested and the results demonstrated that the best size to subdivide the image is 8×8 pixels.

Biometric information of both traits is fused at score level using the procedure described in Section II-A4.

3) In-Air Signature Biometric Module: In-air signature is a biometric mode which analyses the signal provided by a 3D accelerometer included in a mobile device when the user makes a short gesture in the air. A pattern matching algorithm is used to produce a score that represents the similarities between two 3D temporal signals. This algorithm is based on the Dynamic Time Warping algorithm, but using a Gaussian kernel and an alignment with a linear reconstruction of the temporal signals [13], trying to find the best alignment of the signals in order to mitigate small differences of the repetitions of the gesture. During the enrollment, user repeats n times the gesture, generating a 3D temporal signal of the acceleration values at each repetition. These n signals are compared two-by-two, obtaining n score value. The biometric profile is composed by the n acceleration signals and the average of the n score values. During the access user repeats once the gesture producing a new 3D signal of the acceleration values. This signal is compared using the pattern matching algorithm with the three signals stored in the biometric profile, obtaining three score values. The average of this score value is divided by the average of the score values of the biometric profile, creating a final score value which represents to what extent the gesture performed is similar to the gestures made at enrolment.

4) Biometric Fusion: The information coming from the different biometric techniques is fused at score level using the weighting sum rule. In mathematical terms this means that the scores s_i, with $i = 1, ..., N$ and N the number of biometric modalities, are mapped from $\mathbb{R}^N \rightarrow \mathbb{R}$ by the min function to get a single score S:

$$S = min(w_1 s_1, ..., w_N s_N) \tag{1}$$

where the factors w_i weight the influence of each score in the final fusion. The optimum values for the weighting factors are obtained by means of a genetic algorithm which starts with an initial population of 300 individuals and compute 20 generations.

A previous normalization of the scores is required to address the differences between feature spaces that the statistical distributions of the scores present. To this end, min-max normalization is applied, transforming the scores to a $[0, 1]$ domain:

$$\widetilde{S}_l = \frac{S_i - min(\{S_i\}_{i=1}^N)}{max(\{S_i\}_{i=1}^N) - min(\{S_i\}_{i=1}^N)} \tag{2}$$

being S_i the set of scores provided by a biometric modality matcher, where $i = 1, ..., N$ and N the number of scores.

B. Security Levels

Considering that the accuracy of the biometric recognition increases but the comfort of the user decreases with the number of biometrics involved in the authentication process, the proposed system offers three different security levels:

1) Low Level: The low authorization level will concern low-risk services which involve non-sensitive actions but are repeated continuously. In this case, the authentication should be fast and simple but reliable enough to guarantee a secure access to the device in public scenarios. This way, for the low security level just one biometric mode will be requested during the authentication process.

2) Medium Level: The medium authorization level is the natural context for those services which requires a compromise between the management of relatively sensible information and a certain frequency of use. In this case two biometrics will be involved in the authentication process, balancing comfortability and security.

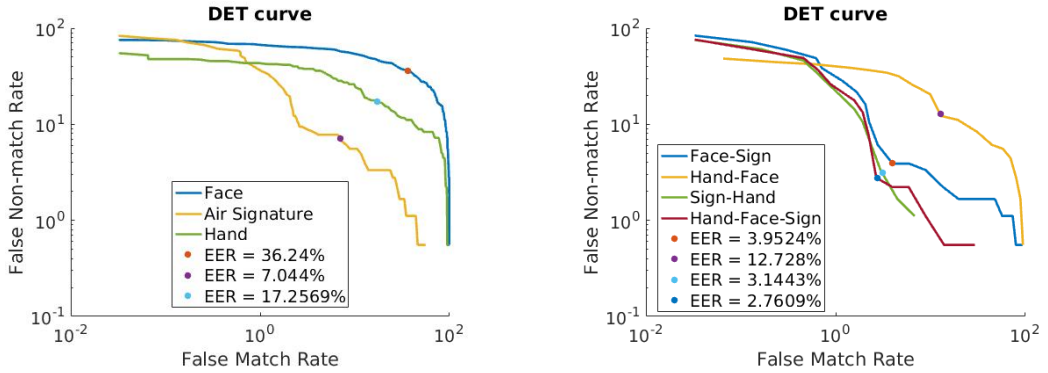
3) High Level: Those services related to highly sensitive information will be accessed through the high security level. This scenario requires a reliable and robust authentication even if comfort and speed are compromised. Accordingly, all the biometrics included in the system will participate during the high security level authentication process.

C. Coercion and Aliveness Detection Modules

Aimed to provide the system with additional security, coercion and aliveness detection modules have been included.

Coercion measures are intended to avoid that unauthorized persons access the final services by forcing the genuine user to unlock the device. To preserve the security of the user, the activation of the coercion mechanism should be imperceptible and easy to recall even under stressful situations. This way, the coercion mechanism is based on triggering a signal when a prolonged pressure over any zone of the touchscreen is detected. The coercion status should be managed by the service which is being accessed without displaying any notification to keep the user safe and the authentication will continue as usual and a simulation of service will be displayed.

Aliveness detection has been designed to guarantee that a real user is behind the biometric authentication and the system is not being tampered with artificial procedures such as to show a picture of the users hand or face. The aliveness detection process consists of a random set of activities that the user has to accomplish within a specified time. Particularly, these activities correspond to head movements and eye winking, which are directly related to face biometrics. Thereby both processes are merged in such a way that the facial features of the user are evaluated at some points of the aliveness detection process, ensuring that the genuine user is behind the aliveness detection test.



(a) Monomodal Results (b) Multimodal Results

Fig. 3. Comparison of the performance of the monomodal and multimodal biometrics included in the system.

III. System Evaluation and Configuration

To decide which biometric mode should be incorporated in each security level, an evaluation of the biometrics has been performed not only in terms of performance but also on the user acceptability.

A. Biometrics Evaluation

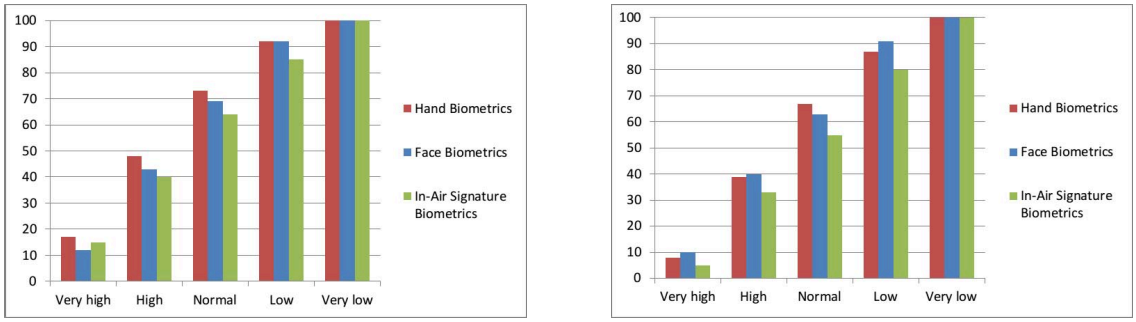
1) Datasets: Biometric algorithms has been evaluated using a database composed of information belonging to 20 contributors which was recorded to this end. For each user, 15 frontal face images, 30 contact-less hand images and 15 in-air signature signals were captured. The capturing process was performed under indoor environmental conditions so the backgrounds are controlled but the lighting conditions slightly differ between users due to variations in the illumination intensity coming from the outside. To capture the face images, no conditions about beard, glasses or hairstyle were imposed, but the users were advised to show neutral facial expression and to avoid wearing accessories which occlude eyes, ears, nose or mouth. In the case of the hand dataset the user is required to keep the hand centred in the image, outstretched and in a plane parallel to the camera plane in such a way that the whole hand is contained in the image but as close as possible to the camera. In addition, it is required that no other piece of skin or object with a similar color appears in the background of the image. Finally, to avoid difficulties during the segmentation process, participants were asked to remove their rings if they wore one. No restrictions on the use of bracelets or watches were imposed.

2) Evaluation Protocol: Biometric algorithms have been evaluated according to a protocol designed following the definitions suggested by the ISO/IDE 19795 [14], [15]. For each user and biometric trait the samples are divided into enrolment and access samples. In the case of face and in-air signature biometrics 5 randomly selected samples are used to enrol the user in the system and the remaining 10 to simulate accesses into the system. For hand biometrics 15 samples are involved in the enrolment process and the other 15 in the access process with the aim to minimize the effects of the high variability that hand images present. From the 15 samples which make up the enrolment set those 5 samples which better describe the user are selected using a brute force algorithm based on EER minimization, while the 10 samples used for the access are randomly selected. The biometric reference template of each authentic user is created through their enrolment samples. A list of genuine scores is obtained by comparing the access samples against the reference template of the same user and a list of impostor scores is generated using a Zero-effort scheme, where the access samples of a user are compared against the reference templates belonging to the rest of the users. Both sets of scores are used to estimate the performance of the biometric system, which is provided in terms of EER.

	Face	Hand	In-Air Signature	EER(%)
Monomodal Biometrics	•			36.24
		•		15.99
			•	7.04
Multimodal Biometrics	•	•		12.79
	•		•	4.00
		•	•	3.12
	•	•	•	2.78

TABLE I. Results of the biometric techniques included in the system in mono- and multi- modalities in terms of Equal Error Rate (EER).

3) Results: Evaluation results of the monomodal biometric techniques included in the system are summarized in the top half of Table I and Figure 3(a). Although hand biometric is composed by hand geometry and palmprint information fused at score level using the weighted minimum rule, it is considered as a monomodal biometric in the sense that information comes from a single image. These two features were also evaluated separately obtaining an EER equal to 26.03% and 23.57% for hand geometry and palmprint respectively. The combination of the information coming from both traits allows the reduction of the EER up to a value of 15.99% revealing the complementarity of both traits. These results, together with the 36.24% obtained by face biometrics ratify the high influence of uncontrolled conditions during the capturing process on image-based biometrics. In-air signature results are also influenced by the capturing device which is bigger and heavier than a mobile phone but not in such a strong way, obtaining a 7.04% EER. Multimodal results between different combinations of face, hand and in-air signature using the weighted minimum fusion rule are presented in the bottom half of table I Figure 3(b). Comparing the best monomodal result, provided by in-air



(a) Comfort Perception

(b) Security Perception

Fig. 4. User's Comfort and Security Perception of Hand, Face and In-Air Signature biometrics

signature (7.04%), with the best multimodal result, obtained when the information related to all the biometric modes is fused (2.78%), a significant increase in terms of effectiveness of the biometric authentication can be appreciated. Two-by-two fusions also present an improvement of the results, even when a biometric trait that has a quite good performance (in-air signature) with a second biometric modality that exhibits a worse recognition ability (face, hand). Thus, it can be derived that biometric fusion compensates the possible individual failures of monomodal biometrics improving the robustness of the system.

B. Users Acceptability Evaluation

Users acceptability of the biometric modalities was evaluated by means of a survey which studied two different aspects of the candidate biometric modes: if users feel comfortable using the technology (comfort) and if users feel that the security of the biometric mode is strong enough (perception of security). At the beginning of the survey, little but clear information about each biometric mode was provided, without including long explanations of the algorithms or the authentication process, looking specifically for security and comfort perception. The survey was answered by 100 participants from different countries of Europe, age, gender, technological experience and biometric expertise.

Figure 4 depicts users perception about comfort and security respectively. In both figures, the cumulative sum of answers from very high to very low has been presented in order to be able to check the proportion of people that provides at least certain value to each question.

According to Figure 4(a) the most comfortable biometric mode in the survey is the hand recognition, which is considered comfortable enough for more than 75% of the participants, followed by face recognition (around 70%). Finally, a bit more than 60% of the people declared comfort of in-air signature recognition as normal.

The security perception results show that the face and hand recognition outstand in respect to voice and gesture recognition. In this case, around 70% of the participants perceived hand biometric as normal security, while 65% and 60% of people assumed that face and in-air signature biometrics are secure enough.

C. System Configuration

Considering the results coming from the biometrics and user acceptability evaluations, the system will be configured as follows:

1) Low Level: According to the monomodal biometric evaluation results, the most suitable biometrics for low security level is in-air signature. It is highly reliable and robust, and the experiments showed that it is not easy to reproduce a particular signature to hack the biometric authentication system. On the contrary, if users acceptability is taken into account, in-air signature is the least accepted biometric in terms of both comfort and security perception. However, it is necessary to consider that it is an emerging biometric and that every novel technology presents this kind of concerns at the beginning, but people usually overcome this sort of limitations if they obtain significant benefits using it. This way, in-air signature biometric is selected to be used for low-level security authentications.

2) Medium Level: In the medium level two of the three biometric modalities are fused to reinforce the security. Multimodal results show that the most reliable two-by-two fusion schemes are those containing the air signature, showing a similar performance when fused with the other two biometric modalities. Due to hand biometrics posing less acceptability concerns than face biometrics, the fusion of in-air signature and hand biometrics has been chosen to be used in medium security level scenarios.

3) High Level: In the case of high security level, all the biometrics will be fused to improve the accuracy of the biometric recognition. In addition, the aliveness detection module is also included at this level to reinforce the security.

Nevertheless, although this is the optimum configuration for most of the users, it should be taken into account that universality concerns could appear. In that case, the system is configurable enough to allow the modification of the biometrics included at each level.

IV. Conclusion

A multibiometric system including face, hand and in-air signature biometrics which is configurable at three different levels of security has been presented. The system is also provided with anti-coercion and aliveness detection mechanisms

to increase its security. As the security level is stronger, the number of biometric modes involved in the authentication process increase. The fusion of biometric information coming from different traits permits overcoming those drawbacks derived from the intrinsic properties of each biometric modality as well as the capturing process and the environmental conditions, making the system more robust.

To decide which biometric mode should be incorporated in each security level, an evaluation of the biometrics has been performed in terms of performance and user acceptability. To evaluate the performance of each biometric mode and the different fusion combinations (two-by-two and as ha whole) a multimodal dataset was used, which was recorded under realistic conditions. According to the results of these evaluations, low-security level will request in-air signature biometric information, medium-security level authentication will fuse hand and in-air signature biometrics and the three biometric modes contribute to high security level authentication together with aliveness detection mechanism. Moreover, the system has been designed to allow the modification of the biometrics included at each level in order to overcome possible universality concerns if they appear.

ACKNOWLEDGMENT

This work has been funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n o 610713.

Authors would also like to thank all the contributors who have participated in the experiments for their patience and cooperation, which has been a really appreciated aid.

REFERENCES

[1] M. Faundez-Zanuy, "Data fusion in biometrics," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 20, no. January, pp. 34–38, 2005.

[2] E. Vildjiounaite, S.-M. Mäkelä, M. Lindholm, R. Riihimäki, V. Kyllönen, J. Mäntyjärvi, and H. Ailisto, *Unobtrusive Multimodal Biometrics for Ensuring Privacy and Information Security with Personal Devices*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 187–201.

[3] C. McCool, S. Marcel, A. Hadid, M. Pietikinen, P. Matejka, J. Cernock, N. Poh, J. Kittler, A. Larcher, C. Lvy, D. Matrouf, J. F. Bonastre, P. Tresadern, and T. Cootes, "Bi-modal person recognition on a mobile phone: Using mobile phone data," in *Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference on*, 2012, pp. 635–640.

[4] D. J. Kim, K. W. Chung, and K. S. Hong, "Person authentication using face, teeth and voice modalities for mobile device security," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 4, pp. 2678–2685, 2010.

[5] B. J. Kang and K. R. Park, "A new multi-unit iris authentication based on quality assessment and score level fusion for mobile phones," *Machine Vision and Applications*, vol. 21, no. 4, pp. 541–553, 2010.

