Public-Key Distribution and Acquisition services over SMS

Germanas Skurichinas

Kongens Lyngby 2017

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 45 25 30 31
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

In this project we overview development of human communication methods and their transition into digital communication era, from an information security standpoint. We argue that individuals are incompetent at deriving trust in digital communications, in part because of complicated cryptographic systems, as well as a lack of Public-Key Infrastructures. We reason for shifting cryptographic key management responsibilities from individuals to application developers. Further, we propose a Public-Key Infrastructure hosted on SMS channel and define Application Programming Interface to provide a necessary infrastructure required for developers to overtake certificate management.

Preface

This thesis was prepared at DTU Compute in fulfilment of the requirements for acquiring an M.Sc. in Engineering.

The first part of the document outlines importance of Public-Key cryptography in providing secrecy and security in current computer communications. In the other half, a Public-Key Infrastructure solution is proposed to publish and manage Public-Key certificates employing Short Message Service (SMS).

This document is self-contained, includes references to information sources used and images to support the ideas, including third-party images released under free for non-commercial reuse licenses.

Kgs. Lyngby, June 19, 2017

Germanas Skurichinas

Acknowledgements

First of all, I would like to thank my supervisor Christian Damsgaard Jensen, most of all for curating computer security studylines and for teaching many security related topics, that I enjoyed a great deal. Also, I would like to thank him for all valuable conversations that we had, as well for taking up bureaucracy related to this project.

Secondly, I would like to thank my colleagues from NorthernVO company for accommodating my schedule during my studies and thesis writing period as well as for all the moral encouragement.

I would like to express my deep gratitude to Unwire ApS, for introduction to mobile networks and mobile network operator activities, as well as for providing an Android device for testing purposes.

Last, but not least, I would like to thank my colleague students Linas Kaminskas and Filip Magic for proof reading this document and their feedback.

Contents

Chapter 1

Introduction

Throughout the last two millenniums human interaction have gradually shifted from direct communication, which required a physical contact, to a more indirect communication - based on modern computer communication technologies. The pace of change since invention of telegraph has been particularly radical, thus we argue that people were unable to adapt to these technological advances with a comparable competency in relation to pre-modern methods, such as speaking and writing. In particular, humans are not able to justly gauge trust properties of an underlying communication method and subsequently unable to adjust their communication manner to a appropriate for the method used.

Our ambition here is to show that internet protocols and services is falling short to provide equivalent security properties to our general communications in comparison to prior means. In addition, we will explore what relevant mechanisms technology provides to achieve these goals and what parts of internet services infrastructure can be enhanced.

When considering communication means from a computer security standpoint it is customary to describe what security principles (a.k.a security attributes) are implied for a particular communication method. Security principles of interest for this discussion will be:

- *Confidentiality* – secrecy of information in transit, which can include iden-

tity of communicating parties.

- *Integrity* – confidence for information to be carried as intended, which can also provide information restoration and checks whether information has been altered in transit.

- *Availability* – accessibility to: communication channel; communication protocol (knowledge of procedural rules of a method) as well as geographic access; communicating parties; and communicated information.

- *Accountability* – identifiability of information source and possible liability attachments.

- *Trust* – a believe in reliability and truthfulness associated to a communication method, channel and communicating parties.

- *Forward Secrecy* - essentially cryptographic term meaning that exploitation of future communications can't reveal matter of prior communications.

We will briefly review archaic and modern mainstream communication methods to outline major shifts from naturally assumed trust (inherited from communication method) to an artificial trust systems provided by the current technologies.

Further, we will employ computing terminology, when referring to communication methods' details, specifically:

- communicating parties or peers as well as sender and receiver notions, respectively;

- communication channel – physical means for information transit;

- communication protocol – knowledge of rules and procedures in order to convey information through communication channel to another party.

1.1 Pre-Modern Era Communication

For our purposes pre-modern times can be split into two development periods: emergence of spoken language; and creation of writing and symbolism.

1.1.1 Natural (Spoken) Language

Emergence of vocalised language gave an ability for human (communicating party) to convey rich information to the parties in its immediate environment, by using human voice (physical vibrations), thus creating the very first complex communication channel and establishing a new communication method. Natural language, with all it's parts, serves as a communication protocol for exchanging ideas (information), as well as information encoding tool - structuring of information in particular manner, according to protocol.

Physical properties of speech and a fact that only another human could produce verbal speech has limited verbal communications to face-to-face interactions. For better or worse this limitation has created a situation, where both communicating parties have to be physically present, therefore deriving trust in communication equivalent to information sensitivity and trustworthiness of parties present. Following, we can argue that speech provides a considerable level of *integrity* and *accountability* - stemming from mentioned limitations.

Interestingly, the *trust* in communication is comparable in both public and private domains independently and influenced more so by contextual information. In situations pertaining liability, speech would be taken at face value, and in argument as 'your word against mine word' basis.

Natural language, being a major part of human culture is learned in early periods of life (with plenty of examples learning additional languages at later stages of life), thus organically acquiring rules of verbal communication protocol. Moreover, considering the large amount of natural languages emerged and their limited geographic reach, differing natural languages was limited to various sub-populations. Consequently, particular communication method was very accessible (*availability*) to the societies internally and on the contrary providing a very limited access to parties coming from a different society/location.

Secrecy was achieved by having a private conversation away from other people. Interestingly for secrecy purposes private chat has remained the most trusted communication method and to this day an easily achievable feat in our modern times. Also, some level of secrecy could although be achieved by using a 'foreign' language as means of obfuscating communication. Such method, in computer security referred as 'security through obscurity'. As we will see in later chapters, obfuscation in one form or another was the main method in strive for communication secrecy, until the advent of modern cryptography.

1.1.2 Messengers and Word Of Mouth

With emergence of the language people gained ability to not only convey information to each other, but also for the first time in history, to relay information to parties physically not present, with the help of the messenger person (proxy person). Here we consider messenger to be a part of a communication channel.

In typical human experience, information relayed by messengers are inherently less trustworthy. Hence a receiving party was able to gauge integrity of information based on the sensitivity of the information and any available evidence. Here we can safely assume that verbal communication *accountability* and *secrecy* principles are highly reduced.

1.1.3 Verbal Channel Exploitation

Evidently it is hard to exploit a direct communication, despite the fact, a rogue party can create a deception for a gullible receiver and exploit this illusory rapport.

A more likely exploitation vector is impersonating or intimidating a messenger person, thus trying to exploit situation by false representation. For example a person could introduce himself as a nobility representative and exploit situation for private purposes. To fight such attacks people have been including with a message a limited resource such as relic, personal artefact, likely even a pre-agreed special code words in the language of the message. Interestingly, coupling communications with a *limited resource* continued to be a working strategy to date, especially in indirect communication methods.

1.2 Symbolism and Written Language

Writing – method to transfer information (ideas) onto a physical medium. Co-invented and reinvented multiple times during the last 10 millenniums of human history. Writing have bootstrapped human civilisation by enabling information flow to coming generations.

Although, having a long evolution period and being as useful as it was for historic populations, literacy only became widely common only in the last two centuries. Thus we will briefly overview writing before it became widely spread and thereafter.

1.2.1 Writing Known to Few

Surviving historical writings show it's very limited use and usually associated to public domain by nobleman and alike in such areas as religion, governance, trade and correspondence.

Moreover, continuous development and increase in complexity of writing systems, such as transition to syllabic writing systems, where words are transcribed in symbols representing sounds, or growth in volume of symbolic vocabulary - thus becoming less intuitive and burdened further and wider adoption of writing as a tool, in ancient societies.

From communication point of view, writing has allowed, for the first time, to transfer information over distance, where information could be reconstructed without the physical presence of another human (information source or a messenger). Therefore, for the first time a message could be sent to another party, without disclosing the message to a person carrying the message. Here we can start to perceive how the needs for secrecy in personal communication are being fulfilled. Also, considering abysmal literacy rates at the time in question, writing was a perfect tool for obfuscating communication to a high degree from the majority of the public.

If language have provided means to only pass the message orally, through another human, where the message is retained in messengers limited mind, writing it down presented a plethora of new inventions in communication methods. Particularly, message could be transferred more precisely (increasing information *integrity*) and messenger services could be significantly scaled as message memorisation was not needed, improving *accessibility* and throughput of a particular communication method. In addition people tamed birds or used environment to carry information between parties, for the first time enabling communicating parties to remote communication, without relying on a third person, further improving the level of communication *secrecy*.

With writing becoming an important tool for people in power, opposing groups and opposed groups need for security qualities in communication grew further. Written artefacts show that early forms of obfuscating writing was already developed in ancient Rome and can be traced to ancient Greek times. These first obfuscation methods relied on letter transposition or substitution techniques and is considered to be the beginnings of cryptography art.