[6] S. Trewin, C. Swart, L. Koved, J. Martino, K. Singh, and S. Ben-David, "Biometric authentication on a mobile device: A study of user effort, error and task disruption," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ser. ACSAC '12, 2012, pp. 159–68.

[7] M. A. Turk and A. P. Pentland, "Face Recognition Using Eigenfaces," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91*, pp. 586–591, 1991.

[8] F. Samaria and A. Harter, "Parameterisation of a stochastic model for human face identification," in *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, Dec 1994, pp. 138–142.

[9] P. Phillips, H. Wechsler, J. Huang, and P. J. Rauss, "The feret database and evaluation procedure for face-recognition algorithms," *Image and Vision Computing*, vol. 16, no. 5, pp. 295 – 306, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S026288569700070X

[10] T. Connie, A. T. B. Jin, M. G. K. Ong, and D. N. C. Ling, "An automated palmprint recognition system," *Image and Vision Computing*, vol. 23, no. 5, pp. 501 – 515, 2005.

[11] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 971–987, Jul 2002.

[12] G. K. Ong Michael, T. Connie, and A. B. Jin T., "Review article: Touchless palm print biometrics: Novel design and implementation," *Image Vision Comput.*, vol. 26, no. 12, pp. 1551–1560, 2008.

[13] J. Guerra-Casanova, C. Sanchez-Avila, G. B. del Pozo, and A. de Santos Sierra, "A sequence alignment approach applied to a mobile authentication technique based on gestures," *IJPRAI*, vol. 27, no. 4, 2013. [Online]. Available: http://dx.doi.org/10.1142/S0218001413560065

[14] "Information technology – biometric performance testing and reporting – part 1: Principles and framework," 2006.

[15] "Information technology – biometric performance testing and reporting – part 2: Testing methodologies for technology and scenario evaluation," 2007.

BioALeg - Enabling Biometric Authentication in Legacy Web Sites

Sharareh Monfared, Daniel Andrade, Luís Rodrigues, and João Nuno Silva
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal
Email:{sharareh.monfared, daniel.andrade, ler}@tecnico.ulisboa.pt, joao.n.silva@inesc-id.pt

***Abstract*—The authentication of users in legacy web sites via mobile devices is still a challenging problem. Users are required to provide passwords, introducing several vulnerabilities: since strong passwords are hard to memorize, users often use weak passwords that are easy to break, and passwords can be intercepted by malware and stolen. In this paper we propose a novel architecture, named BioALeg, to support secure biometric authentication on legacy websites. Our approach leverages the potential of a Secured Personal Device (SPD), a hardware add-on for mobile phones that is being developed in the context of the PCAS European project. The device offers biometric authentication and secure storage services. BioALeg uses this infrastructure, and a companion web site plugin, to support biometric authentication in legacy web sites that currently use username/password authentication.**

In order to perform authentication, the smartphone requests a One Time Password (OTP) to the service provider when the user tries to access the service using the SPD. Due to the architecture and implementation of the SPD, the OTP transfer only occurs after the owner of the phone and SPD is correctly authenticated using biometrics. The PCAS infrastructure guarantees that, after the biometric authentication, the user identity is valid and accepted by all components. BioALeg has been implemented as an Android service and integrated with legacy web sites.

***Index Terms*—Legacy web services; Biometric authentication; One-Time-Passwords;**

I. Introduction

The authentication of users through web sites on mobile devices has always been a challenge. Previous studies have shown that most users of mobile devices have multiple online accounts, and that these accounts still require the typing of passwords for authentication [1]. Furthermore, most users tend to pick passwords that are easy to memorize and to type. Unfortunately, these passwords are also easy to guess and can be targeted by various attacks, such as phishing, dictionary attacks, or simply brute-force. Additionally, users often share passwords over multiple accounts, which makes these attacks more profitable.

To avoid the burden of typing and memorizing password, in recent years, various password managers have been developed and are currently available for various mobile operating systems. The use of password managers makes easier for the users to select stronger passwords when performing web site authentication. Unfortunately, password managers also have some severe vulnerabilities. The credentials used for web authentication need to be stored in a local file or in the cloud. The master password used to access this file can be attacked by sociotechnical attacks such as *shoulder surfing* or it can be compromised if the user selects a weak password [2]. Also, the cryptographic primitives used to encrypt the file and/or the communication with the cloud may not be fully secure. Finally, several of these solutions use the clipboard for exchanging information among the web forms and password managers. This communication channel is open, so every application running simultaneously on an Android device can read the items stored in the clipboard at any time. This can be used by malicious applications to harvest passwords as they are passed through the clipboard [3][4].

Our work aims at solving the main vulnerabilities of password use in mobile devices: insecure storage of the passwords, insecure communication with the browsers, and leakage of the used passwords. We address these problems with the help of a Secured Personal Device (SPD), a hardware add-on for mobile phones that is being developed in the context of the PCAS European project [5]. The device offers biometric authentication and secure storage services, that we use to support biometric authentication into legacy web sites. To support user authentication in this manner, web site providers just need to implement an OTP generator and a simple web service on the server the generation and transfer of the OTP to the SPD/web browser.

We have named our solution BioALeg. It consists of combining the use of One Time Passwords (OTP) with the biometric authentication features of the SPD. An OTP is a unique secret key, that is shared between the application server and the user, and that can be used only once to perform authentication. Typically, this secret key is generated on demand by the server and transferred to the user device using an secondary communication channel, that offers some additional authentication guarantees (SMS for instance). The transfer of this secret key back to the server validates the user identity. In BioALeg, the transfer of the secret from the server to the client and back to the server requires the user to authenticate using the biometric sensors of the SPD.

BioALeg is composed of a set of components running on the web site and on the mobile phone. On the web site, the BioALeg is responsible for the generation of OTP and for transferring it to the SPD. On the smartphone, the BioALeg first request the generation of the OTP on the web site, receives it and transfers it to the authentication form on the browser. Since all communication steps are authorized in the SPD by means of biometric authentication, the completion of these steps guarantees the validation of the user identity.

978-1-5090-5259-2/16 $31.00 © 2016 IEEE
DOI 10.1109/SRDSW.2016.15

BioALeg allows the use of strong OTPs by supporting different algorithms implemented only in the server. This makes this system more resilient against risks and hacks.

In detail, the interactions between these components occur as follows. When an user attempts to login on a server, the server uses a secret key and the current time to generate a token (an OTP). After the OTP has been generated, it will be stored in the server database and sent to the SPD. Every time that a new OTP is generated, the database is updated such that only the last generated token can be used to login. Also, the token is only valid for a short period of time; the token invalidated as soon as the user successfully logs in or by the end of the validity period, if the token is not used. On the smartphone, the tokens received from the server are transferred from the SPD to the web browser only if the biometric authentication of the user succeeds, In this case, the token is transferred to the browser using a generic driver (that interacts with the SPD) and a service to fill the password fields on the web form.

The rest of the paper is structured as follows. Section II presents the related work, followed by a description of the proposed architecture in Section III. An evaluation of BioALeg is presented in Section IV, and Section V concludes the paper.

II. Related Work

In this section, we briefly survey related work on traditional authentication, mobile device authentication vulnerabilities, and multi-factor authentication. We also describe the main features of the SPD device.

A. Usernames and Passwords

Nowadays, most online services authenticate users using the traditional method based on an username and a password: the username identifies which account the client wishes to access, while the password is used to validate the identity of the client [6]. On the server side, passwords are stored using a cryptographic hash function and thus, unless other vulnerabilities are introduced in the authentication work-flow, an attacker is forced to guess the password to attack the system.

Unfortunately, guessing a password may be relatively easy. In fact, users often choose simple passwords, with low entropy, allowing for efficient brute-force or dictionary attacks [7][8]. Other attacks, such as phishing [9] also allow to obtain passwords from users. Furthermore, since users tend to share the same password for multiple services, once a password gets stolen, every associated account is compromised. As a result, obtaining passwords by illegitimate means has become a lucrative business.

B. Vulnerabilities of Mobile Device Authentication

When users rely on mobile devices to authenticate, new vulnerabilities are introduced. This happens because mobile devices inherit most vulnerabilities of desktop computers, amplified by the higher risk of loss or theft, and its usage patterns add new vulnerabilities [2], as listed below:

- *Insecure data storage*: Users may rely on password managers to save passwords that are hard to memorize. However, these managers may store passwords in the memory of the mobile device where it may be read by other applications or use insecure communication when backing up the database to the cloud [10]. This causes the exposure of sensitive information.
- *Infected Device:* Users tend to install applications, such as games, from untrusted sources on their mobile devices. Also, some devices quickly become obsolete and no longer support required software updates. This makes easier for mobile devices to become infected by malware.
- *Shoulder Surfing attacks*: Mobile devices tend to be used in public areas, where it is easy for an attacker to observe or capture the user's movements to get the password [1].
- *Dictionary Attacks*: Because long passwords are harder to type on mobile devices, users tend to pick simpler passwords that are more vulnerable to dictionary attacks [1].
- *Unauthorised use*: Mobile devices can easily get lost or be stolen. Once the attacker has physical access to the device he/she can generally get access to all the information stored on the device [11].
- *Man in the middle attack*: Mobile devices typically access the internet using untrusted networks (for instance, wifi networks), making easier to setup man in the middle attacks: a form of message tampering in which an attacker intercepts the very first message in an exchange of encryption keys to establish a secure channel [12].
- *Cached Passwords*: Since passwords are inconvenient to type in a mobile device, there is a strong motivation to cache them in the browser password manager. This creates a window of opportunity for an attacker to retrieve passwords from this insecure password storage.

C. Multi-Factor Authentication

Multi-Factor Authentication is an authentication strategy that requires the users to provide multiple pieces of evidence. Typically, these evidences are the combination of a password (something the user knows) with an additional token such as something that the user owns or something that captures who the user is (such as a biometric verification). Multi-Factor Authentication aims at strengthening the security of password-based login authentication by requiring an attacker to compromise multiple mechanisms before successfully breaking into the target [13]. The implementation of multi-factor authentication on mobile devices is simplified by the fact that most devices support multiple interfaces and multiple communication channels/protocols (infra-red, Bluetooth, 3G, and WLAN connectivity).

1) One-Time Passwords: As the name implies, One-Time Passwords (OTP) are passwords that can be used only once to authenticate a user with a service. Thus, even if the password is stolen, it cannot be used again [14].

One-time passwords can be generated offline and stored for later use, but such approach shares many of the vulnerabilities enumerated before: if the storage is compromised, the attacker

will be able to use all the passwords that have not been used before. Another approach relies on the generation of OTPs only when needed, such that it does not need to be stored persistently.

OTPs can be generated by the server and sent to the user by an secondary channel, such as via a SMS service. In this case, the user needs to access web service, use a traditional password to request the generation of the OTP and own a mobile phone to receive the SMS.

Another current OTP implementation relies on the use of an smartphone application to generate the one-time password on demand. In this case, the users needs to know a password to activate the generation an OTP on the device. Google Authenticator [15] is a software based solution that generates one-time passwords on mobile devices, that implements this solution.

2) Time-based One-time Password (TOTP): A Time-based One-time Password Algorithm (TOTP) is an algorithm that computes a one-time password from a shared secret key and the current time [16]. The TOTP combines a secret key with the current time-stamp using a cryptographic hash function to generate a one-time password. TOTPs make easier to enforce a limited time validity on the generated token.

3) Mobile OTP implementations: Several system that make use of OTPs or TOTPs have been developed in the past. We briefly present some of the most relevant approaches in the next paragraphs.

a) SMS OTP: One way of deploying an OTP-based authentication scheme, using a channel that is available to wide range of users, is to use the Short-Message Service (SMS) provided by mobile telecom operators as a secondary channel. Typically, the server generates the one-time code using the OTP algorithm, and sends this to the user's smartphone using the SMS. Authentication happens when the server recognizes that the user typed in the correct code during login.

One of the benefits of SMS OTP is that the user does not have to carry an additional hardware device, only her smartphone that it is assumed to carry anyway. Systems that use this approach generally produce four to six digit OTPs that are easy for users to copy and type for authentication.

A limitation of the approach is that, if the mobile phone is stolen, the attacker will be able to receive the SMS that was intended to the user[17]. Also, a powerful attacker may intercept the communications with the smartphone and capture the SMS content in transit. Furthermore, since many delays may occur in the process of delivering a SMS (even up to more than a minute [10]), the validity of the generated token needs to be large to avoid it being invalidated before the user uses it; unfortunately this also widens the window for an attacker to steal and use the token. The use of smartphones allows the installation of mall-ware that intercepts the SMSs.

b) Google Authenticator: Google Authenticator[15] uses an offline variant of TOTP, where the user's device generates the one-time codes and the server verifies it (this works because both can use the same algorithm to create the same OTP). Since secret keys have been previously shared between the client and the server, the server is able to validate the TOTP being presented for authentication. Because both participants have generated the same one-time code on the given time window, the server can verify the code sent from the client.

However, if the mobile device is stolen or compromised (virus),the user credentials can be compromised, allowing the request of OTPs by the attacker. if the smartphone storage is compromised the secret shared key between client and server can be discovered and used by the attacker to generate TOTPs.

c) YubiKey OTP: A YubiKey [18][19] is a small hardware device developed by Yubico that supports two-factor authentication. If a user registers with a service or site that supports these kind of authentication, each time user logs in, the service will request proof that user has his/her YubiKey in addition to the static password. The YubiKey is capable of generating standard OAUTH (Open Authorization Protocol) tokens as well as its own Yubikey-OTP, and implements challenge-response operations. YubiKey is vulnerable to theft: if someone steals the device, data can be compromised, since there is no mechanisms to prevent unauthorized users from using it.

D. SPD - PCAS

PCAS [5] is an European project whose goal is to provide a solution to allow users to store confidential information and to perform web site authentication in a secure manner. Central to the PCAS architecture is a special hardware device, the Secured Personal Device (SPD), that offers secure storage, secure communication, and biometric user authentication. The SPD operates as a sleeve that can be attached to a mobile phone and that is able to provide storage to trusted applications running on the phone and allows establishing end-to-end secure connections with external web seistes using the communication capabilities of the attached mobile phone.

Figure 1 provides an overview of the PCAS architecture. The architecture defined by the PCAS project includes a number of components, as described below:

- The Secure Personal Device (SPD) is a smartphone add-on that stores information securely, authenticates the user, and mediates a safe communication with service provides.
- Secured Trusted Applications (STA) are applications that executes on the smartphone and that is authorized to communicate with the SPD to read and store data. The communication between the SPD and a STA is mediated by a dedicated library and service.
- The Secure Trusted Gateways (STG) is the PCAS component that implements the PCAS PKI and manages all identities for SPD, service providers, and users. The key management done in the context of the STG guarantees the identity of entities communicating, namely: i) between the SPD and the service provider and, ii) the authenticated user and the service provider.
- A Proxy for Secured Trusted Applications (PSA) is the PCAS component that allows the integration with existing web sites. This proxy translates the authenticated calls generated in the SPD into application level web site calls.

This proxy also allows matching SPD identities (validated by the cryptographic algorithms used) into user identities (in the service provider domain).

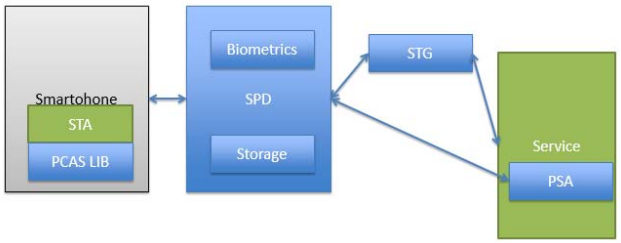


Fig. 1. Architecture for secure web site access using SPD

III. BioALeg

In this paper, we propose a method to use the SPD for authenticating users in mobile devices and to use to such authentication to login on legacy web sites. Since the biometric authentication is local to the SPD, we resort to a token produced in the server that is transferred to the browser on the mobile device *via the SPD* to validate the user identity.

In order to authenticate to a legacy web site, the user will be asked to fill a regular login form. The user will not use a fixed password, but will use an OTP. This OTP will be transferred to the smartphone through the SPD, an action that only occurs after the user biometric authentication is performed successfully. After the biometric authentication is performed by the SPD, the OTP will be delivered to the web browser and, on submission, transferred back to the server. Since the PCAS infrastructure provides identity management to validate SPDs, web site, and users, if the server receives back the produced OTP, it will know that the owner of the SPD was correctly authenticated.

Using the SPD as a secure communication channel, the OTP can be generated by the server using strong algorithms, securely transferred to a valid SPD, and only accessed by the correct user.

This proposed solution resemble TOTPs, but differs in some aspects. TOTPs require the execution of compatible code in the server and mobile device. Since the SPD can not run custom code (developed by external programmers) it is impossible for a TOTP to be generated on the device. The generation and secure transfer of the OTP will overcome this limitation, with added security. Furthermore the algorithm to generate the OTP can change without any modifications on the smartphone code.

A. Architecture

When a user attempts to access remote web sites or services, BioALeg replaces password typing with biometric authentication, relying on the following components:

- The SPD device, which can only be accessed by her owner.
- A Secure Trusted Gateway (STG) controls the registered SPDs, mediates the establishing of communication between users and the service providers, and validates user's identities.
- Applications installed in the mobile phone, called Secured Trusted Applications (STAs), that provide interfaces for secure data and web services access. These applications are implemented by the service providers and will able to securely communicate with the SPD (for data access) and the service provider (for invocation of web site). Each Service provider should implement and sign its own set of STA.
- Security algorithms and protocols protect both communication and data. The public key infrastructure and protocols guarantee the identity of all the entities (SPD, user, services) in the system.

BioALeg extends the SPD/PCAS architecture with the following components:

- A web browser plugin that allows the secure use of this new authentication mechanism;
- Secured trusted applications are installed on the client's mobile phone and request the generation of a new OTP; these requests are forwarded to the service provider through the SPD;
- A server-side software, named the Service provider OTP Generator, required to generate OTPs and transfer them to the SPD.

Figure 2 shows how the user authenticates on a web site. During this procedure the user is not required to type any password; such procedure is replaced by the biometric authentication on the SPD.

The browser plugin is a generic component, that can be used by multiple web site providers. This plugin will automatically fill the forms with the username and the OTP provided by the server, if correctly authorized by the biometric authentication procedure implemented by the SPD. Each web site provider should implement the OTP generator, install it on its server, and configure the client side application to run on the mobile device as a Secured Trusted Application (i.e., an application that is able to interact with the SPD). These two components (both the web site and the STA) should be previously registered into the PCAS/SPD infrastructure. This is required because the interaction between these two components will be mediated by a Secure Trusted Gateway that will check the identity of both peers.

Next we describe in detail the steps involved in the authentication process when using BioALeg:

1) The user accesses a web site through the mobile browser;
2) The mobile browser connects to the web site and presents the login form;
3) The user requests the BioALeg to start the biometric authentication;
4) The BioALeg initiates a web site call to the OTP generator. BioALeg does not contact the web site directly, but uses instead the SPD as a proxy;

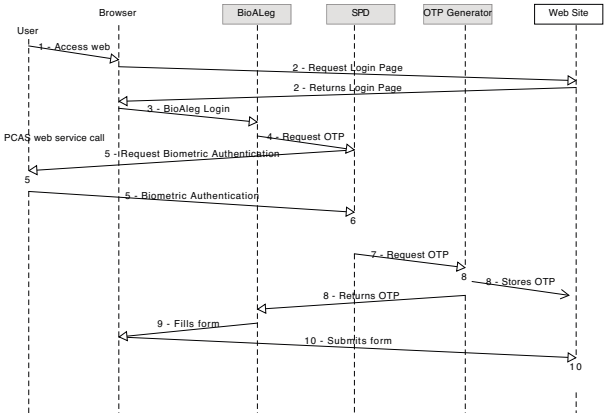


Fig. 2. BioALeg components (in gray) and interaction

5) Before forwarding the request to the web site, the SPD initiates the user biometric authentication and the user performs the requested authentication (e.g hand recognition);
6) The SPD validates the data read from the sensors with the internally stored user biometric template;
7) If the user trying to authenticate is the owner of the SPD, the OTP request is forwarded to the web site;
8) The OTP generator creates a new OTP, stores it on a persistent storage, and returns the value to BioALeg;
9) BioALeg fills the web form. The SPD sends an authentication acknowledge to the service-provider;
10) The user submits the authentication form (containing the received OTP), and the web site comapres the OTP sent by the user with the one generated previously.

Although some of the steps (3 and 9) may be performed by means of insecure communication channels, the more sensitive information is guaranteed by the SPD/PCAS system to be encrypted and valid. The PCAS key infrastructure and the cryptographic algorithms guarantee that if the web site receives a call from a certain SPD (step 7) the owner of the SPD was correctly authenticated. If, on step 6, the SPD fails to validate the user identity, the request will not be forwarded to the server and the OTP will not be generated.

B. Implementation

The proposed solution was implemented and deployed on mobile phones using the Android operating system. On the smartphone the BioALeg is composed of:

- the PCAS library, common to all application that use the SPD;
- a browser plugin to allow the automatic filling of the authentication forms;
- one or more STAs, for communication between the smartphone and the service provider.

Android allows the implementation and use of different software keyboards. The browser plugin is implemented by means of this Android extension. This software keyboard allows the retrieval of information from the forms being filled, and allows the filling of text in a secure manner, without resorting to the clipboard.

Since the SPD/PCAS environment requires the registration (and assignment to a specific service provider) of the STAs, each service provider willing to use BioALeg should implement its own STA. The users should install on the smartphone these Service provider specific STAs. Depending on the web site being accessed, this application receives from the browser plugin a request to authenticate user, and contacts the web site.

Legacy services must be extended with an OTP generator, that will be accessible by the SPD/smartphone trough a web site.

C. User Authentication

Before the first use of the SPD, it is necessary to register that SPD and assign it to a specific user. This is done after the purchase of the device and in front of an administrative officer (i.e., authorized personal). The whole system guarantees that all authenticated accesses performed from that SPD are done by its registered owner.

Before being allowed to deploy services on the SPD/PCAS infrastructure, service providers also have to register the identity of the web site and STA applications. The public keys stored in the gateway (STG presented in Figure 1) guarantee the authenticity and identity of each application and service provider.

The SPD/PCAS infrastructure guarantees that all accesses from a certain SPD were performed after the owner of that device was correctly authentication using biometric.

IV. Evaluation

BioALeg allows the seamless integration of biometric authentication into legacy web sites. The avoidance of requiring users to memorize or type password reduces the risks and attacks, and increased the usability of the available authentication methods available on mobile devices.

A. Security

Table I summarizes the differences among the proposed solution and other solutions based on the various risks and attacks.

	Username password	SMS OTP	TOTP	Password Managers	BioALeg
Insecure Data Storage	-	-	-	👎	-
Infected Device	👎	-	👎	👎	-
Shoulder Surfing Attack	👎	-	👎	👎	-
Dictionary attack	👎	-	-	-	-
Unauthorised Use		👎	👎	👎	-
Man In The Middle	👎	👎	-	👎	-
Cached Password	👎	-	-	👎	-

TABLE I
Authentication Methods and attacks (👎- vulnerable)

As it is depicted in Table I, the classic authentication method (based on username/password) is the most unsafe method, as it can be compromised by several attacks (most

of them related to user selecting weak passwords). Since most of the authentication methods resort to tokens, they become vulnerable to attacks (infected device or shoulder surfing). The unauthorized use of the device leads to other attacks: the user can request a SMS based OTP that will be received in the same smartphone.

BioALeg is resilient of the various attacks due to its characteristics:

- use of OTP - avoiding the use of insecure storage, or cached passwords, voiding the use of dictionaries to find the passwords;
- use of strong OTP generators - even on infected devices the discovery of the OTP generator function will be complex;
- biometric user authentication - shoulder-surfing attack will be impossible;
- no exchanged secrets - since no secrets are exchanged and shared between the smartphone and the service provider, the MiM attack reders no relevant information

B. Usability

Table- II compares how the different authentication mechanism handle several usability concerns.

	Username Password	SMS OTP	Password Managers	Google Authenticator	BioALeg
Memorizing passwords	👎	👎	-	-	-
Form Filling	👎	👎	-	👎	-
Typing ease	👎	-	-	-	-
Memorizing Master Password	👎	-	👎	👎	-

TABLE II
AUTHENTICATION METHODS USABILITY (👎 - ISSUE WITH USAGE)

The use of biometric algorithms tailored to mobile devices (implemented in the SPD/PCAS system) prevents users from typing passwords or tokens on the smartphone touch screen. This greatly reduces most of the issues related with usability of authentication methods on mobile devices.

C. Compatibility

Our proposed solution is compatible with all recent android versions (from Kitkat onward), and browsers (Chrome, Firefox, etc.) running on Android devices.

V. Conclusion

With the increasing use of mobile devices, the threat of compromising credentials grows: weakly-chosen static passwords, reuse of password in multiple services, password manager implementing weak security on the storage, and communication of the passwords greatly increase risks in mobile devices.

In this paper, we propose the integration of a novel smartphone add-on, the SPD, that supports user biometric authentication to implement a more secure mobile user authentication. Thanks to the SPD/PCAS infrastructure, it is possible to integrate biometric authentication into legacy web sites, with increase of mobile authentication security and usability. Service providers only have to implement a server plugin that generates OTPs and delivers them to the SPD using a web site. The OTP generation algorithm only runs on the server, allowing the implementation of custom, stronger algorithms. On the android phone, the BioALeg is composed of a plugin to fill web forms and a application to interact with the service provider. The access to the service will trigger a OTP generation that only completes after the successful user biometric authentication on the SPD.

Acknowledgments: We thank the anonymous reviewers for their comments on a previous version of this manuscript. This work has been partially by the EC through project PCAS (FP7-ICT-2013-10 grant agreement no 610713) and by Fundação para a Ciência e Tecnologia (FCT) through project with reference UID/CEC/50021/2013.

References

[1] M. Raza, M. Iqbal, M. Sharif, and W. Haider, "A survey of password attacks and comparative analysis on methods for secure authentication," *World Applied Sciences Journal*, pp. 439–444, April 2012.

[2] A. K. Jain and D. Shanbhag, "Addressing security and privacy risks in mobile applications," *IT Professional*, vol. 14-5, pp. 28–33, Sept 2012.

[3] X. Zhang and W. Du, "Attacks on android clipboard," in *Detection of Intrusions and Malware, and Vulnerability Assessment: 11th International Conf., DIMVA 2014, Egham, UK*. Springer, 2014.

[4] S. Fahl, M. Harbach, M. Oltrogge, T. Muders, and M. Smith, *Hey, You, Get Off of My Clipboard*. Springer, 2013, pp. 144–161.

[5] PCAS Consortium, "Pcas - personalised centralized authentication system," https://www.pcas-project.eu/, accessed: 2016-05-30.

[6] M. L. T. Uymatiao and W. E. S. Yu, "Time-based otp authentication via secure tunnel (TOAST): A mobile TOTP scheme using tls seed exchange and encrypted offline keystore," in *2014 4th IEEE Int. Conf. on Information Science and Technology*, 2014.

[7] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proc. of the 12th ACM Conf. on Computer and Communications Security*. ACM, 2005.

[8] B. Pinkas and T. Sander, "Securing passwords against dictionary attacks," in *Proc. of the 9th ACM Conf. on Computer and Communications Security*, ser. CCS '02. New York, NY, USA: ACM, 2002, pp. 161–170.

[9] P. Soni, S. Firake, and B. B. Meshram, "A phishing analysis of web based systems," in *Proc. of the 2011 International Conf. on Communication, Computing & Security*, ser. ICCCS '11. ACM, 2011, pp. 527–530.

[10] A. Sethi, O. Manzoor, and T. Sethi, "User authentication on mobile devices," Cigital, Tech. Rep.

[11] P. Ratazzi and el al, "A systematic security evaluation of android's multi-user framework," in *Proceedings of the Third Workshop on Mobile Security Technologies (MoST)*, 2014.

[12] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed. USA: Addison-Wesley, 2011.

[13] P. Crocker and P. Querido, "Two factor encryption in cloud storage providers using hardware tokens," in *2015 IEEE Globecom Workshops (GC Wkshps)*, Dec 2015, pp. 1–6.

[14] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "Totp: Time-based one-time password algorithm," Internet Requests for Comments, RFC 6238, May 2011. [Online]. Available: https://tools.ietf.org/rfc/rfc6238.txt

[15] "Google authenticator opensource," https://github.com/google/google-authenticator, accessed: 28-06-2016.

[16] D. M'Raihi and el al, "Hotp: An hmac-based one-time password algorithm," Internet Requests for Comments, RFC 4226, December 2005. [Online]. Available: https://www.ietf.org/rfc/rfc4226.txt

[17] C. Mulliner, R. Borgaonkar, P. Stewin, and J.-P. Seifert, "Sms-based one-time passwords: Attacks and defense," in *Proc. of 10th Int. Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA 2013, Berlin, Germany*. Springer, 2013.

[18] Yubico Inc., "Yubico for individuals," https://www.yubico.com/why-yubico/for-individuals/, accessed: 2016-01-7.

[19] R. Künnemann and G. Steel, "Yubisecure? formal security analysis results for the yubikey and yubihsm," in *Security and Trust Management: 8th Int. Workshop, STM 2012, Pisa, Italy"*. Springer, 2013.

2016 IEEE 35th Symposium on Reliable Distributed Systems Workshops

SafeRegions: Performance evaluation of multi-party protocols on HBase

Rogério Pontes, Francisco Maia, João Paulo, Ricardo Vilaça
HASLab, INESC TEC & University of Minho

***Abstract*—On-line applications and services are now a critical part of our everyday life. Using these services typically requires us to trust our personal or company's information to a large number of third-party entities. These entities enforce several security measures to avoid unauthorized accesses but data is still stored on common database systems that are designed without data privacy concerns in mind. As a result, data is vulnerable against anyone with direct access to the database, which may be external attackers, malicious insiders, spies or even subpoenas.**

Building strong data privacy mechanisms on top of common database systems is possible but has a significant impact on the system's resources, computational capabilities and performance. Notably, the amount of useful computation that may be done over strongly encrypted data is close to none, which defeats the purpose of offloading computation to third-party services.

In this paper, we propose to shift the need to trust in the honesty and security of service providers to simply trust that they will not collude. This is reasonable as cloud providers, being competitors, do not share data among themselves. We focus on NoSQL databases and present SafeRegions, a novel prototype of a distributed and secure NoSQL database that is built on top of HBase and that guarantees strong data privacy while still providing most of HBase's query capabilities. SafeRegions relies on secret sharing and multi-party computation techniques to provide a NoSQL database built on top of multiple, non-colluding service providers that appear as a single one to the user. Strikingly, service providers, individually, cannot disclose any of the user's data but, together, are able to offer data storage and processing capabilities. Additionally, we evaluate SafeRegions exposing performance trade-offs imposed by security mechanisms and provide useful insights for future research on performance optimization.