Even though few cryptographic methods being developed so early in writing history, cryptography will not become mainstream until the development of internet in part due to high costs associated to employing cryptographic obfus-

cation by-hand. Utilisation of such methods further obfuscates communication adding a layer of *secrecy* to a communication protocol and thus communication itself.

1.2.2 Writing Becomes Common

Age of enlightenment has significantly accelerated adoption of writing in european populations and already at the beginning of 20th century literacy rates were as high as 90% in some european countries. With the rise of literacy rates, communication carriers grew as well and in particular postal and package delivery services. This growth had been further accompanied by technological advances such as transportation system developments. It's worth mentioning that these developments improved deliverability and quality of post transit, improving on message *integrity* in transit, compared to earlier periods.

Sending a letter became a common communication form for over distance communications. Letters could be sent using global postal services. To access this communication channel sender can buy a post stamp, which can be perceived as limited resource needed to *access* the postal services. A letter usually is enclosed in an envelope on which an appropriate amount of post stamps is placed and delivered to post collection point.

Party receiving a message (letter), can evaluate the state of an envelope, find out who is the sender, if any indicated, and start assuming certain *trust* and *secrecy* qualities, such as: if sender is a familiar party; if envelope has any visible signs of tampering; writing style and other. Further, receiver having built some expectations of trust in relation to the message will evaluate the content of the message. These steps are intuitive and natural for receiving party and is supported by physical evidence.

Furthermore, people have adapted this communication channel for a wide spectrum of applications, including for conducting daily business operations in particular by posting business related documents. A good example of a use case, would be sending a contract, associated to a legal status of a sender entity. Such document is considered a viable evidence in the court of justice and can be used as a liability instrument, thus providing *accountability* property for this communication channel. Though this accountability is directly associated to a legal identity (limited resource) and therefore on the legal system enforcement.

1.2.2.1 Implications

As we seen previously, writing offers an improved method of communication compared to sending oral messages and even improved certain qualities of communication security, in particular message *integrity*, slight information *secrecy* improvements and *accountability*, through public law framework, where applicable.

On the other hand, prevalence of written communication, has encouraged societies in embracing this method and becoming common part of human life. For the first time a large part of human communications was being transmitted in public domain and thus became a subject of interest and tampering for various third parties - here we would like to mark a point in time, where we lost *privacy* of our communications.

As we will see, people continue to improve communication methods and their communication secrecy properties, and how technology solved some security problems. Interesting to note, that oral and written communication have transitioned in large part to a new communication channels and has been further enhanced by these modern channels.

1.3 Modern Era Communications

Intro to digital communications (brief review of early methods) Discovery of electromagnetism and related developments has enabled information transfer over physical copper wires, air and later over an optical medium; and again the original messenger and his alternatives was yet again succeeded by a new method, only this time it relied on inanimate electromagnetic waves. As these differing methods matured it provided a wide access to a new kind of communication methods for a big part of developed society and as expected became the main means of remote communications.

If electronic communications improved remote communications, as we will see, technological advances have created an attack vectors on a private conversations. Now technology can be used to collect and transmit private face to face conversations, without the knowledge of communicating parties, due to physical nature of sound, thus degrading secrecy qualities of that communication method. Although, writing could be used to easily defeat this attack method.

Historically we developed language first and then followed the writing. For dig-

ital communications writing proved to be rather more suited information structure for transmission over electromagnetic communication channels, also spoken communication means were introduced almost half a century later. Moreover, besides written and spoken information people managed to generalise these communication channels for general purpose information transfer.

1.3.1 Telegraph

Telegraph was the first means to transmit information over distance, without travelling. It relied on connected electric lines were electromagnetic signal modulation was used as the method to relay written information. Information relay required experienced individuals on both ends of line, for message encoding and decoding into to appropriate, agreed signals, protocol if you will, such as Morse code. If a message sent over the postal services is referred to as a letter, in the case of telegraph the message is called a telegram.

To communicate to another party, message sender has to physically deliver his message to a telegraph service, where telegraph operator would encode the message in transfer; and his counterpart on the other side of the line would decode it on the fly and put it back into written message form. The message, in the last mile of transfer, would be delivered to receiver by post services or a similar method.

It is evident that in order to send a message using telegraph we need to disclose the message to third parties. Moreover, anybody wiretapping the lines were able to fully intercept all communications between the end nodes, thus communication method has drastically reduced secrecy properties. It's worth mentioning that a single line between two operators could only be used in one direction and for a single message at the time - limiting information throughput and complexity. From an overview we can deduce that telegraph system provided poor privacy options for it's users, and due to physical architecture was not as accessible as post services to the general public.

1.3.2 Landline

With invention of telephone people further developed landline network and by the 1970 telephone have reached major part of households in the 1st world countries, with further high penetration levels in the remaining parts of the world. Moreover, telephone systems allowed people for the first time to communicate

directly to each other over considerable distances in real time without the need of active assistance from other individuals.

First telephone networks grew locally and organically and separate networks would be connected by intercity and inter-country relays, first operated by human operators. Having human operators connecting the parties (addressing) and managing the line connection meant every communication had to be initiated by a third party, in addition operator always had access to the same communication channel to assist parties on the line. This implies that users could not assign any secrecy properties to this communication channel. Interestingly expectations for telephone communications have not changed significantly and in current times phone conversations is assumed to be a non-secure private communication method.

Advancements in microelectronics industry allowed for telecoms to replace human operators with machine relays and later with fully electronic systems. Particular shift has created an addressable identifier, in this case phone number, to connect parties willing to communicate on interconnected network. This address can be seen as electronic address of a device on the network, in the same manner as house address on the network of roads. We consider house address as wall as network address, and phone number, to be a kind of limited resource, which is also required to access particular communication channel. Furthermore, network address can be associated for a time period to an individual or legal entity in the same manner as post address is an extension of particular entity, in social contexts.

1.3.3 Computer Networks

Developments in digital communication technologies and advent of a personal computer has pushed to adapting landlines and later other more dedicated type of networks to create an interconnected global computer network - internet. On this network, machines on the behalf of human programmer and occasionally end-user are able to communicate to other machines and any user using them, using a common protocol.

To adapt communication technologies for digital communication purposes, information is sent in small packets, exactly because of this feature internet is also called packed-switched network, wherein 'switching' implies addressing and directing packets on the network. The first widely adopted internet protocol and as to date the most used protocol is called IPv4[P+81]. As a possible replacement in 90's Internet Engineering Task Force (IETF) have defined a next generation protocol called IPv6[DH14].

1.3.3.1 Protocols

The biggest differences between IPv4 and IPv6 internet protocols relevant to our discussion is that on IPv4 networks, not all devices are equally addressable on the global network. In particular, machines connected to internet using IPv4 is often behind an internet gateway and can easily access machines on the public network space, though other machines can not directly address packets to this machine. This discrimination on the network level had both positive and negative consequences. On one hand it has shielded machines from direct attacks, on the other hand, it has created a challenge of creating a direct communication channel between two parties behind internet gateways. Contrary, IPv6 does not discriminate it's nodes on the network, where all peers are publicly addressable and are able to engage in peer-to-peer connections. A similar setup can be found in phone networks, where a local phone network is connected to a larger network, through a border gateways placed in between connecting networks; phone users on both networks can be addressed and reached respectively. Though, IPv6 is still in it's early deployment and barely reaches 15% of market penetration [Inc]

While discussing communication technologies based on computer networks it is convenient to use Open Systems Interconnection model (OSI model), see figure A.1, where underlying communication channel is divided into 7 protocol layers, corresponding to particular defined functions of the layer. There is various methods to provide communication security properties at every OSI layer, though application developers usually have access only to Transport layer and layers above. Introduced IP protocols fall unto layer 3, in this model and is generally managed by operating system. Most of the communications relevant for our discussion is carried out through Transport Control Protocol (TCP)[Pos03] and User Datagram Protocol (UDP)[Pos80].

Moreover, IPv4 and IPv6 both provide Internet Protocol Security (IPsec)[KS06] suite to provide security mechanisms through cryptographic means on the network layer, though only IPv6 requires a support of these features. Specifically, IPSec describes protocols for authenticating peers, encrypting payloads for TCP and UDP packets and further provides a higher level of data integrity.

However, IPsec is rarely provided by default on the networks, as it requires a prior setup of security elements between nodes wishing to employ suite in question. As these management tasks is the responsibility of the computer administrator, furthermore, protocol is transparent and inaccessible to an application developer - we can not rely on IP layer for our security purposes, thus in our conversation we will focus on layers above.