***Keywords*-HBase, Secure databases, Multi party computation**

I. Introduction

Convenience and significant economical savings are spurring many enterprises and end users to move their data and applications to third-party cloud services. This naturally implies trusting that these services employ the necessary mechanisms to ensure that their data is kept private. However, such trust is misplaced. In fact, recent studies have shown that a large amount of the data stored in third-party infrastructures is unprotected against the prying eye of a system administrator, a government subpoena or even an external attacker [1]. Notably, leveraging cloud database storage and computational capabilities while ensuring data privacy is still an open research challenge.

An immediate approach to achieve stronger data privacy guarantees is to strongly encrypt the data before uploading it to third party infrastructures. However, this solution raises several issues. For instance, if a traditional symmetric cypher is applied to every database entry many of the query capabilities of that database become unusable since data properties, such as order, are lost with the encryption scheme. As a result, the database becomes nothing more than a storage service and computation over the data must be made at the user side where data can be decrypted and processed. Needless to say that this defeats the purpose of using the storage and processing capabilities of a third-party infrastructure. This limitation can be partially solved by privacy-aware databases such as CryptDB that leverages multiple encryption schemes and rewrites database queries at the client side so that useful computation can be done over protected data [2]. However, this approach has been shown to still leak a significant amount of sensitive information [3].

This paper explores an alternative solution for providing a privacy-aware database. The core idea behind this solution is to divide data management and processing amongst multiple service providers with independent infrastructures. By doing so, no provider is able to access the original data or extract any kind of useful knowledge from the protected data, while the client still has a fully-functional NoSQL databases. Rooted on this idea, we propose a solution that relies on secret sharing to divide sensitive data into a set of secrets that are stored across several domains that are not expected to collude. Each secret reveals nothing about the data and can be seen as a piece of a puzzle. With a single piece the puzzle cannot be solved but, after every single piece is put into the right place the puzzle is complete and the original data is recovered. Additionally, the proposed solution relies on secure multi-party protocols (MPC) to leverage distributed computation on top of the secrets without leaking any information. To sum up, by combining secret sharing and multi-party protocols our proposal achieves both private data storage and private processing.

The main contribution of the paper is SafeRegions, a system built on top of the HBase NoSQL database that resorts to secret sharing and MPC. SafeRegions leverages the concept of a virtual cloud database composed by multiple independent untrusted cloud infrastructures, a concept that was previously discussed for secure storage systems [4, 5]. In more detail, our prototype resorts to different HBase clusters deployed on independent infrastructures and on a proxy that abstracts clients from this distributed setup, thus presenting SafeRegions to clients' applications as a regular HBase-like NoSQL database.

The paper is structured as follows: an introduction to secret sharing, multi-party protocols and HBase is presented in Section II. Section III presents SafeRegions' architec-

978-1-5090-5259-2/16 $31.00 © 2016 IEEE
DOI 10.1109/SRDSW.2016.16

ture and discusses its components. Section V presents the results and an analysis of SafeRegions prototype's evaluation. Section VI discusses related work while Section VII concludes the paper.

II. Building Blocks

SafeRegions combines secret sharing and MPC protocols with HBase to provide a privacy-aware NoSQL database solution. Next we discuss each of these building blocks in more detail.

A. Secret Sharing and MPC

Secret sharing schemes consider two types of entities, a *dealer* D and a set of *players* $P = \{p_1, \ldots, p_n\}$. The purpose of a dealer is to transform a value v in a set of n *secrets* and store on each player a single secret. Each secret by itself does not leak any information about v which conforms to our privacy needs. Value v can be reconstructed only when all secrets are grouped together [6].

The purpose of secret sharing is to ensure the privacy of stored data. This way, this technique is not intended to provide processing capabilities over the secrets. Secure multi-party protocols solve this limitation by calculating an arbitrary function over a set of private inputs, such as the generated secrets, without leaking any sensitive information. While many protocols have been proposed, there are few practical implementations, which has lead us to focus on the Sharemind protocols [7].

In Sharemind, values are protected with additive secret sharing over a finite field $\mathbb{Z}_{2^n}$. Additive secret sharing enables arithmetic calculations to be performed over the finite field which is crucial to compare values through secure protocols. Namely, two protocols are proposed, an *Equality* protocol and a *GreaterThan* protocol. The protocols are bound to three parties. With a lower number of parties, privacy cannot be ensured while an higher number would only increase the overhead of computation.

The Sharemind protocols follow the following scheme. There is one Dealer that, for each input value, generates three secrets (1 in Figure 1). Each secret is stored in one of three players (2 in Figure 2). To give an overview of a protocol execution (depicted in Figure 2), let us consider that the Dealer wants to search for a certain value v is stored in the system. In order to do so, the Dealer generates new secrets from v and sends one of them to each player requesting a comparison protocol execution (1 and 2 in Figure 2). Each player, in order to execute the protocol, follows a set of secret generation and secret exchanging steps with other players (3 in Figure 2). These computation and message exchanging steps can be repeated. Next, each player returns its computation result to the Dealer (4 in Figure 2). Finally, the Dealer can extract the computation result from the combination of the three secrets (5 in Figure 2). In this case the result is one if the value v was found in the system and zero otherwise.

Trust Model. The previous protocols are proven to be secure under the passive (honest-but-curious) model. This model assumes an attacker capable of gaining access to the infrastructure of at most a single party, before the execution of any protocol. The attacker can see all the secrets held by the party, as well as all the messages exchanged by such party. However, the attacker is not capable of interfering in any way on the output of the computation nor see any of the messages exchanged between other entities [7]. Furthermore the protocols are proven to be universally composable, meaning that every protocol on the framework can be composed and remain secure.

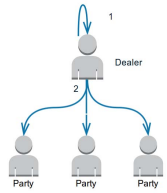


Figure 1. Store operation.

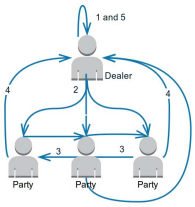


Figure 2. Search operation.

B. HBase

HBase is an open-source, widely used NoSQL data store that offers high efficiency and scalability [8]. This data store is heavily inspired by Google BigTable [9] and offers a powerful and scalable data model. Similarly to relational databases, information is stored on tables. However an HBase table, contains two levels of columns: column families and column qualifiers. Column families are defined on table declaration and group multiple column qualifiers. As such the column qualifiers are always associated to a column family and are only defined on value insertion. In the case a column family does not contain a column qualifier, the new qualifier is dynamical added. Each row on a table has an unique identifier and the cells of the table can contain empty values. Table I contains a model of an HBase table with one column family, Name, and two column qualifiers (First and Last Name).

As tables grow in size, HBase can scale dynamically by partitioning a table horizontally, creating multiple regions. Each region of a table holds a subset of the rows and is stored on a single RegionServer. These servers do most of the computation in HBase and are the backbone of HBase scalability. The reminder of the computation is performed by the HBase master. This entity manages the cluster in a master/slave architecture and is the entry point of the system. Every client operation must first interact with the

Identifier	Name	
	First Name	Last Name
1412	John	Alistar
2231	James	Smith
6262	Jane	

Table I
EXAMPLE OF AN HBASE TABLE

Master that redirects the clients to a RegionServer that can handle the request.

The HBase client API is composed by *PUT*, *GET*, *DELETE* and *SCAN* operations. The PUT and DELETE are the main operators used to insert, update or delete a record. A simple PUT operation requires declaring a row identifier, a column family and a column qualifier to be inserted/updated. On the other hand, a DELETE only requires specifying a row id in order to remove the corresponding row. By default the GET operator returns every column value from a single row of a table. However, all these operations can be enhanced to only specify certain column families or/and column qualifiers to be inserted, deleted or retrieved. Finally the SCAN operator returns the set of rows whose id is between a minimum and maximum value. Similarly, a SCAN can return all rows and corresponding columns or a filter may be used to discard unwanted records. For instance it is possible to define a SCAN operation that only returns the rows from Table I where the value of column *First Name* is "John".

The implementation/behavior of the previous operations can be modified/extended without breaking the API or changing the database source code due to the concept of Coprocessors. Coprocessors enable developers to load custom code for each operation that is executed by the database. While there are several types of Coprocessors, this paper focuses on the Endpoint Coprocessors. These coprocessors allow adding new operations that can be used by an HBase client application as a regular operation and are essential to provide the MPC protocols on SafeRegions in a generic and transparent manner.

The vanilla HBase system does not provide any mechanisms to protect stored data's privacy. In the next sections we present SafeRegions, a novel solution to ensure private storage and computation on top of HBase.

III. SYSTEM DESIGN

The main idea of *SafeRegions* is to use additive secret sharing to encrypt users' data and then to leverage the Equality and GreaterThan MPC protocols to provide secure HBase operations. These two protocols are essential since most computation done in HBase requires equality or order comparison of values. However, there are two major challenges that must be addressed to achieve a privacy-aware HBase MPC solution. Firstly, the previous protocols require three independent computation parties, which in our case are three HBase clusters. Secondly, we want to still offer a unified interface to clients, which implies enhancing HBase client to abstract this distributed deployment and the novel security mechanisms while providing the same API as the vanilla HBase client. In Figure 3 we present an high level overview of the SafeRegions architecture. Along this section we briefly describe its three main components: client, computation parties and communication middleware.

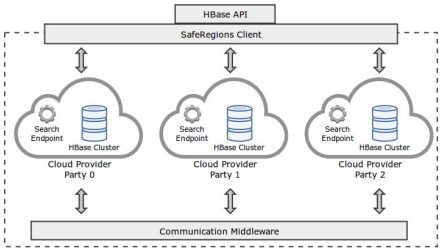


Figure 3. SafeRegions Architecture

A. SafeRegions Client

The client component is the entry point to the SafeRegions system and provides the same API as the vanilla HBase client. This component abstracts all the complexity of protecting users' data (secrets generation and decoding) and the communication with the three HBase clusters.

The client is also responsible for orchestrating the interaction with the different computation parties to process user's requests. It is the client component that transforms user's data into protected data to be stored and queried in the same way as the Dealer in Figure 2.

In order to query protected data, the client queries each one of the HBase clusters that then perform the necessary MPC protocols on the locally stored secrets. In more detail, this request can be made by issuing parallel remote-procedure calls (RPC) calls to each HBase SearchEnpoint *coprocessor*. Each HBase cluster processes the request resorting to its local storage, and returns as a response a vector with the resulting secrets of performing the desired MPC protocolos. With the three vectors, the client can merge the correct secrets and rebuild the information contained in the query response. Lets take as example a *GET* operation over protected row identifiers. The secure client must encode, with secret sharing, the identifier value being requested by the user and issue requests to the HBase clusters using these secret shared identifiers. Resorting to the *Search* operator, each cluster checks for a corresponding row in their local storage. If found, the cluster will reply to the client component with the corresponding row. Finally, the client component is responsible for decoding the answers and replying back to the user with the actual queried data.

B. Computational Parties

In order to support the MPC protocols discussed in Section II-A, we consider three computational parties in our architecture that correspond to the players in Figure 2. In SafeRegions these components are three HBase

clusters. Each cluster is responsible for one of the three secrets generated per data piece. Naturally, this implies that these clusters are structurally similar. They share the same table structure and will hold the same amount of data.

However, the clusters are necessarily deployed in independent infrastructures and can be uniquely identified by a number *ID*. This identification is important for the client to store and collect secrets from the distinct clusters. For instance, some data A is transformed into three secrets A_1, A_2 and A_3 to be stored in cluster 1, 2 and 3. Similarly, if A_1 is stored in $TableOne$ at cluster 1, then A_2 is stored in an identical table at cluster 2 and so forth.

In addition to maintaining an identical structure across HBase clusters, MPC protocols requires communication between the clusters. This is achieved with a *coprocessor* endpoint, which we call *SearchEndpoint*. The *SearchEndpoint* does not require any modification to the architecture neither to the implementation of the HBase core mechanisms. Nevertheless, the *SearchEndpoint* plays a central role in the architecture. Each RegionServer must contain this endpoint in order to expose the secure MPC protocols as a RPC to be used by the client. Additionally, the *SearchEndpoint* relies on a communication middleware for exchanging secrets across parties (clusters), that is further described next.

Upon the reception of a query request, the endpoint starts a MPC protocol for every record stored in that region server. The actual implementation of the protocol is contained in a MPC library that performs all the necessary computation (secret generation) and uses uses the communication middleware to exchange messages across parties (clusters). The validity of secrets computed with the MPC library is verified at the endpoint that is also responsible for replying back to the client with the query response.

C. Communication Middleware

By default HBase does not support communication across RegionServers. However, SafeRegions requires communication across distinct HBase clusters. To achieve this, we introduce a communication middleware between the RegionServers to handle the exchange of messages across parties. In order to support MPC protocols, the communication middleware offers two essential primitives. A $send(secret, party)$ primitive that delivers a secret to a target party in a non-blocking fashion and a $receive(party)$ primitive that waits for incoming messages. Moreover, the communication middleware ensures that the messages sent from one party to another arrive in order and that it is possible to always know the source party that sent the message. These two properties are needed for correctly supporting MPC protocols.

IV. Implementation

The system architecture proposed in the previous section leaves open implementation decisions that impact the inner working of the SafeRegions system. Starting with maintaining different HBase structurally similar, there is an immediate implementation detail that must be addressed, which is using secrets as row identifiers. In detail, if secrets are used to protect HBase row identifiers, then these cannot be mapped directly to the protected HBase schema as identifiers because secrets generated by the MPC library do not have deterministic content (due to MPC randomness). This means that it becomes impossible to match identical rows in different HBase clusters and to update, retrieve or delete a specific row in SafeRegions. To solve both problems, an extra column is added to the HBase table for storing the protected identifiers (secrets). Then, the row identifiers of each HBase table are replaced with virtual identifiers managed by the SafeRegions client. These identify uniquely each row and are identical across different clusters. This virtual identifier does not leak any sensitive information and allows matching rows and the corresponding secrets across different HBase clusters.

For each MPC protocol execution request, our current coprocessor implementation performs a simple sequential iteration over every record stored in the corresponding RegionServer and uses our implementation of the Sharemind protocols to execute the necessary computation. When this computation is finished, two of the RegionServers send the generated secrets to a third RegionServer. This third RegionServer combines the resulting secrets from every record and discovers what are the HBase records that match the NoSQL query being executed. This information is then sent to the SafeRegions client that retrieves from each HBase cluster the needed records (protected as secrets) and combines them to decode the original values. This last step does not compromise the privacy of stored values and is an optimization of the standard Sharemind protocol in order to lower the computation and number of messages/data received at the SafeRegions client.

Finally, Sharemind protocols were implemented in Java since there is not a freely available implementation. The communication middleware is implemented using Java NIO.

V. Evaluation

In order to evaluate the performance of adding privacy to common HBase operations such as, creating a table, inserting records, and retrieving records, we performed a set of experiments. The experiments ran on a cluster of servers equipped with an Intel i3 CPU with four cores at 3.7 GHz, 8 GB of RAM and a 128GB SSD. Each machine ran Ubuntu 14.04 and the SafeRegion Cluster was built using HBase 0.98 in standalone mode. The results obtained consist of one run of half an hour.

Finally, in the experiments we chose to preserve the privacy of HBase identifiers with secret sharing. This decision was taken because the tested operations perform computation over the identifiers and not across the column values. This way, to understand the overhead of our solution identifiers must be protected while values can be left in clear text.

We started by evaluating the latency for creating a

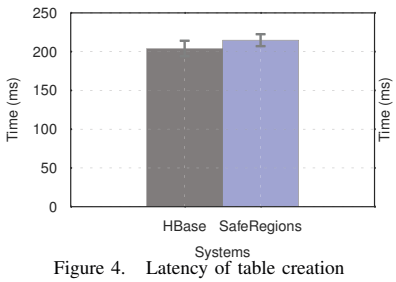


Figure 4. Latency of table creation

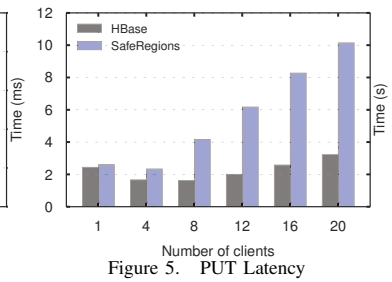


Figure 5. PUT Latency

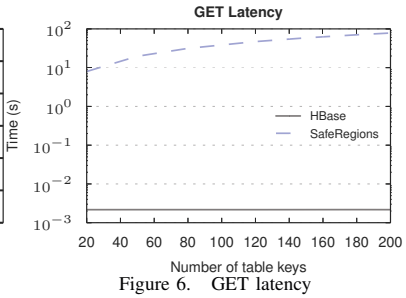


Figure 6. GET latency

table on vanilla HBase and on the SafeRegions system. SafeRegions system introduces an overhead of 4.9% in comparison to the vanilla HBase as depicted in Figure 4. As expected, this overhead steams from adding a column family in the table create statement and from issuing three concurrent requests to each HBase cluster.

We then proceeded to evaluate the overhead of PUT operations on both systems. In these experiments the client is only inserting new keys. Since updates are not being executed, no MPC protocol has to be performed. This way, we are evaluating the overhead of creating three secrets, and issuing three parallel PUT operations, one for each cluster. The evaluation benchmark simulated multiple clients by resorting to independent threads.

As can be seen in Figure 5 with a single client, the latency overhead of PUT operations is 7%, however as the number of clients increases the latency overhead also increases, being the highest overhead value approximately 220%. This significant increase on latency comes from spawning one thread per simulated client, each issuing three additional concurrent PUT requests (1 per cluster). In fact when there are n clients, there are $n * 3$ PUT operations to be made and $n * 3$ threads. For instance, when there are 20 clients, our benchmark has 60 PUT operations and threads being executed concurrently. On the other hand, the generation of random numbers by the clients has small impact in performance. In detail, we use the Uncommons maths library [10] that takes on average 0.016 ms to generate a random number.

Finally, we have evaluated the overhead incurred by executing a MPC equal protocol. To evaluate this cost, we ran a benchmark that pre-populated multiple rows on the system and then performed several GET operations. Each GET requires the equal protocol to be performed, as described in section IV, in order to retrieve the correct row identifier. From Figure 6, with a logarithmic scale, it is possible to verify that the MPC equal protocol incurs a significant overhead, while the default HBase can perform an operation with a latency of milliseconds, the SafeRegions operation is in the orders of seconds. While the latency of the vanilla HBase has a constant 2 ms latency with the increase of rows, the SafeRegion solutions sees an increase on latency. For two hundred keys the latency reaches the 78 seconds on average. This significant increase is due to the equal protocol that exchanges multiple messages between the parties, for instance, with 8 byte keys it requires 458 messages per comparison. Furthermore the MPC protocols requires a comparison with every record on the table, which is also a costly operation.

VI. Related Work

Bringing privacy guarantees to the database-as-a-service model (DBaaS) is a field still in expansion with multiple paths being pursued. In 2002, NetDB2's challenged the database community to explore several open challenges in databases [11]. One of the issues discussed was data privacy, which was tackled with symmetric or asymmetric encryption at the client side to protect users' data before being stored in a remote database.

The previous proposal requires queries computation to be placed on the client side, which defeats the purpose of leveraging the cloud's computational resources. In order to shift some of the computation away from the client, multiple systems proposed to use a trusted third-party service to handle the communication and computation between the client and the database service provider [12, 13, 14]. These solutions still depend on trusting a third-party entity.

In CryptDB a proxy mediates the communication between the client and database while rewriting queries to leverage computation over protected data [2]. Data can be encrypted with different schemes, deterministic encryption, order preserving encryption or Homomorphic encryption, with each one enabling different types of queries and having different types of security guarantees. In the case of deterministic encryption it has been shown that cryptDB is susceptible to frequency analysis attacks [3]. Monomi builds on top of cryptDB and improves the performance of query processing by choosing in anticipation the most appropriate encryption scheme for each database column. Also query plans are used to decide which parts of the query are processed in the untrusted service and which are executed on the proxy/client [15].

Wai *et al.* takes a different approach to existing systems by removing the proxy and sharing the computation between the client and the server [16]. Not only is the architecture different but also the encryption schemes. This system is based on secure MPC while sensitive data is encrypted using secret sharing which encodes the information in two secrets. One secret is stored on the client while the other is stored at the server. With secret sharing and secure MPC SQL operators can be pipelined in a query, unlike in CryptDB where the operators that can be executed are bound by the encryption scheme.

The existing solutions focus mostly on SQL databases and are not concerned with distributed databases that may provide better performance and scalability. Our approach is similar to Wai *et al.* since it also applies MPC protocols. However, all queries are processed in the untrusted servers, thus requiring minimal computation at the client side. Furthermore we do not rely on the client to store meta-data or secrets *i.e.*, all data is stored on the HBase clusters.

VII. Conclusion

This paper introduces SafeRegions, a novel system that combines secret sharing and MPC to provide privacy-aware data storage and computation in NoSQL. Our prototype resorts to the widely used HBase NoSQL data store and shows that it is possible to provide the full HBase API for clients while ensuring that their data is stored in a completely private fashion. In fact, by spreading the data (secrets) into multiple HBase clusters, even if one of these clusters becomes compromised, there is not any leakage of sensitive information.

Like any other security mechanism, secret sharing and MPC introduce a performance penalty in SafeRegions. This is a necessary tradeoff, that is discussed in our experimental evaluation section, in order to have stronger security guarantees.

As future work, many design and implementation improvements are still possible. For instance, the secrets can be calculated in parallel and the implementation of the MPC protocols can be improved with batching or additional parallelization.

To conclude, privacy in NoSQL comes always associated with a performance/functionality cost. However, we predict that in the near future, novel solutions and optimizations will be proposed to tackle the information privacy challenge that is now a global concern.

VIII. Acknowledegements

The research leading to these results has received funding from different sources. Rogério Pontes is financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project (POCI-01-0145-FEDER-006961), and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013. The remainder authors received funding from the European Union's Horizon 2020 - The EU Framework Programme for Research and Innovation 2014-2020, under grant agreement No. 653884.

References

[1] T. B. Pedersen, Y. Saygın, and E. Savaş, "Secret sharing vs. encryption-based techniques for privacy preserving data mining," 2007.

[2] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: Protecting confidentiality with encrypted query processing," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, ser. SOSP '11, 2011, pp. 85–100.

[3] I. H. Akin and B. Sunar, "On the difficulty of securing web applications using cryptdb," in *Big Data and Cloud Computing (BdCloud), 2014 IEEE Fourth International Conference on*, Dec 2014, pp. 745–752.

[4] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "Depsky: Dependable and secure storage in a cloud-of-clouds," in *Proceedings of the Sixth Conference on Computer Systems (EuroSys)*, 2011.

[5] H. Tang, F. Liu, G. Shen, Y. Jin, and C. Guo, "UniDrive: Synergize Multiple Consumer Cloud Storage Services," in *Proceedings of the 16th Annual Middleware Conference (Middleware)*, 2015.

[6] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[7] D. Bogdanov, M. Niitsoo, T. Toft, and J. Willemson, "High-performance secure multi-party computation for data mining applications," *International Journal of Information Security*, pp. 403–418, 2012.

[8] "Hbase." [Online]. Available: https://hbase.apache.org/

[9] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, pp. 4:1–4:26, Jun. 2008.

[10] "Uncommons maths." [Online]. Available: http://maths.uncommons.org/

[11] "Providing database as a service," in *Proceedings of the 18th International Conference on Data Engineering*, ser. ICDE '02, 2002, pp. 29–.

[12] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *Quality of Service, 2009. IWQoS. 17th International Workshop on*, July 2009, pp. 1–9.

[13] N. Jefferies, C. Mitchell, and M. Walker, *Cryptography: Policy and Algorithms: International Conference Brisbane, Queensland, Australia, July 3–5, 1995 Proceedings*, 1996, ch. A proposed architecture for trusted third party services, pp. 98–104.

[14] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, N. Mishra, R. Motwani, U. Srivastava, D. Thomas, J. Widom, and Y. Xu, "Vision paper: Enabling privacy for the paranoids," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, ser. VLDB '04, 2004, pp. 708–719.

[15] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," *Proc. VLDB Endow.*, vol. 6, no. 5, Mar. 2013.

[16] W. K. Wong, B. Kao, D. W. L. Cheung, R. Li, and S. M. Yiu, "Secure query processing with data interoperability in a cloud database environment," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14, 2014, pp. 1395–1406.

ARM TrustZone for Secure Image Processing on the Cloud

Tiago Brito, Nuno O. Duarte, Nuno Santos
INESC-ID / Instituto Superior Técnico, Universidade de Lisboa
{*tiago.de.oliveira.brito,nuno.duarte,nuno.m.santos*}*@tecnico.ulisboa.pt*

***Abstract*—Nowadays, offloading storage and processing capacity to cloud servers is a growing trend. This happens because high storage capacity and powerful processors are expensive, whilst cloud services provide a cheaper, ongoing, and reliable solution. The problem with cloud-based solutions is that servers are highly accessible through the Internet and therefore considerably exposed to hackers and malware. In this paper, we design and implement Darkroom, a secure image processing service for the cloud leveraging ARM TrustZone technology. Our system enables users to securely process image data in a secure environment that prevents exposure of sensitive data to the operating system. We evaluate our system and observe that our solution adds a small overhead to image processing when compared to computer platforms that require the entire operating system to be trusted.**

I. Introduction

Over the latest years, the cloud has become immensely popular due to the proliferation of numerous online services for storage, streaming, and processing of content. However, these services frequently handle user sensitive data which have security and privacy requirements that are not always properly considered by the providers of these services. In some cases, this negligent behavior has even lead to serious scandals such as celebrity photo [1], and user document [2] leaks. To make matters worse, instead of trying to enforce security protocols capable of preventing this type of accidents, these providers end up stitching user contract terms so they can be absolved of these incidents, giving a bad reputation to cloud providers in general [3].

To secure user generated content in the cloud, such as personal documents, images, or videos, a commonly used approach has been to encrypt the content at the client side before it is sent to the cloud. While this approach is effective whenever the cloud is used for persistent storage, it can no longer be applied in scenarios where content needs to be processed by the cloud service. Notable examples include cloud services such as Facebook or Instagram which apply transformation functions to the images uploaded by the users, for example to rescale, rotate, or reencode images. To perform such operations, the image data must be in its unencrypted format, point at which it may become vulnerable, e.g., to an attacker that managed to exploit some critical bug in the application code or in the operating system.

A promising alternative to encryption is to leverage Trusted Execution Environments (TEE) in order to safely perform image transformations at the cloud server without the need to rely on the rich operating system running on the server. Thus, if the OS is compromised, the TEE ensures that an attacker cannot access the memory regions allocated to the TEE where security-sensitive images are located. This approach has been recently adopted in many mobile device studies, whether to provide safe storage [4], enforce authentication mechanisms [5], [6], or provide OS introspection and monitoring capabilities [7], [8], [9]. One of the reasons this approach has been so popular in the mobile landscape, has to do with ARM TrustZone [10], a technology that allows the implementation of TEE systems and is present in the majority of mobile device processors.

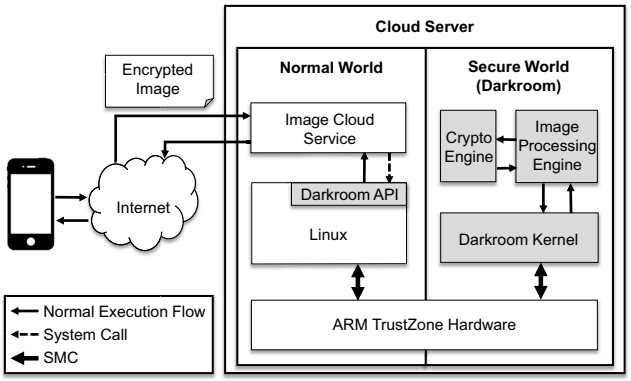


Figure 1. Architecture

In this paper, we explore the adoption of ARM TrustZone technology in order to provide an isolated environment for processing images securely on the cloud. Specifically, we present the design and implementation of Darkroom, a system that leverages ARM TrustZone to offer a secure image processing environment for cloud-hosted services. Next, we start by describing the design of our system.

II. Design

This section describes Darkroom, a TrustZone-based system which allows for secure image processing on the cloud. This system is capable of processing images without exposing them to the operating system. This is done by storing image data in encrypted form and using TrustZone-enabled ARM processors to securely process such images in isolation from the operating system.

Figure 1 shows the components of our solution and represents a possible execution flow for the Darkroom system. The flow starts in a client application, which is

978-1-5090-5259-2/16 $31.00 © 2016 IEEE
DOI 10.1109/SRDSW.2016.17

represented by a mobile device. The client application sends an encrypted image to the Image Cloud Service component, which can store the image data locally. Both the Image Cloud Service and the operating system run in the normal world context of the TrustZone-enabled processor. After uploading the image data, the client application issues a transformation request for the image. Upon receiving a transformation request, the Image Cloud Service sends the encrypted image data to the secure world. It also sends the transformation request and triggers a world switch via a Secure Monitor Call (`SMC`). This `SMC` gives control to the secure world which decrypts the image data and executes the requested transformation on it. After processing, the image is encrypted once again and sent to the normal world.

Note that we are primarily concerned about potential security breaches resulting from software exploits to normal world programs. In particular, we want to prevent external attackers that manage to take over control of the application or the operating system residing in the normal world from violating the confidentiality of users' images (e.g., by leaking them out). We assume that both cloud provider and cloud administrators are trusted. The trusted computing base (TCB) of our system comprises the hardware platform and the Darkroom components hosted in the secure world.

A. Rich OS Isolation Using ARM TrustZone

To isolate the rich operating system from Darkroom, we leverage ARM TrustZone. ARM TrustZone is a security extension present in ARM processors that provides a hardware-level isolation between two execution domains: *normal world* and *secure world*. Compared to the normal world, the secure world has higher privileges, as it can access the memory, CPU registers and peripherals of the normal world, but not the other way around. Isolation is enforced by checking and controlling the state of the CPU through a NS bit, and partitioning the memory address space into secure and non-secure regions. To perform a context switch between the different worlds, TrustZone offers the *SMC* instruction, which generates a software interrupt that is then handled by the secure monitor. TrustZone follows a similar approach with interrupts. Basically, Interrupt Requests (IRQs) and Fast Interrupt Requests (FIQs) can be configured to be either secure or non-secure interrupts. This means that secure interrupts can only be intercepted by the secure world and not the normal world. Both interrupt management and memory isolation mechanisms are fundamental in guaranteeing peripheral access isolation between worlds. Additionally, TrustZone features a secure boot mechanism that ensures the integrity and authenticity of the system running in the secure world.

B. System Deployment on ARM-based Cloud Servers

We envision Darkroom to be deployed on cloud servers maintained by the service provider. As opposed to Intel-based servers commonly adopted in today's cloud infrastructures, Darkroom's target servers must be based on ARM TrustZone technology. To deploy the system, the cloud provider must flash the firmware of its ARM servers so that the servers bootstrap into the secure world and run Darkroom's setup code before switching to normal world and loading up a rich operating system or hypervisor.