Even though there is a wide range of transport layer protocols available, more

than >60% [Arc] of internet traffic employs predominantly TCP and to a lesser extent UDP, as an underlying transport protocols. Developers construct session or application layer protocols above mentioned protocols as per OSI model definition. First widely adopted application protocol standards were not security oriented and usually transmitted data in plain-text; to enumerate a few: Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), TELNET. Later, due to grown security requirements a Transport Layer Security (TLS) protocol was developed to offer intermediary security layer first for HTTP [Res00] in particular and subsequently to other legacy protocols such as FTP, various e-mail retrieval and transfer protocols. Moreover, a multitude of new general purpose and specialised protocols was developed were some of them have built-in security features. For example, a well known protocol amongst computer administrators and developers is Secure Shell (SSH) protocol, providing communication security for operating system level tools.

A better known instant messaging (IM) protocols include: Internet Relay Chat (IRC, security can be provided over TLS layer); Extensible Messaging and Presence Protocol (XMPP)[SA11], which has TLS security built-in; Session Initiation Protocol (SIP)[SCH+02], with optional TLS security, in large provides Voice over IP communications; Off-the-Record (OTR)[BGB04] protocol defines secure message communication protocol, in design similar to OpenPGP e-mail suite. Though, some of these protocols are suited for a rather more complex services than a peer to peer, or provide security properties only during data transmission between service points and is not suitable for our purposes. Out of mentioned protocol selection, only OTR could fit our requirements, if communication is transferred with the help another party. Otherwise, remaining protocols could suit our requirements in peer to peer communications, if applicable.

E-mail communications usually are relied using Post Office Protocol (POP/POP3), Internet Message Access Protocol (IMAP) and Simple Mail Transfer Protocol (SMTP). All these non-secure protocols can be upgraded with TLS layer of security, though as common with (IM) protocols, TLS provides security in these protocols only between machines relying the messages, but the communication is stored and processed in plain-text on all end and mid nodes. To provide communication security between communicating parties using email channel, users have to actively encrypt and decrypt their messages using OpenPGP, S-MIME or a similar email security system.

In addition, there is a wide range of peer to peer communication protocols, unfortunately, majority of them are specialising in file-sharing services. Specifically, for performance reasons, because of large amounts of data these protocols need to handle, developers did not include strong security features and rather use payload obfuscation methods. Also interestingly, BitTorrent file sharing project have created of BitMessage protocol for anonymous trust-less secure

instant messaging.

Security properties in computer network communications and specifically in mentioned security protocols are provided by cryptographic algorithms. Utilisation of cryptography, also requires more computational time, therefore causing an additional overhead to the communication channel, in many cases this has been one of the main factors stopping a wider adoption of cryptography in everyday computer communication networks.

1.3.3.2 Cryptography role in network communications

Modern cryptography is usually split into symmetric cryptography and asymmetric cryptography, latter also often referred to as public-key cryptography. As we will see these two branches of cryptography fulfil very different niches in communication security requirements.

Symmetric cryptography in practice is used as the main means of communication obfuscation method for secrecy purposes. Keyword symmetric implies that same cryptographic key is used for encrypting and decrypting information, therefore parties obfuscating their communication must share the same key for successful communication. Another useful application is constructing of a one way function to transform variable length information into a fixed length - in cryptography called hash function.

Predictably, asymmetric cryptography uses a pair of keys - key-pair, where one key can be used to obfuscate information and obfuscation can only be reversed with the other key respectively to key-pair. This unique mathematical mechanisms have allowed for a wide range of useful applications in network communications and computing in general.

Relevant security mechanisms provided by asymmetric cryptography:

- **Authentication, Identification.** If one of the keys (public-key) can be associated with an identity (such public-key is also called a certificate) and placed in trusted repository (public-key infrastructure) and the party claiming to possess the other key (private key) can cryptographically prove that he is the owner then we can assume the identity of that communicating party with certainty.
- **Digital Signatures.** If we include a 'digest' of a message with a cryptographic hash function encrypted using our private-key bundled with a message, a receiving party can repeat 'digestion' process on the message

and compare to 'digestion' decrypted using the public-key of a sender. If they match receiver can be certain that: contents have not been changed in transit (integrity); asserting message sender identity.

- **Encryption and Symmetric key exchange.** Public-key cryptography can also be used to encrypt small amounts of information and is often used in securely exchanging symmetric key information, through Diffie-Hellman or other key exchange mechanism. In strongly authenticated communications receiver can also be sure that he was an intended destination for that particular communication as well as anything that is sent to this party encrypted using his public-key.

As we stressed beforehand, predominantly symmetric cryptography is used for obfuscating information in transit for privacy and secrecy purposes. Though as per usual asymmetric cryptography can be used and often is used to identify and authenticate parties and securely exchange symmetric cryptography security elements among other use cases. A common occurrence in today's world is to derive trust for cryptography based communications based on a public-key infrastructure.

As we see cryptography is capable of fulfilling our originally stated security needs. Though an effective use of public-key cryptography requires a working public-key infrastructure or manual public-key distribution throughout devices on the network, the latter being not a viable solution for global deployments. On the other hand, as we will see in chapter 2, current globally available public-key infrastructures fall short in providing a more general and universal access to these services. Therefore, limiting effective application of both symmetric and especially asymmetric cryptography.

Chapter 2

Overview of Public-Key Infrastructure Systems

2.1 Public-Key Infrastructure

Public-Key Infrastructure (PKI) system provides security services for managing public-keys, often in centralised manner. It's role is to provide public-key signing, storing, revoking and other key management procedures. There are a few different types of public-key infrastructures available for a public access. The main differences between them is the model of how trust qualities are derived in particular infrastructure, and by which parties the system is supported and managed, as well as the type of access provided to them.

A common feature among all PKIs is that their primary function is distribution and management of digital certificates. There is a three widely accepted Digital Certificate types - X.509 [CSF+08], OpenPGP public-key [CDF+07] and OpenSSH certificates[LY06]. Even though these formats are not completely comparable with each other, it is possible to convert one certificate to another format, though such cases occur, but are infrequent. Moreover, there is a plethora of certificates storage formats available.

Certificates, for storage are encoded using Distinguished Encoding Rules (DER)

format, described in [CSF+08]. As encoding in DER produces binary blob it is difficult to transmit as text, therefore, it can be further be encoded using Base64 encoding into a text format. This procedure first introduced and further described for Privacy Enhanced e-Mail (PEM), defined in [KL93] and [Kal93]. Certificates encoded using this scheme are referred to as certificates in PEM format.

Furthermore, in the overview of current PKI systems we will see that responsibilities of private-key and certificate management fall on a dedicated administrators or end-users, where in the first case end-user is often oblivious to existence of any PKI, and in the latter, user is overwhelmed by certificate management tasks and the complexity of the system. Later we argue on shifting key-management responsibilities to an underlying application developers, who are likely to have more experience in security contexts of digital communications.

2.2 Synchronizing Key Servers and OpenPGP

OpenPGP is an open standard defined in [CDF+07], describing a cryptographic suite and procedures predominantly used for email signing and encryption/decryption. Though, it can also be used for authentication, identity and information signing purposes as well as for data encryption in data-at-rest manner. Cryptographic services in this system are provided at a layer above an application layer in OSI model. OpenPGP can employ Synchronizing Key Servers as an underlying PKI for centralised key storage and retrieval.

2.2.1 PGP Private-Key and Public-Key

OpenPGP defines it's own format for packaging both private-key and public-key, latter is also frequently referred to as certificate. In particular for our discussion we are interested in parts providing identity (UserID) associations to public-key as well as unique key references such as Key ID and Digital Key Fingerprint. Mentioned and unmentioned certificate fields are signed with corresponding private-key, therefore certificate and all fields can be self-verified.

Private-key can be protected with a password, in such case user has to unlock key first so it could be used for information decryption and signing purposes. Unlocking of a private-key is always done by a PGP key managing application; unlocked keys are often held in program's application memory for the remaining duration of the session.

Next we will overview a typical communication practice using OpenPGP system.

2.2.2 Typical OpenPGP System Use

To begin using OpenPGP (PGP) a user first has to create a key-pair, this has to be accomplished with user's input, either manually in terminal, or in graphical environment using OpenPGP compatible software. When creating a key-pair user is required to provide his name and an email address that will be associated to particular name identity and optionally a password for encrypting private key. The public-key part corresponding to the private-key is a PGP type certificate, with specified identity labels and signed using private key. It is important to note that files generated will be likely stored on the same machine that is used for generating key-pair and either will be managed by supporting software or the user himself - to his best abilities.

Once generated, certificate part can be shared with other parties that user might want to communicate in secrecy. Certificates can be exchanged as files locally or remotely, though most practical approach is to upload it to a public or private SKS Keyserver. SKS Keyserver is a public-key infrastructure based on web-of-trust concept and is the key component relevant to our discussion in this section and we will overview it in short.

Particularly in email communication PGP can be used for only signing the email message, or encrypting and signing the whole content of an email. In the case, where email is being signed only, sender signs 'digest' of an email body with his own private-key, that is to provide a proof of his identity and to ensure integrity of a message in transit. Here, receiver(s) can use sender's public-key to decrypt signature and verify the integrity of the message, if successful.

A user (sender) can start securely communicating with another party (receiver), only when he has procured certificate - allegedly associated to the receiver, either, through mentioned file exchange, or from a relevant SKS Keyserver. Receivers public-key is used to encrypt symmetric key used for message encryption as well as signature from the sender included with encrypted message. If receiver can successfully decrypt the symmetric key and thus the message itself - he can be sure that he was the intended receiver. Furthermore, if signed 'digest' included by the sender can be decrypted using public-key associated to his identity, receiver can assume a level of trust in integrity of the message, identity of the sender and secrecy of the communication - given that receiver's private key has not been compromised.

2.2.3 Synchronizing Key Servers (SKS keyserver)

SKS keyserver is an open source project providing public-key management services rooted in web-of-trust model, through HTTP Keyserver Protocol (HKP) as defined in [Sha03]. With Synchronizing in SKS' name, developers tried to imply that SKS keyserver can synchronise with a pool of other keyservers, thus can provide distributed key storage and retrieval infrastructure. SKS is currently used to support and host a decentralised global pool of keyserver available for public use, although keyservers and their pools can be configured for either public or private use.

Keyserver use cases, as defined per HKP, can be divided into personal public-key and second-party public-key procedures. Regarding users personal public-key, he can use keyserver to publish the key to be used by other users. In addition, keyserver supports public-key revocation, though a corresponding private-key is required to generate a revocation request.

Furthermore, keyserver provides search functions for finding published public-keys. In searching for a public-key user can use look-up for keywords in certificates User ID field, search for specific Key ID or Key Fingerprint. Furthermore, any user can sign another public-key and upload it to a relevant keyserver. Signing of another certificate can be done on User ID or Key ID, where in the first case user participates in web-of-trust model and endorses another user's identity, according to OpenPGP requirements and in the latter case user creates certificate chain extending and confirming identity associations to that new sub-key.

Web-of-trust is a concept to build a complex hierarchies of trust between peers. Trust in this model is derived from specific public-key endorsements, as mentioned, where user signs other users identity and public-key binding, rather than relying on trusted-third-party. Trust built in this way closely mimics social human trust network. Furthermore, peers in such PKI system are completely equal, thus establishing a flat trust network.

2.2.4 Pros and Cons of PGP

OpenPGP provides a strong infrastructure for secure email communications, if used according to standard, especially following recommendations for securely managing private-keys and participating in web-of-trust creation by endorsing each other certificates. Also, for an expert user it provides a flexible model for managing public keys related to parties of interest, where keys can be im-

ported from files, exchanged through another secure communication channel or downloaded from SKS Keyserver.

PGP is also often used, notably in an open source communities, to sign public messages in open forums, signing electronic documents as well as any other type of information such as software packages, code contributions to open source repository, etc., where identity associations are required.

On the other hand, as study [SBKH06] indicates OpenPGP has a high learning curve for novice users. Findings show that the main obstacles for users is public-key certification in particular - publishing public-key and procuring public-key of a receiving party. Furthermore, it is highly non-intuitive for any person, who has not been introduced to public cryptography, as to what particularly signing a message means and if the message will be really decryptable by the recipient as well as roles of public and private keys.

Moreover, user is responsible for managing his key-pair securely e.g.: have a strong password protection for a private key; if needed, responsibly manage key-pair between multiple devices; revoke old keys. Most people find themselves overwhelmed with these responsibilities and justifiably so in comparison to simplicity of username and password protected systems and evident poor end-user performance.

Also, PGP does not provide Perfect Forward Security as communicating parties keep using the same public-keys to encrypt the symmetric keys used in message encryption and, if one of parties private-keys are compromised all collected or saved communication messages directed to that recipient can be decrypted. For our point-to-point communication purposes PGP provides too of a narrow range of communication methods, particularly in it's current form it's only suitable for non-real-time communications, such as email communication or an internet bulletin board systems. And as we saw it's a difficult system to use.

2.3 X.509 and Certificate Authorities

As we saw in a section on OpenPGP infrastructure, trust is derived completely from certificate and it's properties and any user knowledge held about the certificate. A different approach is to derive trust from a trusted third-party vouching on an identity of another party of interest. Such third-party providing trust services is also called Certificate Authority (CA). Certificate Authorities responsibility is to provide roots of trust and trust chains and manage certificate validations - implicitly including and excluding certificates to the chain. CAs

provide these services according to X.509 family of standards. In order to derive any trust in a certificate, which is a part of certain trust chain, particular chain's root certificate must be trusted explicitly beforehand.

X.509 standard [CDH+05] defines a format specifications for digital certificates often used with TLS protocol, especially with HTTP protocol - HTTP over TLS (HTTPS)[Res00]; as well as best practises for validation of identities tied to certificate. Contrary to OpenPGP's flat infrastructure, X.509 defines certificates that could be used for many different purposes depending on the properties assigned to certificate and can be generalised into categories as follows: CA certificates, self-signed certificates to provide root for chain of trust and certificates for validating other certificates; Server certificate, usually validated by CAs is to provide authentication and identity for service providers; Client certificate, usually validated or issued by service providers, besides client authentication and identification can be used for digital signing. Also, server and/or client certificates are often used to securely exchange session cipher keys, once parties have authenticated or co-authenticated.

2.3.1 Issuing Certificates

Public and private keys conforming to X.509 specification can be generated using any supporting software such as OpenSSL or internally in applications using native or with library provided Application Programming Interfaces (API). Any party creating a key-pair have to go through same steps with few key differences, when providing identity fields and public-key properties. Once key-pair is generated following actions would depend on purpose of the key, as we will see.

A user generated key-pair, can be self-signed, though such certificate would not be able to provide any identity associations as it has not passed any identity validations and only proves that entity has a corresponding private-key used for signing. Such certificates are often used in development environment by developers and in local intranet deployments managed by system administrators. Additionally, such certificates are often distributed among trusted devices in local deployments, where trust is implied from a set of local trusted certificates. Such method can be perceived as a form of certificate pinning, meaning that by selecting specific certificates or certificates signed with specific certificate is to be trusted explicitly. Pinning with HTTPS protocol specified in [EPS15].

Described infrastructure can be managed manually by a systems administrator or automated with the help of a private PKI system, for example using Lightweight Directory Access Protocol (LDAP) services or Microsoft's Active Directory Certificate Services (AD CS) services. Further, these certificates can

be used for establishing a secure communication channel with services such as Kerberos system and Internet Key Exchange (IKE) protocols, among other, for negotiating security elements needed to fulfil communication security requirements. Equivalent schemes has proven themselves to be very successful internally in enterprise deployments. Also, mentioned systems can work with certificates validated by CAs.

2.3.1.1 Certificate Authority Chain

Certificate Authority has to manage two types of certificates, specifically Root Certificate and Intermediary Certificate, see figure A.2. Root certificate is a self-signed key with enabled `CAcert:true` field. The private-key part is explicitly only used to sign Intermediary certificates, otherwise protected from being disclosed by highest means achievable. Certificate or public-key of Root certificate is used to validate Intermediary certificates and is distributed freely. Private-key of an Intermediary certificate is used for validation (signing) of end-user certificates (server/client certificates) associated to certain identity claims and corroborated by CA. Intermediary certificate itself is usually distributed and presented together with end-user certificates and is used to validate the particular certificate as well as to validate Intermediary certificate with a relevant Root Certificate.

2.3.1.2 Certificate Validation

In order to obtain a valid end-user certificate it must be validated by a trusted CA. For example if a user generates a key-pair that he intends to use as server certificate, then he needs to generate a Certificate Signing Request (CSR), which includes a public-key, and is signed with a private key. CSR is forwarded to a validating CA services, who according to X.509 standard requirements and internal company procedures validates the identity of a user and issues a certificate. A most common case is to validate that user controls an internet domain name, which is used as an identity field (CN) as per best practises defined in [SAH11].

Procedures for issuing client certificate is equivalent, except when it comes to identity validation. In such cases as there is no domain name to validate, certificate issuing authority usually has an internal identity validation process.

2.3.1.3 Certificate Trust

Major difference, in comparison to OpenPGP key infrastructure, is that trust can be extended to many validated certificates, and only CAs Root Certificate is needed beforehand, to validate trust chain leading to certificate in question. Customarily, current operating systems have a repository of trusted Root Certificates, which provides a system wide trust architecture, to be used by any applications employing X.509 PKI systems. Occasionally, applications might internally include supplementary CAs Root certificates and other type of certificates for application wide use. It's usually the case that a set of trusted Root certificates CAs is managed by operating system developers and in enterprise environment system administrators.