In this process, the cloud provider must generate a *root key* for each server. A root key is a public key pair KR whose private part (KR^-) is bundled into the Darkroom image right before the image is flashed onto a server's firmware. The private part of the root key will be accessible only by Darkroom within the secure world and there will be a unique root key for each cloud server. The public part KR^+ is certified by the cloud provider in order to assert the authenticity of this key. This key is fundamental to ascertain the authenticity of the Darkroom platform to a remote client and to securely share secrets with the Darkroom runtime without the need to trust the rich operating system. Note also that, in addition to the root key, the Darkroom image contains public keys of cloud administrators authorized to perform some security-critical operations, e.g., setting up image transformation functions, as explained next.

C. Setting Up Image Transformation Functions

Once a cloud server is up and running, the rich operating system takes up control of system resources and Darkroom is suspended until requests arise from the normal world to perform image transformation operations (e.g., image rescaling). To allow for additional flexibility, rather than hardcoding transformation functions (TF) into the firmware, these functions can be loaded dynamically by cloud administrators into the Darkroom runtime. This makes it easy to upgrade the system with new or more efficient functions without the need to reflash the servers' firmware.

To support dynamic loading of transformation functions, we must ensure that the code to be loaded is trustworthy. This is because such code will have access to the raw image data submitted by client applications and can potentially compromise that data, e.g., by leaking it from the server. In addition, we must provide mechanisms that allow transformation functions to be uniquely identified in order to ensure that the correct transformation is applied to a given image. Such mechanisms must be able to tolerate modifications in the set of available transformation functions, as these can be installed from the system.

To ensure that the transformation functions installed into the Darkroom runtime are trustworthy, we require that such functions are installed or uninstalled only by trusted cloud administrators. In particular, to install a TF, a cloud administrator must issue a *load* request which must be signed by the administrator key (KA). Darkroom validates the request against the public part of the authorized admin key $KA+$ and allows the operation to proceed if the test

passes; otherwise the operation is aborted. To uninstall a TF from the system, the corresponding *unload* request must also be signed by a trusted cloud administrator. To allow for unique identification of a given TF, Darkroom leverages the hash of the TF binary. This binary is included into the (signed) load request provided to Darkroom. This request contains additional meta-data that indicates the TF name and the type of input parameters, e.g., *rotation (int degree)*.

D. Performing Image Transformation Operations

Normally, performing image transformations involves a three-stage life cycle. First stage is to securely *upload* a security-sensitive image from the client to the cloud server. In this process, we must ensure that the image is encrypted in such a way that it can only be decrypted within the Darkroom runtime. Second stage corresponds to *transforming* the image by invoking a transformation function of Darkroom. Note that one or more transformations can be chained together (e.g., rotation followed by scaling, etc.). Third stage is to *download* the result of the transformed image while allowing that only the client is able to decrypt the resulting image. In addition, the client must be able to trace the authenticity of the transformation sequence in order to ensure that the resulting image derives from a valid sequence of transformations properly applied to the original image submitted by the client to the cloud provider.

But before explaining these stages, we must describe a necessary pre-configuration step. In particular, Darkroom must be set up with a public key pair called *service key* (KS). The role of this key is to associate the image transformation operations to the context of a specific cloud service (whose logic runs in the rich OS and client). Furthermore, it also aims to allow image transformations to be performed on any of the cloud servers properly configured with a root key. To this end, cloud administrators must generate a public key pair KS and securely load it to the Darkroom runtime of each cloud server. Security is achieved by mutually authenticating the cloud administrator's key with the server's root key. Once the service is loaded into each server, it is now possible to perform the following three stages:

1. Image upload: To securely upload the image to the cloud server, the client simply generates a symmetric key KI and encrypts the image with that key. Then, KI is encrypted with the public key KS^+ to ensure that the image can be decrypted only by Darkroom-enhanced servers allocated to the service to which KS^+ is bound. For integrity verification, the client also computes the HMAC of the message using key KI. The resulting blob $\{I\}_{KI} hmac\{KI\}_{KS^+}$ is then uploaded to the cloud service. We name this blob: *envelope*.

2. Image transformation: Whenever the cloud service needs to perform some specific transformation to the encrypted image, it makes a request to Darkroom by invoking a specific system call (through the Image Cloud Service). Through this system call, the service provides as input the image envelope, the ID of the transformation function to be invoked, and any additional parameters required by the transformation function. The system call issues a world switch to Darkroom, which performs three steps: 1) obtains the encryption key KI by decrypting it with the private part of the service key (KS^-), 2) based on KI, recomputes the HMAC of the message and validates the message integrity, and 3) decrypts the image using KI. If all goes well, Darkroom executes the requested transformation function and produces another envelope containing the resulting image encrypted with the symmetric key KI (which means that only the original message owner and Darkroom-enhanced servers will be able to decrypt the resulting image). To be able to trace the transformations that were applied to a given image, Darkroom builds a cryptographic-protected log which is included in the resulting envelope. The log contains a hash chain of the history of transformations applied to the image.

3. Image download: Finally, the last stage in the image transformation life cycle is to download the resulting image to the client and recover it. Since the client has access to the key KI, it can perform the same sequence of steps as Darkroom in order to validate the integrity of the image and decrypt it. In addition, since the envelope contains a history of transformations performed to the image, it is possible to verify that the received image has resulted from a sequence of transformations applied to the original image. (For space constrains, we omit the details of this verification.)

III. Implementation

We implemented a prototype of Darkroom for the Freescale NXP i.MX53 Quick Start Board. A fundamental concern when building our system was to keep a small Trusted Computing Base (TCB), which essentially consists of the components that live in the secure world: Darkroom Kernel, Cryptography Engine, and Image Processing Engine. Next, we describe the most relevant implementation details of the components of our prototype.

The Darkroom kernel is a fundamental component of our system. It lives in the secure world and is responsible for memory management, thread execution, and context switch operations between worlds. To reduce the chance of code vulnerabilities and keep the kernel size small, we adopted the Genode [11] framework to build our secure world kernel. Genode contains a custom kernel called *base-hw* which runs in the secure world and has a codebase of 20 KLOC (thousand lines of code), which is much smaller than the Linux kernel. In addition, Genode implements a Virtual Machine Monitor (VMM) which can manage a paravirtualized full-featured operating system running in the normal world. In the normal world, we run a paravirtualized Linux kernel, custom-made for Genode. This is because resources such as the framebuffer, and signals such as the data abort interrupt must be trapped and managed by the secure world.

Name	Description
T1	Grey-scale transformation
T2	Color invert transformation
T3	Color swap transformation
T4	90 degree rotation transformation
T5	180 degree rotation transformation
T6	Mirror transformation

Table I
DESCRIPTION OF THE TRANSFORMATION FUNCTIONS IMPLEMENTED.

Genode also provides basic mechanisms for context switching. The VMM saves the processor state (registers and stack) when a `SMC` instruction is executed and restores this state whenever the control is given back to the normal world. Although Genode implements a VMM for the normal world OS, the secure world needs to identify the source of `SMC`. To solve this problem we used CPU registers to send arguments for the secure world to interpret. Since the VMM saves the state of the processor before switching security contexts, the secure world can read the saved register values and interpret them as arguments.

To communicate between worlds, we adopt a shared memory strategy. ARM TrustZone provides low-level mechanisms that allows for memory regions to be securely shared. In Darkroom, we use the so called *watermarking* feature implemented by the i.MX53 QSB board to protect secure world's memory regions from normal world accesses. However, using this feature it is possible to allocate a region of memory in the normal world and access it in the secure context once the secure world controls the execution. This effectively creates a shared memory region between both worlds which can be used for the exchange of messages. We leverage this mechanism in Darkroom as follows. Whenever the system call is invoked by an image server, we allocate a memory region with the same size as the image to be processed using `kmalloc`. After allocating this memory region the `virt_to_phys` function is used to translate the array's virtual address to a physical address so it can be used by the secure world to retrieve the image.

Before triggering the `SMC` instruction, the array's physical address is written to register one (`r1`) so it can be used by the secure world. Darkroom must then copy the contents of the shared memory region to a secure memory region so it can be processed and avoid exposing the decrypted data to normal world memory regions.

The Cryptographic Engine runs on top of the secure world kernel and must support the management of both symmetric and asymmetric cryptographic keys and implement core cryptographic algorithms, including a random number generator. In order to maintain a reduced code base, we used the AES-128 implementation from the *mbed TLS* library [12] as the symmetric key encryption and decryption algorithms. We also adapted RSA from *mbed TLS*.

The Image Processing Engine runs on top of the Darkroom kernel and manages the transformation functions

```
for(i = 0; i < length(oldp); i++) {
    color = oldp[i];
    alpha = (color >> 24) & 0xff;
    red   = (color >> 16) & 0xff;
    green = (color >>  8) & 0xff;
    blue  =  color & 0xff;
    lum = (red * 0.299 + green * 0.587 + blue * 0.114);
    newp[i] = (alpha<<24) | (lum<<16) | (lum<<8) | lum;
}
```

Listing 1. Sample code of gray-scale transformation function.

loaded into the system. These functions are responsible to effectively process the image data sent from the client in an isolated environment managed by the kernel. In our current prototype, we implemented a small set of simple transformation functions just to demonstrate the feasibility of our approach. In a real world setting, more sophisticated functions could be developed in order to serve the needs of external services such as image managing websites, social networks and personal record management services. The transformation functions currently implemented in our system are listed in Table I. All transformations offered by this component were implemented from scratch without having to rely on any image library. Listing 1 provides the sample code of a transformation function to change the color palette of the image to gray scale.

IV. EVALUATION

To study the performance of our prototype, we measure the execution time of the image transformations it supports through micro-benchmarking. To be able to do a more in-depth analysis of these numbers, we performed measurements both in the normal and secure worlds. Our evaluation testbed consisted of an i.MX53 Quick Start Board, featuring a 1 GHz ARM Cortex-A8 Processor, and 1 GB of DDR3 RAM memory. The board executed our system from a mini SD card, which was flashed with the modified Genode and Linux versions. For each experiment, we report a mean of 50 runs, and provide no standard deviation values because these were negligible (less than 3 milliseconds). We also stress that the results recorded in these experiments were collected after both the normal and secure world components were compiled with the O3 optimization flag.

Table I presents the six different image transformations supported by our system, as well as the names we picked to identify them during our result analysis. To get a clear perspective of the performance costs of these transformations, we chose to measure their execution time over a single image. The image we picked is 1024x1024 pixels, not only because it is a resolution supported by many of today's mobile devices, but also because it is a common resolution for network computing devices.

Figure 2 shows the execution times of all these transformations. As we can see, the grey-scale transformation (T1) is by far the most expensive. This happens because even though every transformation consists on a simple for-loop with

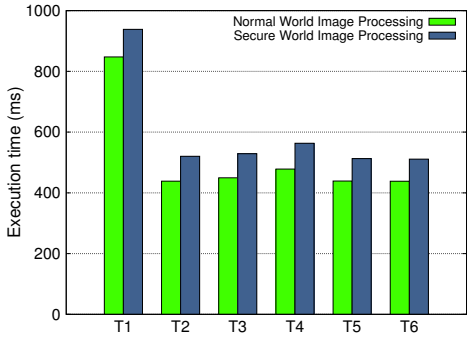


Figure 2. Execution time of all transformations.

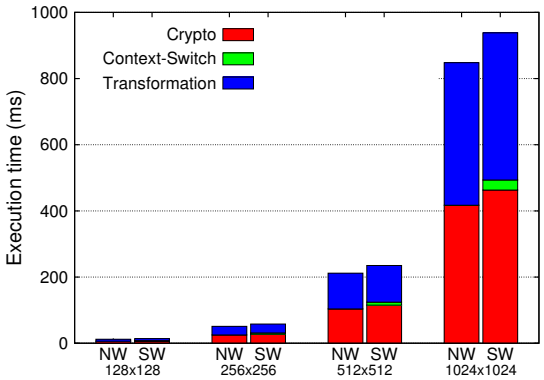


Figure 3. Execution time of the grey-scale transformation.

complexity $O(N \times M)$, where N is the width and M is the height of the image, T1 features additional math and bit shift operations. In addition, since this transformation requires mathematical and bit shift operations, the execution time difference becomes more prominent because the compiler cannot optimize it as much as the other transformations. Regarding the execution time differences between both worlds, we observe an almost constant penalty in the secure world, associated to the world switch operation, but mostly to buffer allocations necessary to share data between worlds.

To study the performance variations of these transformations, we measured the execution time of transformation T1 for different image resolutions. We focused on T1 since it is the most computationally demanding of all transformations supported by the system. For consistency reasons, we downgraded the original 1024x1024 picture resolution and tested T1 with three additional lower resolutions. Figure 3 shows the impact of T1 on a 128x128, 256x256, 512x512, as well as on the original 1024x1024 picture. The execution time is broken up into three parts: the transformation itself, context-switch, and cryptographic operations. As we can see, the cryptographic operations, more specifically decryption of input image and encryption of the resulting image, are slower in the secure world than in the normal world. This happens because the measurements in the secure world involve copying contents from a world-shared buffer to a buffer in secure memory, and then from this buffer back to the shared buffer after the transformation. On the other hand, the context-switch times are almost negligible, corresponding to the times taken by the system to execute our custom syscall, allocate memory for the shared buffer, translate a virtual to a physical address, and call `SMC`, which triggers the context-switch. In this case, the larger the image, the longer it takes to allocate the shared memory region, which explains why we see an increase in the time the context-switch takes from smaller images to bigger images. Regarding transformation times, we can see the execution times between the normal and secure worlds are very similar.

V. Related Work

Over the past years, extensive work has been focused on data security for untrusted cloud-hosted storage services. Systems such as Venus [13] or DEPSKY [14] protect data confidentiality by relying exclusively on cryptographic techniques performed at the client side. This approach tends to be attractive for users because it precludes the need to trust in specific components controlled by the cloud provider. The downside of this approach, however, is that encrypted data cannot be processed by applications, e.g., for performing image transformations, which reduces the applicability of this technique in the cloud. To overcome this limitation, researchers have studied ways to allow for encrypted data processing by leveraging advanced cryptographic techniques. CryptDB [15] is a representative example of such a system which employs homomorphic encryption to enable SQL query processing over encrypted relational databases. Nevertheless, systems such as CryptDB tend to limit the functions that can be executed, introduce considerable performance overheads, and depend on weaker cryptographic schemes when compared with traditional ones.

An alternative approach to securing cloud processing is to leverage Intel's upcoming technology Security Guard Extensions (SGX). SGX is a set of hardware extensions to the Intel architecture which enables processes to maintain secure address space regions. These regions are called *enclaves* and provide a Trusted Execution Environment (TEE) for running security-sensitive application code in isolation from the OS. Although this approach requires users to trust the SGX-enabled hardware deployed by the cloud provider, enclaves can run arbitrary functions at native processor speed, hence faster than homomorphic encryption schemes, and provide strong security properties. In particular, enclaves' internal state cannot be accessed neither by privileged system code nor through memory probe attacks. Projects such as Haven [16] and V3 [17] have started to explore ways to run unmodified legacy code and verified code, respectively, inside enclaves. However, some fundamental limitations need to be overcome in order to make this technology fully practical and robust [16]. Thus, similarly to SGX, we take a TEE-based approach in the design of Darkroom, but explore the use of ARM TrustZone technology, which is more mature than SGX and available in commodity hardware.

Given that the majority of mobile devices are equipped with ARM processors, ARM TrustZone has been studied mostly to overcome security issues on mobile platforms. Many authors propose solutions based on TrustZone-enabled TEE for hosting mobile security services, which allow for: detecting and preventing mobile app ad frauds [18], implementing OS introspection mechanisms [7], enabling secure storage of sensitive data [4], providing secure authentication mechanisms [5], implementing one-time-password solutions [6], or providing forensic tools for trusted memory acquisition [8]. Other systems provide general-purpose frameworks for splitting mobile app code and run it in the TEE [19] or enabling trusted I/O between the user and TrustZone-based services [20]. Existing systems, however, are not directly applicable to the cloud setting due to the cloud's specificity in terms of application requirements and system administration model. Brenner et al. [21] took some first steps to using ARM TrustZone on the cloud by building a TEE-protected privacy proxy for Zookeeper. Their system provides a confidential coordination service for distributed applications, which constitutes a different application scenario than the focus of our work.