2.3.2 Typical use of a CA managed PKI

There is minor differences in using certificates validated by CAs arising from differences in an environment used, in essence the application used and for what purposes certificates are employed. Specifically, in web browsing user is actively involved in starting a communication using browser's address bar, however applications employing certificates for TLS protocol to secure communications, establishes connection in background without overview of the user.

2.3.2.1 Web Browser Environment

Browser can utilise both server and client type of certificates. Server certificates are essentially used for authenticating and identifying service providers, using Domain Name System (DNS). Essential web browser component is address bar, where user types a desired service provider domain name, to access the remote service.

Upon access, if using HTTPS protocol, a server certificate is presented to users web browser and is validated against the local trusted root certificates. Further certificate's Common Name field is matched to a domain name of the service. Specifically, to improve trust in service provider entity, independent CA/Browser forum laid guidelines for an Extended Validation certificates (EV certificates). To issue EV certificates, accredited CA, must extensively validate the identity of the certificate requesting party as a valid legal entity. Also, EV certificates can be issued for multiple domain names.

Web browsers provide feedback on the state of server certificate validation by

changing address bar background colour to green or red appropriately as well as clearly indicating by text or visual icon at the beginning of the address bar. Furthermore, in the case of EV certificates, mentioned guidelines by CA/Browser forum instruct browsers to indicate the identity of the party claiming the EV certificate as well. Likely, that using HTTPS connections through a web browser, is the only common communication method designed to provide a feedback on validation of certificates, identity and implied level of trust of an underlying communication channel and remote party.

To use X.509 client certificate, certificate and corresponding private-key must first be installed in browser's certificate store. Keys can either be installed manually by end-user and system administrator, or pushed to browser by a remote service, with user's consent. Once client is accessing a service over HTTPS, service provider upon presenting his server certificate may request client certificate from a specific trusted CAs and if available browser will provide appropriate client certificate and authenticate against private-key. Client certificates in browser environment are used rarely, and is usually employed by governments for employees and citizens to access e-government service and in enterprise environment internally.

2.3.2.2 Use within applications

When using TLS layer in HTTPS or any other protocol in applications, communication channels are opened and established in the background of the application. In such situation, there is no set procedures to inform a user about underlying communication security properties. Therefore, setting up a communication channel and handling exceptional cases is application developer's responsibility as well as preparing trusted root certificates sources.

Also, x.509 certificates are often used for Access Control. For example, controlling access to wireless network, where client certificate is used to authenticate user or a device.

Although, there is a wealth of specifications and proposals that employ and extend X.509 cryptographic certificate uses, covering them is not our objective. It suffices to note, that public-key cryptography is a highly desirable feature in communications and the distribution of certificates presents a tremendous challenge to this day.

2.3.3 Pros and Cons

X.509 is the most proliferated certificate type in use with current communication technologies. Arguably, it success was in large part due to success of trust model rooted in independent distributed Certificates Authorities, which offers a flexible trust system that can be customised per user environment or even per application.

PKI based on X.509, does not rely on any central service for certificate distribution. Certificates and their trust chains a rather distributed by service providers as well as clients during the communication establishment. Lack of central services removes possibility of a Denial of Service attack on PKI services.

On the other hand, almost all of CAs charge fees for certificate validation services. This particularly limits application developers from easily obtaining a free validated and widely trusted certificates, even more so, if a client application used is not curated by the said developer. Notable exceptions to paid CA services is CAcert, internet community run CA providing certificates largely used in open source projects, as well as Let's Encrypt CA. Let's Encrypt is a recent CA providing a free of charge X.509, exceptionally only server certificate, validation services.

Furthermore, as system trusted root certificate store is used to derive trust, it provides a single point of failure. Having so many independent trusted CAs with different management processes creates multiple attack vectors for a a rogue party. If a single widely trusted CA was compromised, it would undermine trust in the whole PKI as an attacker could impersonate any valid identity. In such situation, specific trust chain could be added to a revoked certificates list (CRL), though revocation lists are rarely implemented and used during certificate validation, with rare exceptions.

Considering, that end-user can fully manage trusted certificate store, by adding and removing Root, Server and Client certificates and that common computer user is poorly acquainted with different certificate types, certificate files and trust models, end-user becomes a weak link. Specifically, end-user can install a certificate with few mouse clicks, thus himself undermining the whole trust mechanism and becoming susceptible to Man-In-The-Middle (MITM) attacks. To put simply, end-users understanding in these matters is at odds with responsibilities provided and expected from the user.

Chapter 3

Motivation

As has been stated at the beginning of this document our primary goal is to enable secure point to point communications between different parties. Considering, that majority of our communications take place over digital communication networks, which are often established and managed by many actors, we advocate providing security features at least at the Session layer or layers above in relation to OSI model. Not to mention, we expect the reuse and adaption of currently defined and implemented communication protocols, as well as those that are proposed by IETF.

Cryptography, is the main means for providing security features as required in Chapter 1. In particular public-key cryptography is at the heart of such communication channels, nevertheless options for distributing and managing public-keys are still very limited. Arguably, OpenPGP community have successful created a synchronising decentralised PGP certificate server system, which has been very successful in it's function and can be easily replicated for a private use. We propose an analogous system for public-key storage and retrieval, though with support of multiple certificate formats, introduced in section 2.1.

All overviewed PKI systems, require some level of user input or interaction regarding certificate management. As we argued, users rarely have any familiarity to these systems and are unable to adequately take responsibility of those tasks. In our solution, we shift these responsibilities from an end-user to an un-

derlying software developer, by providing them with an appropriate APIs and infrastructure.

Furthermore, besides certificate handling, a user, whether it's service provider or a client, is expected to be able to protect any relevant private keys and safely utilise them. Again, applying the same reasoning as in prior point, we argue that private-keys should be handled by software developers. Fortunately, many operating systems provide KeyStores (KS), that provides APIs for key-pair generation, storing and utilisation. Often, such KS is backed by security focused hardware.

Trust systems, in relation to PKIs, that we have introduced so far, namely Web-of-Trust and Trusted Third Party, provide systems that is: either too vague with trust properties as in OpenPGP system, where a user has to evaluate and derive trust in the certificate; or too rigid and overarching as is the case with current X.509 certificate environment, where root certificates provide system wide trust. Although, the PKI system that we propose could incorporate widely accepted CAs chains, but also can easily accommodate certificates trusted in any application locally, without enforcing system wide trust. Thus, opening possibilities in providing services for an enterprise and private entities alike. In fact, we do not propose any new security protocols as per se, but propose some infrastructure innovations, such as API services, as well as transfer some end-user's responsibilities to software developers. Furthermore, we discuss what new doors these innovations could open for us in dedicated security services.

Chapter 4

Public-Key Infrastructure based on SMS channel

To be able to consider design details of our PKI and its security properties we need to familiarise with relevant mobile networks parts as well as SMS channel.

4.1 Introduction to Mobile networks and SMS services

4.1.0.1 Mobile networks

Mobile network is a cellular network, where enrolled parties can access their telecommunications provider network using any of multiple radio cell towers, also called Base Station Subsystem (BSS), in a physical proximity. Traditionally, telecommunication networks have been organised into a smaller networks per geographic location usually per country basis, as we reflected in 1.3.2 section, analogously mobile networks are organised in a same fashion. User or a device enrolled to mobile network is referred to as network subscriber.

Current standards and specifications for communicating between mobile device

and a BSS as well as internally in mobile networks are curated by 3rd Generation Partnership Project (3GPP). Project is a participation of many organisations and entities interested in standardisation and evolution of mobile communications. European Telecommunication Standards Institute (ETSI) is a partner and a co-founding party that provides Information and Communications Technologies standards and specifications at international level.

Present day mobile networks support provide access to network subscribers through Global System for Mobile Communications (GSM), Universal Mobile Telecommunications System (UMTS) and Long Term Evolution (LTE)[Zyr08] standards, protocols also commonly referred to as 2G, 3G and 4G protocols, where a number denotes an order of succeeding protocol and G stands for generation. Proceeding protocols always provided backwards compatibility on the protocol level as well as introduced extensions and security improvements.

Although, some operators started to phase out the least secure 2G networks, networks supporting all different generations of protocols as still widely deployed, because networks grow organically based on demand and available resources.

On the mobile network operators side GSM and UMTS protocols are supported by Signalling System No.7 protocol (SS7). ETSI provides a specification of SS7 in [ETS95]. For mobile network data transfers SS7 protocol can be used over a dedicated Public Switched Telephone Network (PSTN) or over the internet. For supporting SS7 network over internet infrastructure, IETF have developed a suite of specifications in common referred to as SIGTRAN.

On the other hand, LTE (4G) networks rely on Signal Initiation Protocol (SIP) with Diameter cryptography protocol for Authentication, Authorization and Access Control. LTE infrastructure provides a higher communication security levels in comparison to SS7 network. Though, LTE is still at its early adoption stage and currently heavily relies on SS7 network as underlying technology on mobile operators side.

To access a mobile network a valid Subscriber Identification Module (SIM card) is required. SIM is a smart card that is used to securely store private and public information associated to a module and subscriber. Relevant information stored:

- Mobile Subscriber Identity (IMSI), a secret number that uniquely identifies a subscriber on the network,
- Temporary Mobile Subscriber Identity (TMSI), a secret temporary number to be used instead of IMSI, as IMSI is transmitted as rarely as possible.
- TMSI can be used on the SS7 network to find a corresponding IMSI of a

particular subscriber.

All the devices on the mobile network are addressed using its MSISDN. Essentially, MSISDN is a phone number. MSISDN together with IMSI provide a mapping to a particular subscriber on the network.

Among many specified GSM services and protocols, the one of interest for us is a Short Message Service.

4.1.0.2 Short Message Service

Short Message Service is a very widely adopted text messaging protocol defined in GSM standard and its successors, as well as in SS7 specification. ETSI defines implementation details in [ETS].

Essentially SMS is a small data package, containing the receiver's address, an address of relay service, SMS specific flags and a message body. By definition SMS package size is limited to 160 bytes. Therefore, depending on text encoding used can carry 70 to 160 text characters in its message body. To overcome this limitation, a long form SMS was introduced, in a specification referred as concatenated SMS. Concatenated SMS uses part of a message body to identify concatenated SMS parts and their order, thus slightly shortening the message body. Theoretically, maximum length of concatenated SMS can consist of up to 255 separate messages. In our solution we use concatenated SMS.

SMS messages are relayed by Short Message Service Center (SMSC) over SS7 network. Mobile network access providers usually have a local SMSC service that supports its customers. From a protocol point of view there is two kinds of SMS messages:

- Mobile Originated (MO SMS), a message that originates on a mobile device and can be directed to another mobile device or a digital service hosted on SMS network.

- Mobile Terminated (MT SMS), a messages that terminates on a mobile device, and could have originated from a mobile device, or a digital service connected to SMS network.

4.1.0.3 Mobile Network and SMS Security

It has been known for some time that SS7 network has many vulnerabilities. As overviewed in [Wel17], SS7 vulnerabilities can be grouped in such categories:

- Obtaining subscribers information, such as IMSI or TMSI.
- Determining Subscriber's Location.
- Eavesdropping on subscribers traffic.

Relevant attack to our solution is 'Man-In-The-Middle' attack, between mobile device and BSS. Here attackers strategy is to introduce a rogue BSS station that would force nearby devices to connect to it. Rogue BSS acts as a proxy between real BSS and the user's device. Here an attacker can attempt to capture TMSI or preferably IMSI.

If IMSI is found out and and an attacker has a direct access (such as mobile operators) to SS7 network an attacker can divert SMS messages to a false SMSC. Thus attacker is able to divert and capture MT SMS messages. But as we will see later known SS7 vulnerabilities does not raise considerable threats in our solution. Though, found considerably vulnerabilities weakens Multi-Factor Authentication, where One Time Passwords are distributed using SMS channel.

4.2 SMSPKI

Here we propose a PKI system very similar in design to OpenPGP SKS PKI, in particular, providing a certificate hosting online in a key-value structure, where a key is an alphanumeric identifier of the certificate and the value is the certificate file. Furthermore, we define how CA services can be incorporated into proposed infrastructure.

This PKI is intended to be employed primarily by software developers, who wish to employ public-key cryptography and require procedures to distribute these public-keys. Therefore, we consider this system from a PKI service provider's and a software engineer's point of views.

The key parts of this PKI system are:

- SMSPKI - a remote certificate hosting service hosted on SMS channel.

- SMS Certificate Manager - a local operating system service providing API for certificate registration with SMSPKI.
- SMSPKI clients - applications that employ public-key cryptography and utilises a Certificate Manager's API.

As we noted in 1.1 section, that limited resource can sometimes be coupled with a communication channel to improve its security properties. Here we argue for using communications over a mobile network, specifically using Short Message Service (SMS), as we perceive access to mobile network as a kind of limited resource. A user must have a valid subscription to be able to communicate over mobile networks. The subscription is a distinctive limited resource and often closely coupled with an individual.

Furthermore, using additional communication channel serves as out-of-band communication channel, a technique often used to lower risks of using single non-trusted communication channel. For example, many internet services use SMS to distribute One-Time-Passwords (OTP) for Multi-Factor Authentication (MFA) in an out-of-band manner, but as discussed in 4.1.0.3 section, it's susceptible to attacks.

Also, usage of SMS channel limits availability and access to a proposed PKI, however according to a statistics provider StatCounter, at the beginning of 2017, mobile devices accounted for more than 51% [Sta] of all devices used on internet and over 70% of them are running Android operating system. Therefore, we focus mainly on considerations pertaining Android platform and short comparisons to Apple's iPhone platform, where relevant.

4.2.0.1 General description of SMSPKI

A high level design of proposed PKI is visualised in A.3 figure. Here proposed PKI provides services over SMS channel using mobile networks and HTTPS channel over internet. SMS interface of PKI is used to register/publish user certificates and to query certificate databases, with later function also available over HTTPS. Certificates are stored in PKI as key-value entries and are further categorised depending on the type of certificate. PKI is used by mobile device clients and 3rd party service providers.

SMSPKI is to be used with a smart mobile devices capable of SMS communications. As pictured in A.4, certificates are generated on the mobile device and forwarded by MO SMS to PKI provider (1) hosted on mobile network with specific address (MSISDN). Upon receiving the certificate SMSPKI validates the

certificate, signs it, if valid, and replies to a mobile device with MT SMS (2), containing signed certificate, where mobile device stores validated certificate on the device for later use.

Besides registering certificates, a specific certificate can be requested and downloaded from SMSPKI by SMS and/or HTTPS interfaces. Due to SMS channel limitations we specify that certificates over SMS channel can be requested using only unique keys identifying a particular certificate, while HTTPS can be used for wider, but limited searches over the certificate databases.

Further, we define a system wide service - SMS Certificate Manager (SMS CM), for Android based mobile device, that provides API service to a mobile applications (client apps) on the device and communicates to SMSPKI over SMS channel, as pictured in A.5. API service is to be utilised in mobile application (app), where app developer seeking to utilise public-key cryptography, generates a key-pair and registers certificate part with specific a SMSPKI provider. Communication to SMSPKI from an app point of view is abstracted through a SMS CM service API and only that service has a direct access to SMS communications.

From a high level, functional, point of view, PKI provides the same functionality for both OpenPGP and X.509 certificates, though there are considerable differences stemming from the purpose of certificates and trust models they employ, and they will be discussed in more detail in later sections.

To iterate, services provided by PKI, is for storing certificates that can be used for identification and authentication of a user or a device as well as for exchanging symmetric encryption keys for securing communication channel for secrecy purposes.

To summarise, described PKI's main concern is to provide a certificate registration, validation and publishing services. A goal of this service is to facilitate a wider adoption of public-key cryptography to enhance security attributes of underlying communications in software applications. As we argued in previous sections, end-users are not sufficiently familiar with PKI services and infrastructure and currently is the main barrier for adopting public-key cryptography. Therefore we push tasks related to certificate management to software developers by providing a PKI to be used primarily by software applications, where certificate management can be automated.

We expect that PKI services described will be likely provided by a third party, offering certification services, over SMS and HTTPS channels. Any party that has access to mobile network and SMS services could in principle host infrastructure system in question.

4.3 SMSPKI Server-side

As described earlier, we strive to design a PKI supporting OpenPGP and X.509 types of certificates, as they are most utilised with current technologies with considerable difference between both.

For managing both types of certificates, we propose to adapt SKS based solution and provide services over SMS channel, using equivalent protocol to HKP. Also, to provide equivalent, but limited access through HTTPS protocol.

Regarding the X.509 certificates, we propose services that will be applicable to only client certificates. Server certificates, on the other hand are already managed by service providers and certificate authorities. Even though, the certification services differ between these two types of certificates, interface to the server - certificate registration and querying procedures are the same in both cases.

4.3.1 SMSPKI Server-side Design

As already stated, SMSPKI hosts certificates in key-value pair structures. Certificates can be indexed on multiple literal certificate fields such as identity labels and unique certificate identifiers to be used as key.

4.3.1.1 Certificate Fields and Identifiers for SMSPKI

SKS keyservers uses User-ID as the main subject identifier, we opt-in for the same solution for OpenPGP and in X.509 we use CommonName as equivalent field to User-ID in OpenPGP's case. Furthermore, we propose to organize X.509 certificates in common pools. Organizing X.509 certificates into common pools would allow for a more flexible PKI infrastructure, where certificates databases could be split per application, per domain or other basis. Further in document, we refer to these separate certificate databases as sub-PKIs of the SMSPKI.