VI. Conclusion

We presented Darkroom, a system that leverages ARM TrustZone to bootstrap trust in a cloud based image processing service. Darkroom provides clear isolation between a potentially compromised cloud server OS and a smaller trusted execution environment and guarantees that all users' data stored or processed in a cloud server is handled by a smaller code base. For Darkroom, we designed a set of cryptographic protocols that builds trust when cloud administrators add dynamic code in the server, and provides integrity and authenticity properties to users' image processing requests to cloud servers. The image processing performance in Darkroom incurs a reduced time penalty.

Acknowledgments: This work was partially supported by the EC through project H2020-645342 (reTHINK), and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 (INESC-ID).

References

[1] BBC, "FBI investigates 'Cloud' celebrity picture leaks," http://www.bbc.com/news/technology-29011850.

[2] N. Security, "Google Drive security hole leaks users' files," https://nakedsecurity.sophos.com/2014/07/10/google-drive-security-hole-leaks-users-files.

[3] C. Weekly, "Why unfair contract terms put end-user trust in cloud at risk," http://www.computerweekly.com/blog/Ahead-in-the-Clouds/Why-unfair-contract-terms-put-end-user-trust-in-cloud-at-risk.

[4] X. Li, H. Hu, G. Bai, Y. Jia, Z. Liang, and P. Saxena, "DroidVault: A Trusted Data Vault for Android Devices," in *Proc. of ICECCS*, 2014.

[5] D. Liu and L. P. Cox, "Veriui: Attested Login for Mobile Devices," in *Proc. of HotMobile*, 2014.

[6] H. Sun, K. Sun, Y. Wang, and J. Jing, "TrustOTP: Transforming Smartphones into Secure One-Time Password Tokens," in *Proc. of CCS*, 2015.

[7] X. Ge, H. Vijayakumar, and T. Jaeger, "Sprobes: Enforcing Kernel Code Integrity on the TrustZone Architecture," in *Proc. of MoST*, 2014.

[8] H. Sun, K. Sun, Y. Wang, and J. Jing, "Reliable and Trustworthy Memory Acquisition on Smartphones," *Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2547–2561, 2015.

[9] B. Yang, K. Yang, Y. Qin, Z. Zhang, and D. Feng, "DAA-TZ: An efficient DAA scheme for mobile devices using ARM TrustZone," in *Trust and Trustworthy Computing*, 2015, pp. 209–227.

[10] ARM, "ARM Security Technology – Building a Secure System using TrustZone Technology," ARM Technical White Paper, 2009.

[11] "The Genode OS Framework," http://genode.org/.

[12] "mbed TLS," https://tls.mbed.org/.

[13] A. Shraer, C. Cachin, A. Cidon, I. Keidar, Y. Michalevsky, and D. Shaket, "Venus: Verification for untrusted cloud storage," in *Proc. of CCSW*, 2010.

[14] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "DEPSKY: Dependable and Secure Storage in a Cloud-of-Clouds," in *Proc. of EuroSys*, 2011.

[15] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting Confidentiality with Encrypted Query Processing," in *Proc. of SOSP*, 2011.

[16] A. Baumann, M. Peinado, and G. Hunt, "Shielding Applications from an Untrusted Cloud with Haven," in *Proc. of OSDI*, 2014.

[17] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich, "VC3: Trustworthy Data Analytics in the Cloud Using SGX," in *Proc. of IEEE S&P*, 2015.

[18] W. Li, H. Li, H. Chen, and Y. Xia, "Adattester: Secure Online Mobile Advertisement Attestation Using Trustzone," in *Proc. of MobiSys*, 2015.

[19] N. Santos, H. Raj, S. Saroiu, and A. Wolman, "Using ARM Trustzone to Build a Trusted Language Runtime for Mobile Applications," in *Proc. of ASPLOS*, 2014.

[20] W. Li, M. Ma, J. Han, Y. Xia, B. Zang, C.-K. Chu, and T. Li, "Building Trusted Path on Untrusted Device Drivers for Mobile Devices," in *Proc. of APSys*, 2014.

[21] S. Brenner, C. Wulf, and R. Kapitza, "Running ZooKeeper Coordination Services in Untrusted Clouds," in *Proc. of HotDep*, 2014.

2016 IEEE 35th Symposium on Reliable Distributed Systems Workshops

Thwarting Data Exfiltration by Repackaged Applications

Daniel Andrade
INESC-ID
Lisbon, Portugal
Email: daniel.andrade@ist.utl.pt

Thor Kristoffersen
Norsk Regnesentral
Oslo, Norway
Email: Thor.Kristoffersen@nr.no

Ivar Rummelhoff
Norsk Regnesentral
Oslo, Norway
Email: Ivar.Rummelhoff@nr.no

Alex Gerdov
OS New Horizon
Tel Aviv, Israel
Email: alexg@osnewhorizon.com

João Nuno Silva
INESC-ID,
Instituto Superior Técnico,
Universidade de Lisboa
Lisbon, Portugal
Email: joao.n.silva@inesc-id.pt

***Abstract*—Android applications are subject to repackaging attacks, where popular applications are modified, often by inserting malicious logic, re-signed, and then uploaded to an online store to be later on downloaded and installed by unsuspicious users.**

This paper presents a set of protocols for increasing trust in special-purpose Android applications, termed secured trusted applications, during communication with a trustworthy external hardware device for storing sensitive end user data, termed secured personal device. The proposed approach requires neither operating system modification nor root privileges.

The evaluation of our solution shows that the authenticity and integrity of applications, and the confidentiality and integrity of communication, is ensured as long as Android operates correctly.

I. Introduction

According to the Nokia Threat Intelligence Laboratories, in the second half of 2015, smartphones accounted for 60% of the infections detected in mobile networks and, in December 2015, 0.3% of smartphone devices exhibited signs of malware infection. The main platform targeted is Android [1].

Google Play, the most popular Android market, employs a service, codenamed Bouncer [2], that analyses new applications, and applications already in the market, for malicious software. However, this service is not infallible [3]. Tools with code obfuscation capabilities, such as ProGuard [4] and DexGuard [5], increase the difficulty of reverse engineering applications, but do not make it impossible.

In a previous study, authors gathered 1260 malware samples, from multiple markets, and compared the package name of each collected sample with the package names of applications on the official Android market, in an attempt to quantify the use of repackaging techniques. When a match was found, the application was manually analyzed to ascertain whether it was repackaged. 86% of the 1260 malware samples were found to be repackaged applications [6].

The presented set of protocols aim to thwart repackaged applications from accessing a custom-built external hardware device and the sensitive end user data it stores. This applies only to special-purpose Android applications, termed *secured trusted applications* (STA), and during communication with a trustworthy external-storage hardware device, termed *secured personal device* (SPD). Notice that this scenario is materially different from the usual Android application security case. The objective is not the security of the data contained in the application private storage in the smartphone. There is a form of pairing between the smartphone and the SPD, and what we propose is to protect that link and the data contained in the SPD. The proposed approach requires neither operating system modification nor root privileges, but requires the Android OS to be trusted as well as Google Play.

II. Background

The next sections offer an introduction to the *Personalised Centralized Authentication System* (PCAS[1]), which the SPD is a part of, and of the Android security mechanisms upon which the protocols are built.

A. PCAS

The aim of PCAS is to provide end users with a trustworthy handheld device—the SPD—to securely authenticate, store data and share it with trusted applications. The SPD takes the form of a smartphone add-on and uses the communication services of the smartphone to reach the other nodes in PCAS.

1) Nodes: The system is composed of multiple nodes as depicted in Figure 1.

a) Phone: The smartphone is a generic consumer device connected to the Internet via Wi-Fi or the cellular network, and running a non-rooted and up-to-date version of the Android OS. The PCAS system software (PSS) and the secured trusted applications run within the smartphone.

[1] https://www.pcas-project.eu/

978-1-5090-5259-2/16 $31.00 © 2016 IEEE
DOI 10.1109/SRDSW.2016.18

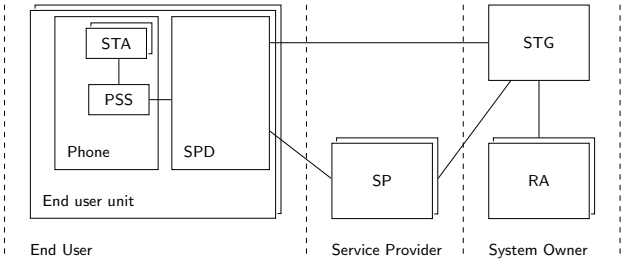


Fig. 1. Logical overview of nodes composing the PCAS system.

b) SPD: The SPD is a hardware device built specifically for PCAS. It contains its own sensors for biometric authentication and is tamper resistant. The smartphone and the SPD are under the control of the end user.

c) SP: A Service Provider (SP) is composed of an STA and a remote web service. The STA runs on the smartphones of end users, and can access the web service of the SP as well as an isolated secure partition for data storage in the SPD.

d) STG: The Secured and Trusted Gateway (STG) is a remote server controlled by the system owner. The STG registers SPDs, and manages their identity and security information. The public key infrastructure of PCAS is managed by the STG.

e) RA: The Registration Authority (RA) is a computer, provided by the system owner to registration authorities to perform end user enrollment.

2) Stakeholders: The main stakeholders are the end users, the service providers, the system owner and the registration authorities. Each end user operates a unit composed of a smartphone paired to an SPD. The service providers offer services to end users via STA and web service pairs. The system owner oversees the entire system and provides third-party entities, the registration authorities, with the means to enroll new end users into PCAS.

3) Communication: An end user acquires an SPD and enrolls at a registration authority before being granted access to the PCAS ecosystem. During enrollment, the end user installs the PSS on the smartphone, pairs the smartphone to the SPD, and trains the SPD to recognize its owner by creating a biometric template. After a successful enrollment, the end user can install and use secured trusted applications, which interact with the respective SP and with the SPD.

4) Security: PCAS relies on public key cryptography, and its own public key infrastructure, to secure the communication between nodes. The smartphone operating system and regular applications are outside of the PCAS domain. However, the PSS on the phone is part of the PCAS management domain, and the STAs are part of the Service Providers management domain.

B. Android OS

The Android operating system is an open-source software stack based on the Linux kernel and geared primarily towards touchscreen mobile devices. Android includes a large selection of applications via application stores, such as Google Play, and these mobile applications are developed on top of the application framework. The application framework reaches the system services through Binder, the chief inter-process communication mechanism in Android. This section describes key security features of Android [7] applied to the design of the protocols on the sections that follow.

Applications must be digitally signed before they can be installed, but the certificate does not need to be signed by a certification authority. The certificate is used to identify the author of an application and make sure future updates are coming from the same entity (same-origin policy); however, this does not imply the author can be trusted. In addition, multiple applications can share data in a secure manner when signed by the same private key, taking advantage of signature-based permissions.

The `PackageManager` manages applications in Android, independently of how they are installed, and keeps a database of installed applications that includes their signing certificates and granted permissions. An application can safely acquire the signing certificate of another application via the package manager.

Permissions in Android are divided into four categories (or levels): normal, dangerous, signature and signatureOrSystem. PCAS makes use of both dangerous and signature permissions. Dangerous permissions must be requested by an application, and explicitly accepted by the end user, before accessing the resource being protected. Signature permissions are only granted when the requesting application is signed using the same private key as the application declaring the permission.

The `AndroidKeyStore` is both a keystore—a storage facility for cryptographic keys and certificates—and a security provider. The Android Keystore prevents the extraction of key material by application processes. Depending on the smartphone model, key material may be bound to secure hardware, in which case the keys cannot be extracted even when the operating system is compromised or an attacker can read the internal memory of the phone. However, the attacker would still be able to use the keys even if the key material cannot be exfiltrated. The keystore enforces access controls to mitigate this scenario: each key is given a set of authorized uses by the owning process, during key import or generation, that remains immutable during the lifetime of the key. When an application is uninstalled, the associated keys are automatically deleted from the keystore. The Android security provider also supports the generation of keys for symmetric and asymmetric encryption.

III. Threat Model

The objective is to protect the end user data stored in the SPD. PCAS gives strong guarantees to the remote SPs that the end user behind a specific Phone–SPD pair is the owner of the device, but this falls outside of the scope of this article.

The Android OS is trusted, including the application framework, the system services, and the Linux kernel. Google Play must also be trusted since that is where the end user downloads the PSS and the STAs.

Attackers may be looking for sensitive end user information or the glory that comes with breaking into a system; want to disrupt the system; or are financially motivated. An adversary may control the network or manage to install malicious applications on the smartphone. The end user may unknowingly facilitate attacks by installing software from untrusted sources.

IV. Notation

The notation used to describe the security protocols is the following:

K_x, K_x^{-1}	Respectively public key and private key for principal x
$gen \rightarrow K_x, K_x^{-1}$	Generation of a new key pair
$\{t\}K_x^{-1}$	Term t signed with private key K_x^{-1}
$verify(\{\{t\}K_x^{-1}\}K_x)$	Verify signed term t
$hash(t)$	Hash of term t
$store(t)$	Store term t locally
$lock(t)$	Lock term t into read-only mode
$compare(x, y)$	Compare terms x and y for equality

The principals are: the STG, G; the set of SPs, S_i; the set of SPDs, D_j; the set of STAs, A_l; and the set of Phones, P_m. Keys include not only the name of the principal but also the role of the key:

$K_{G.root}$	Self-certified key for certification of intermediate keys
$K_{G.ccode}$	Certified by $K_{G.root}$ for certification of code signing keys
$K_{S_i.code}$	Certified by $K_{G.ccode}$ for data origin authentication of STA
$K_{S_i.store}$	Self-certified key for same-origin policy of applications in Google Play
$K_{D_j.phone}$	Self-certified key for communication with Phone
$K_{P_m.spd}$	Self-certified key for communication with SPD

V. PCAS System Software

The PCAS system software is composed of multiple components, including the PCAS application (PA), the PCAS service (PS) and the Remote UI (RUI; [8]). The PA is a management application for the PCAS software on the smartphone. The PS is the core PCAS component running on the smartphone. The RUI provides the SPD with the means to interact with the end user by displaying content on, and receiving input from, the touchscreen.

The system must strive to ensure the authenticity and the integrity of these modules. In addition, the confidentiality and integrity of the communication between the PCAS service and the SPD must be ensured since it is via this service that all the PCAS components on the phone are able to interact with the other nodes in the PCAS system. These requirements are achieved by the proper installation of the PSS components, the subsequent exchange of keys between the PS and the SPD, and the use of custom Android permissions.

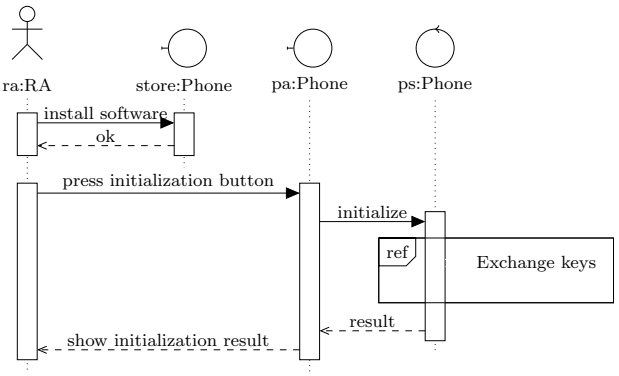


Fig. 2. Installation and initialization of system software on the smartphone.