To maintain certain level of anonymity, we propose to use Universally Unique Identifiers (UUID) in the UserID or CommonName fields of the certificates. We will refer to this field commonly for both type of certificates as Unique Certificate Key (UCK). Uniqueness is only guaranteed with-in a single sub-PKI. Furthermore, using UUID for UCK, provides a consistent certificate referencing, and is very suitable for indexing purposes in the databases.

4.3.1.2 High-level Server Design

A high level design of the server side is shown in A.6 diagram. Here,we have multiple SMS API interfaces as well as HTTP interface per every sub-PKI provided. All interfaces of SMSPKI are always in listening mode and are used for specific service requests.

4.3.1.3 SMS Interface

Multiple API interfaces connecting to SMS router, in diagram A.6, show that our SMSPKI, can be accessed using different phone numbers on the mobile network. Hosting PKI on numerous addresses can serve multiple purposes, some of those are:

- Load balancing SMS channel, to spread SMS communications through different points on the network.
- Providing services in multiple countries, where a 'home' network and a local phone numbers could be used, to circumvent international charging rates for SMS messages.
- Separating services per certain certificate domains, to isolate specific PKI databases per different phone number.

SMSPKI's main functionality is provided over SMS interface. Server interface is always in listening mode, thus it operates in request-response mode. Following operations are available through SMS Interface:

- registration of personal certificates,
- revocation of personal keys,
- retrieval of single certificates,
- registration of signed third-party certificates - to provide a possibility of building Web-of-Trust, as in OpenPGP infrastructure.

All operations regarding certificate operations on SMS interface are performed on single certificates.

4.3.1.4 SMS Router

To provide SMS channel flexibility we define an SMS message routing component indicated in figure A.6. A more detailed design of the SMS routing component is visualised in picture A.7. It is important to remember, long text messages when transferred over SMS channel, in our case certificates as SMS payload, is broken up into multiple SMS messages and then reassembled into single message on an end point; In our design we receive messages in full first, before starting the message routing.

Messages are received and collated by SMS Receiver noted in figure A.7. SMS Receiver is pictured on the boundary as it can be a service provided by a dedicated SMS gateway provider or a third-party application interacting with mobile network modem. Though due to the differences in SMS solutions - some additional processing might be needed.

Once message is received in full it is passed to a routing modules. Routing pipeline is shown in A.7 , labeled as SMS Router Here we clearly see that PKI services can be managed per phone number (MSISDN), certificate type OpenPGP or X.509, where X.509 can be further split into domain specific sub-PKIs. In a simplest setup, certificates would only be routed based on a certificate type to a single OpenPGP or X.509 certificate sub-PKI, to handle any requests.

4.3.1.5 HTTPS Interface

As voiced earlier, HTTPS interface to SMSPKI, has somewhat limited functionality compared to functionality provided through SMS interface. Access through HTTPS only provides certificate querying and retrieving.

By querying we mean certificate database look-ups on following full or partial fields: Unique Certificate Key within SMSPKI, Certificate Serial Number and Certificate Fingerprint. As partial field searches are allowed, multiple row results can be expected. Tough we highly recommend to not serve look-ups, that discover certificate count over a certain threshold, similarly to OpenPGP SKS PKI as mentioned in section 2.2. Limitation is to protect from a mass certificate extraction from a PKI. On the other hand, certificate retrieving can only be done using it's UCK, akin to certificate retrieving through SMS interface.

As we allow for multiple certificate databases (through routing in section 4.3.1.4 in our design, these different database should be addressable by different addresses, such as distinct Domain Names, or additional PKI instance information

in URL path of the HTTP request.

Since SMSPKI is not meant to be interacted with by an end-user's or any human users, but to be used for automatization purposes by SMSPKI providers, third-party service providers and SMSPKI clients utilising our PKI. Therefore, we strongly recommend to use server certificates and client certificates for access control via TLS system using HTTPS protocol. If providing access to SMSPKI clients, specific certificate already registered with this database could be used to identify and authenticate the client. If PKI services are provided to third-party service providers, we recommend setting up and independent certificates of our PKI to establish a secure channel trusted by both parties. In any cases SMSPKI should use a server certificate that is validated by CA and is trusted by the client systems.

4.3.1.6 Certificate Management

Certificate registration. When certificate is submitted for a registration, through SMS interface and routed to a right sub-PKI service, it is validated first. Validation process encompasses of following steps:

- if possible validating that clients MSISDN is not spoofed;
- checking if certificate is formed according to certificate specifications;
- checking if there already isn't a key with same UCK.

On the assumption that certificate has passed validation process, it is signed by sub-PKI service provider and appended to certificate database.

Certificate signing. Certificates are signed with a Private Root PKI Key. By signing a certificate that is being registered, SMSPKI provider implies that certificate is valid and is part of the relevant certificate database. For certificate validity checking purposes, we provide retrieving of Root certificate in the same manner as any other key, though we use a Nil UUID (special case of UUID, where all numbers are zeros) as Root Certificates UCK. Registration of certificates with Nil UUID must be forbidden.

Revoking certificates. Certificate revocation is similar to registration process, though when request is routed to a relevant sub-PKI, PKI checks validity of revocation request against the to-be-revoked public-key and processes according to the type of certificate, discussed in more detail in 4.3.1.6 and following sections.

Certificate expiration. OpenPGP and X.509 certificates, both support certificate expiration fields. X.509 certificates require that start and end dates (valid period) fields are present, whereas OpenPGP system supports certificate expiration date field, but is not required. Our recommendation is to always employ expiration fields as it could provide a viable solution to limit continuous growth of certificate databases. In our opinion, SMSPKI providers should choose certificate validity period according to the type of service they offer. For example, SMSPKI could provide a certificate store for certificates with short period validity, thus creating a rapidly changing certificate database; on the other hand, if certificates are intended for a long time use, SMSPKI could require longer validity periods. In general, certificate expiration should be seen as tool to keep certificate databases fresh.

X.509 Certificate Management

Specifically in the case of X.509, we use Root Certificate for certifying (signing) client certificates that are being registered. CA's Root Certificate here can be trusted system wide as well as trusted only in applications, by SMSPKI clients. Though, it is important that clients trust that Root certificate and are able to acquire trust chain for validating certificates within that trust domain. Root certificates can be retrieved using Nil UUID as voiced in 4.3.1.1 section.

As for certificate revocation, users are able to upload certificate revocation request. SMSPKI in order could add this certificate to a Certificate Revocation List (CRL). Then client could check if certificate is still valid by looking up said CRL.

OpenPGP Certificate Management

As OpenPGP employs a flat certificate infrastructure, we specify Master keypair for OpenPGP sub-PKI as another OpenPGP certificate with special purpose. This Master certificate and its private-key is used to sign any validated OpenPGP type certificates and can be retrieved with Nil UUID for certificate validation by SMSPKI clients. In a way it's a stepping stone of building a Web-of-trust model, where SMSPKI is explicitly imparting trust on that certificate by signing it. All certificates within OpenPGP sub-PKI must be signed with a Master key. Additionally, SMSPKI are able to re-register third-party certificates signed by other clients using the same certificate registration pathways. Thus, it's possible to create a meaningful Web-of-trust network, though, on the client side this is accomplished by software developer and not human individual, in comparison to original OpenPGP. Nevertheless, software developers retain power of including a human user supervision in this process through apps User Interface (UI).

OpenPGP client certificates can be revoked by registering a properly signed revocation certificate, using SMSPKI certificate registration pathway. In such case certificate can be labeled as revoked and/or removed from a certificate database.

If using OpenPGP certificates with expiration dates, we can also provide certificate validity extensions, again using the same certificate registration pathway. In this case, sub-PKI matches the certificate that is being extended and replaces it with a new one, given that certificate is properly formed and signed using the same corresponding private-key.

4.3.1.7 Distributed SMSPKI

As we introduced in our overview of OpenPGP PKIs in 2.2 section, OpenPGP SKS keyservers are able to provide certificate database synchronisations amongst multiple servers in the same pool. Additionally, considering the fact that SKS keyservers store certificates in key-value structures, we propose of adapting SKS keyservers infrastructure for OpenPGP as well as X.509 type of certificates - as both cases are merely a key-value objects.

Synchronisation of separate SMSPKI databases could allow a PKI provider to distribute their infrastructure around the world. Distributing infrastructure, could enable deployment of dedicated services per different geographic locations and would allow SMSPKI providers to facilitate global seamless PKI services.

Furthermore, synchronising multiple servers provide a suitable design feature for providing a fallback for PKI services and easy load balancing in busy environments.

On the other hand, having more than one certificate entry point increases risks of certificate UCK collisions. There is no clear method for solving this issue, though within trusted pool of servers, certificate registered earlier could take precedence.