A. Installation

Figure 2 depicts the installation and initialization of the PSS on the smartphone. The first step is to make sure the software is authentic and for that it must be installed by the Registration Authority directly from Google Play. When the software is already present on the smartphone, the Registration Authority uninstalls the current version of the PSS and proceeds with a clean installation from the official provider.

To begin the initialization process, the RA, or the end user under the supervision of the RA, opens the PA and presses the setup button. A message is then delivered from the PA to the PS, where the initialization protocol takes place.

B. Exchange Keys

Figure 3 depicts the exchange of communication keys between Phone and SPD, which is composed of the following steps:

1) The smartphone generates and stores a new key pair, $K_{P_m.spd}, K_{P_m.spd}^{-1}$, for communication with the SPD using both the Android Keystore type and provider.
2) The public key of the PS, $K_{P_m.spd}$, is sent to the SPD. The SPD stores this key. The SPD generates and stores a key pair, $K_{D_j.phone}, K_{D_j.phone}^{-1}$, for communication with the phone. The public key of the SPD, $K_{D_j.phone}$, is sent within the reply to the PS. The PS stores the public key of the SPD using the Android Keystore.
3) The phone sends a message to the SPD confirming the received public key of the SPD is stored. The SPD then locks the public key of the PS.

The public key of the PS is only locked after the phone confirms it has stored the public key of the SPD, because if a failure occurs then the initialization process can safely be restarted. The public key of the PS is accepted by the SPD on a first-come first-served basis, i.e. the first key to arrive is locked and cannot be changed without re-enrolling the end user at the Registration Authority.

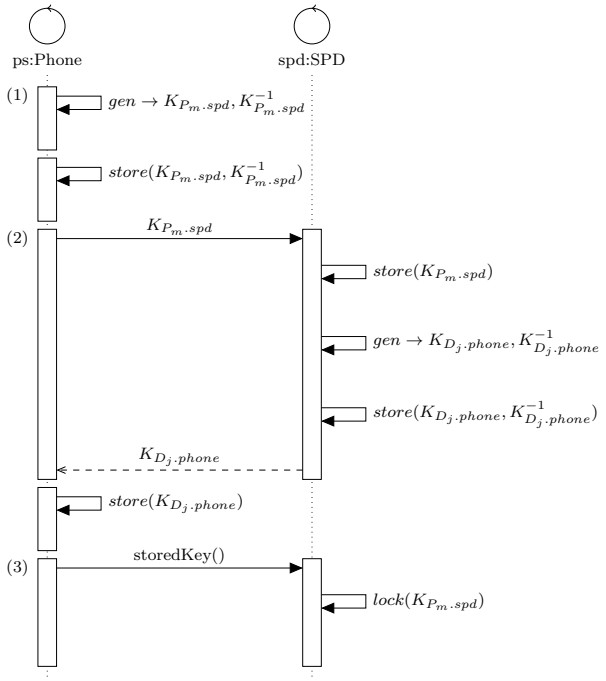


Fig. 3. Exchange of communication keys between Phone and SPD.

C. Permissions

PS declares multiple permissions to protect its interfaces:

- For each system module there is an individual signature-level permission that is requested before access is granted to operations on the respective interface in the PS. The PCAS system modules are all signed by the same private key. The signature guarantees the authenticity of the system module since a malicious component would not be signed by the same private key as the PS. A module that is not signed by the same key as the PS cannot successfully access system operations.
- A dangerous-level permission protects the interface exposed to secured trusted applications. Each STA requests this permission, which the end user must explicitly accept, before access is granted to PCAS operations. The end user only needs to explicitly grant, or deny, the permission once for each STA; the decision can be saved by the operating system and updated using the application manager of the phone.

VI. Secured Trusted Applications

The authenticity and integrity of the secured trusted applications is ensured by having each STA relying on the respective SP as trust anchor as well as in the PS. Trusting the PS is only meaningful after initializing the PCAS system modules as presented in Section V.

Figure 4 depicts the process for verifying the authenticity and integrity of an STA; the sequence diagram assumes the STA and the required system software are installed on the smartphone. This process is composed of the following steps:

1) The STA requests an operation from PCAS by sending the request to the PS.
2) The PS retrieves the certificate that signed the STA, $\{hash(A'_{l_v})\}K^{-1}_{S_i.store}$, and sends it to the SPD in the operation request header.
3) The SPD requests the code signing public key from the Service Provider as well as the signature, $\{hash(A_{l_v})\}K^{-1}_{S_i.code}$, for this version of the STA. The SPD then validates the signature and compares the hashes received from both the PS and the SP.
4) The SPD replies to the PS and included in the response header is the response status. A successful status indicates the operation body can now be exchanged between the PS and the SPD. Otherwise, the operation body is not exchanged and the status indicates the failure reason (e.g. hash mismatch).
5) Finally, the operation response is sent to the STA.

The SPD cannot directly calculate the hash of an STA because it cannot access applications on the smartphone. For this reason, the smartphone sends the code signing certificate of the STA to the SPD. The SPD then extracts the hash from the certificate and uses it in its comparison.

VII. Security Evaluation

Under the conditions set forth in Section III, and when operating the protocols described in Section V and Section VI, the proposed solution offers the following properties:

- peer entity authentication between the PS and the SPD,
- confidentiality and integrity of messages exchanged between the PS and the SPD,
- authenticity and integrity of the PSS, and
- authenticity and integrity of the STAs.

This section offers an informal security evaluation to increase confidence in the protocols. Side-channel attacks are ignored.

A. Communication Security

The authenticity, integrity and confidentiality of the communication between the PS and the SPD is ensured using a two-way TLS connection set up using the keys exchanged during enrollment. This assumes that the smartphone was not already compromised at that point. Each node uses exclusively the public key certificate of the other node as trust material during TLS handshakes.

B. Systems Security

1) PSS Authenticity and Integrity: The authenticity and integrity of the PSS on the smartphone is achieved in two steps. Ensuring first the authenticity and integrity of the PS, and then of the other PSS components.

The authenticity and integrity of the PS on the smartphone is initially assured because it is installed from Google Play at the RA. After the exchange of communication keys protocol is executed, the possession of the private key, related to the public key locked in the SPD, confirms the authenticity and

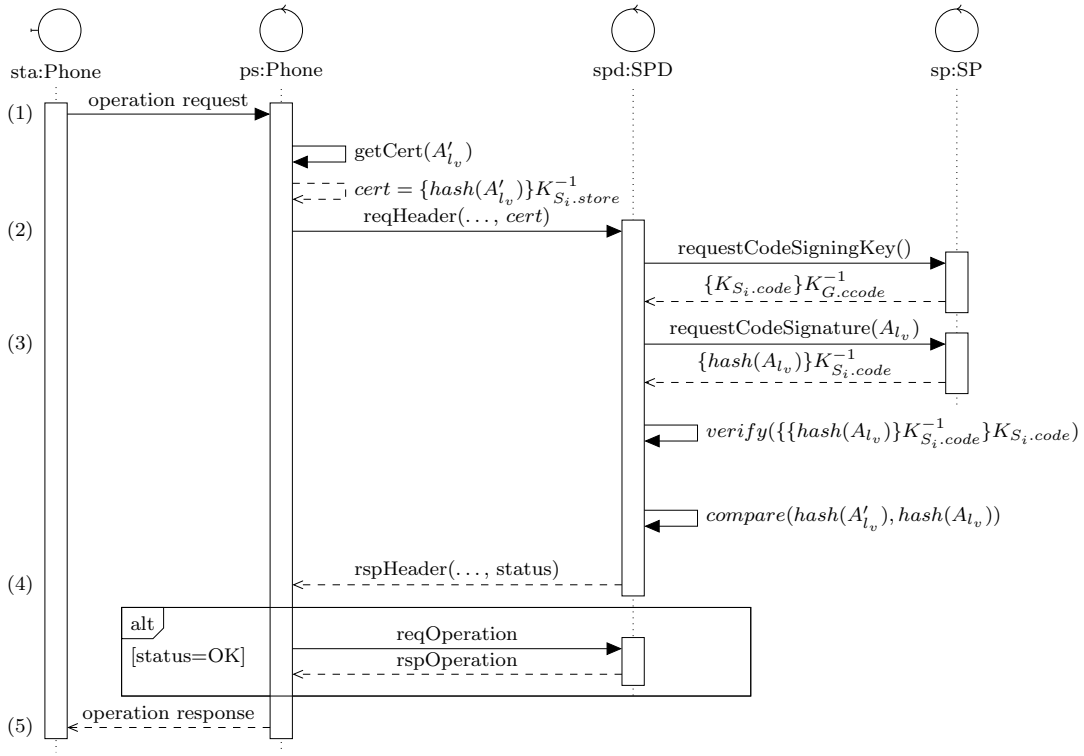


Fig. 4. Verification of authenticity and integrity of an STA.

integrity of the PS in future events. Recall from Section II-B that when an application is uninstalled, the associated keys are automatically removed by the operating system. If the PS itself is compromised, the attacker will be able, with sufficient time and effort, to exfiltrate all data stored in the SPD.

The other PSS components are digitally signed by the same key that signs the PS. Access to interfaces exposed by the PS to the PSS components, require the caller to request a custom signature-level permission declared by the PS. The Android OS itself blocks access when the signatures of caller and callee are different. Consequently, if a PSS component other than the PS is compromised, that component will no longer be able to interact with the PS and the data remains safe.

2) STA Authenticity and Integrity: The code signing certificate of the STA, $\{hash(A'_{l_v})\}K^{-1}_{S_i.store}$, is self-signed. The signature is not verified in the SPD since it adds no additional security. The package manager of Android, and the PCAS service, are both trusted. The package manager retrieves the code signing certificate of an application and the PCAS service sends the certificate to the SPD for verification. The SPD requests the certificate of the application, $\{K_{S_i.code}\}K^{-1}_{G.ccode}$, from the Service Provider, whose authenticity is guaranteed by the PCAS PKI, and compares it with the certificate received from the smartphone. If the application hashes stored in the certificates are not equal, then the hash comparison returns a negative result and the operation is refused by the SPD. When an STA is compromised, only the data associated with that SP can be potentially leaked.

C. Vulnerabilities

The current solution has some weaknesses, in addition to any flaws in the implementation and operation of the system. Two issues of concern are having Google Play as part of the trusted computing base, and the potential for denial of service attacks.

1) Large TCB: The trusted computing base includes not only the Android OS but also the Google Play store. Removing Android OS from the TCB is unfeasible without the use of trusted hardware. The trust in Google Play, however, can be reduced or entirely eliminated.

There is one key point in time when the potential for abuse through Google Play exists: during Phone and SPD pairing during user enrollment. Assuming Android OS is trusted, and excluding software bugs, the PS can only be compromised at this stage.

An STA can be verified for both authenticity and integrity according to the protocol detailed in Section VI, but there is no trusted entity to perform the verification of the PS and the PS cannot verify itself.

As said in Section VII-B1, there is an assumption that software installed from Google Play is correct. During the installation of an application, the Android OS verifies the integrity of the APK; and future updates to this APK have to be signed by the same private key. However, it is possible

for a potentially malicious online store to repackage an APK. The signing key would be different, but Android does not check whether the key is the correct one for a given software provider.

There are three ways to remove Google Play from the TCB: verify the fingerprint of the PS manually; install the PS from an alternative, trusted, application distribution service; or install the PSS modules from different sources.

a) Manual fingerprint check: Checking the fingerprint after the installation of the PS can be achieved by using a third-party application to display it on the smartphone screen, or by pulling the installed APK from the smartphone using a tool such as the Android Debug Bridge [9] and then extracting the certificate from the software bundle. The fingerprint can then be compared with a version obtained from a secure source such as the software vendor website. If using a third-party application to display the fingerprint of the certificate of the APK, then it should be downloaded from a location other than Google Play; otherwise, this application could also be compromised.

b) Different application distribution service: Instead of installing the PS from Google Play, it can be installed from an alternative, trusted, distribution service. A possibility is to use a different application market, but the PS can also be sideloaded, for instance by the RA. However, this does not really solve the problem since in practice we are replacing trust in Google Play with trust in the new software source. The STG is the obvious candidate for providing a distribution service, since it is already trusted.

c) Distributed application distribution service: The PSS modules are installed from two—possibly more—application distribution services. Recall from Section II-B that the Android OS throws an error during the installation of an application, if an already-installed application, signed by a different private key, declares the same permission.

The individual components of the PSS are all signed by the same private key and make use of custom signature-level permissions to communicate with the PS, which is the core PSS component. If the PS is installed from Google Play, and the PA is installed from a different application market, then there is an implicit check during application installation in Android on whether both applications are signed by the same private key. If the keys are different, then at least one of the distribution services is compromised; if the signing keys are the same, then either both application distribution services are functioning correctly or both are compromised by the same entity.

This approach replaces one-hundred-percent trust in a single distribution service by partial trust in two distribution services; and can be extended to include more than two distribution services, to further dilute the trust, since there are more than two PSS components. Following this principle, all application distribution services would have to be compromised in order to successfully replace the legitimate PSS modules by malicious versions.

2) DoS: The system expects the PS to always be present on the smartphone after a successful execution of the exchange of communication keys protocol described in Section V-B. If the PS is uninstalled, then the SPD public key certificate is removed from the Android Keystore by the Android OS. This, in turn, forces the end user to physically go to an RA to re-enroll the unit. In other words, removing the PS results in a denial of service to the end user.

This can be the result of an accidental event, such as the removal of the PS by the end user, or an intentional act, such as the removal of the PS by an entity that gains temporary physical access to the smartphone. In addition, a malicious application installed by the end user can send a PS uninstall order to Android; the end user is provided with a dialog to either accept or reject the service uninstallation but an ignorant end user may accept the request without realizing what is happening.

VIII. Conclusion

This paper presents a set of protocols for increasing trust in the authenticity and integrity of secured trusted applications during communication with PCAS.

Future work involves the implementation, and performance evaluation, of a prototype. And to consider the use of a Trusted Execution Environment (TEE), which is an isolated execution environment running alongside the main OS, to strengthen its security. The use of a TEE may reduce, or fully eliminate, the need for trust in the Android OS.

Acknowledgment

Thank you to Miguel Pupo Correia for his comments on the first version of the protocols. This work was supported by the European Commission under the 7th Framework Programme through FP7-610713—Project PCAS. The author João Nuno Silva was additionally supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

References

[1] N. T. I. Lab, "Nokia threat intelligence report: H2 2015," Nokia, Malware Report, 2016.

[2] H. Lockheimer, "Android and security," https://googlemobile.blogspot.pt/2012/02/android-and-security.html, Feb. 2012, accessed: 2016-06-22.

[3] J. Oberheide and C. Miller, "SummerCon 2012: Dissecting the android bouncer," https://jon.oberheide.org/blog/2012/06/21/dissecting-the-android-bouncer/, Jun. 2012, accessed: 2016-06-22.

[4] E. Lafortune, "ProGuard," http://proguard.sourceforge.net/, accessed: 2016-06-22.

[5] GuardSquare, "DexGuard," https://www.guardsquare.com/dexguard, accessed: 2016-06-22.

[6] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *2012 IEEE Symposium on Security and Privacy*, May 2012, pp. 95–109.

[7] N. Elenkov, *Android Security Internals: An In-Depth Guide to Android's Security Architecture*, 1st ed. No Starch Press, oct 2014.

[8] M. A. Carvalho and J. N. Silva, "Poster: Unified remoteu¡ for mobile environments," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '15. New York, NY, USA: ACM, 2015, pp. 245–247. [Online]. Available: http://doi.acm.org/10.1145/2789168.2795170

[9] Google, "Android Debug Bridge," https://developer.android.com/studio/command-line/adb.html, accessed: 2016-06-22.