4.3.2 SMSPKI Server API Description

SMSPKI API's are harmonised with HKP protocol specification proposed at [Sha03]. As mentioned in earlier sections due to limitations of SMS channel we can only provide a limited set of HKP operations. Additionally, as HTTPS

interface is used for only support purposes alongside SMS channel and other considerations expressed in section 4.3.1.5 provides an intentionally limited access, as well.

4.3.2.1 SMS API Request and Response Formats

Request format. When submitting requests to PKI using HKP over HTTPS, we use absolute path of HTTP request to indicate operations and to provide any parameters. As on SMS channel we do not have any equivalent to HTTPS URL path, we include it into the text message, prepended with label `req:`.

Furthermore, as SMS channel does not provide any stateful information on it's protocol we add a Session ID code together with our requests. Session ID, is used to separate different requests and responses, thus allowing multiple requests from the same client and receiving responses in any order. Session ID is to be transferred as part of the text message preceding with `sid:` label. As clients do not require constant communication and is used infrequently, we advice to use a four digit number as Session ID and possibly treat it like a circular counter. An example of request query is shown in A.14 figure.

Response format. Analogously, response message format is very similar to a request. In a responses we use `res:` label to provide HTTP error and success codes. Session ID is duplicated by SMSPKI from a corresponding request. If a response is expected to return a certificate it will be in PEM format, as well. An example of request message show in A.15 picture.

4.3.3 SMS API Description and Examples

Further, we specify concrete functions provided by SMSPKI SMS interface.

4.3.3.1 Certificate Registration

In order to register a certificate, client forms a request message, with appropriate `req:` and `sid:` fields, followed by a certificate encoded in PEM format. For, certificate registration, request line would be `req:/pks/add`. An example of such request is shown in A.14 figure, note that `sid:` here is an arbitrary number.

If registration is successful, SMSPKI service responds with HTTP status code

200 and signed certificate, otherwise appropriate HTTP error code is returned. Example of a response to a successful request is shown in A.15.

4.3.3.2 Certificate Retrieval

For certificate retrieval we use `/pks/lookup` path with appropriate options. For example, if we to retrieve a Master Certificate of given SMSPKI, and remembering that Master Certificates have a Nil UUID, full request field would be `req://pks/lookup?op=get&search=0000000000000000`.

An example of full request is shown in A.14 figure, where `sid:` field is an arbitrary session number.

As indicated earlier, over SMS interface we only provide key retrievals by it's UCK and request with any other options should return HTTP 501 error code, meaning that operation is not implemented.

If the request is successful, server responds with the message formatted shown in A.15 figure, with corresponding certificate included - in our example it would be Master certificate of the PKI. Alternatively, an appropriate HTTP error code is returned.

4.3.4 HTTPS API Description

In general HTTP API behaves like described in HKP specification in [SBKH06]. Though as we limit access through HTTP interface, certificate registration operations should be responded with HTTP error code 501 Not Implemented.

4.3.5 Important Considerations for SMSPKI's Server-side

4.3.5.1 SMS Channel and Certificate sizes

As we noted in section 4.1.0.2, that long text messages are broken down into smaller 145-160 character long messages. Therefore, PKI providers should consider the type of certificates and key bit size that they choose to use. An easy way to estimate how many single SMS packets will be required to transfer a request or response with public key is using following formula:

$$SMS_{cnt} = \lceil (MSG_{len} - 160)/145 + 1 \rceil$$

SMS_{cnt} is an estimated amount of messages that would be required to transfer a request (including any certificates) of length MSG_{len}. Here 160 is first SMS message character capacity and 145 for the rest of SMS packets.

OpenPGP certificates support RSA and DSA with ElGamal public-key algorithms. For RSA key recommended key size is at least 2048 bits, though keys up to 4096 bits are supported. To transfer RSA keys 16-23 SMS packets are required, depending on the key size. In the case of DSA with ElGamal public-key algorithm minimum recommended key size is 2048 bits and supported key size of up to 3072 bits. Estimates for SMS packets are equivalent to RSA's case.

Remembering from section 2.3.1.2, when requesting X.509 type certificate validation from a CA, a Certificate Signing Request (CSR) is sent. A CSR can be sent in 4-15 SMS packets depending on which certificate fields are used and the type of certificate. With X.509 certificates RSA and Elliptic curve (EC) public-key algorithms are used. Recommended key size for Elliptic curve algorithm is 256 bits.

Nevertheless, when considering appropriate certificate types and implications of the choice on the SMS channel, software developers should consider pricing models of SMS messages. More importantly with increasing amount of text messages per operation we are increasing risks levels, for example risks of losing or delaying an SMS packet which is part of a request. Therefore an appropriate balance should be set between length of text messages and mobile network reliability and costs.

To minimise amount of messages need for transfer we recommend using certificates with Elliptic curve key-pairs.

4.4 SMSPKI Client-Side

As introduced in section 4.2, on the client side we have SMS Certificate Manager (CM), which provides abstraction of our PKI functions over a local API service to client applications utilising PKI infrastructure. Firstly, we overview a client-side infrastructure from a general perspective and later we discuss implementation specifics for mobile devices running Android operating systems with access to SMS services.

Returning to the figure A.5 where a client-side infrastructure is shown. Here we have a Certificate Manager, which has access to SMS interface of a device and implements SMSPKI API. Certificate Manager can be used by multiple client applications, which interact with the manager service over an Inter-Process Communication (IPC) mechanism - possibly provided by underlying operating system. CM exposes its API over IPC. All the communications with CM is initiated by client applications.

In a scenario, where a client application is trying to utilise PKI service, for example for a certificate registration, where process is visualised using UML activity diagram in A.8. Here steps are explained in order as labeled in a later diagram:

1. Client generates key-pair and prepares a certificate or a CSR, in case of X.509 certificates.
2. Client application passes certificate to appropriate CM's API function for registering certificates together with the MSISDN address of the SMSPKI service to be accessed. Additionally, client passes callback function to be executed once response is available from SMSPKI and then Certificate Manager.
3. Certificate Manager forms a registration request message as described in section 4.3.2.1. Here `sid:` field is added by the manager, which will be used for matching response messages to specific SMSPKI requests from a CM.
4. Request is sent as MO SMS, to the address provided by the client application in its request.
5. SMSPKI processes request and forms an appropriate response, with validated certificate or applicable HTTP error code.
6. Response is forwarded back to the MSISDN of the device from which MO SMS has originated.
7. Certificate Manager captures a response by inspecting incoming MT SMS messages and interprets a response from SMSPKI.
8. Certificate Manager invokes a pending callback, and passes a successfully registered certificate or raises an appropriate exception, that should be handled by a client application.

Described flow is applicable to all functions provided by a Certificate Manager and SMSPKI infrastructure in general.

Certificate Manager is not a required part of our PKI infrastructure, as PKI could be used by any client implementing our SMSHKP protocol needed to access API of SMSPKI. Although we introduce this component most of all to abstract SMS channel from client software developers as well as to improve transparency of this infrastructure, from a point of view of an end-user; discussed in more detail in following section.

4.4.1 SMS Certificate Manager

Purpose of Certificate Manager is to mediate communications between PKI clients and SMSPKI server-side, by abstracting all PKI functions in a CM. This serves as a convenient abstraction layer that further distances app dependencies, in our case - SMS channel. Also, if clients were to use infrastructure directly over SMS channel, they would require permissions to SMS services per every client application.

Another essential feature provided by CM is to store revocation certificates. CM stores revocation certificates so it would be possible to revoke a specific certificates even if client application is uninstalled before revoking any certificates it has registered. This feature is intended to be used as a last resort in certificate management by the end-user themselves.

Considering that revocation certificates can only be generated with the help of a private-key corresponding to the particular certificate and that private-keys are owned only by a client applications. For that reason, CM must be supplied with a revocation certificate by a client application. We require revocation certificates to be generated at the same time as key-pair is generated by the client application, and that both are provided to a CM, when attempting a certificate registration. Though, only a certificate part would be forwarded to SMSPKI providers and revocation certificate stored for later use.

As we, discussed in 4.3.1.3 section, a single SMSPKI service can have more than one MSISDN associated. In addition we expect that SMSPKI service can be provided by multiple service providers. Therefore, we suggest grouping of certificates (creating a folder view) in CM per unique MSISDN of the SMSPKI basis.

Grouping certificates would provide a valuable and constructive feedback to an end-user on following objections:

- MSISDN addresses of SMSPKI that their system utilises, in particular user

would be able to infer a possible geographic location based on country code of MSISDN.

- Identify which applications is utilising said PKI.
- Type and amount of certificates used with particular SMSPKI provider.

4.4.1.1 SMS Certificate Manager Design

High level design overview of a Certificate Manager is shown in figure A.9. Here we see folder views per particular MSISDN provider and certificates associated with it. Furthermore, we show that CM provides an interface over IPC, through which API for clients is exposed. CM implements SMS HKP protocol and uses SMS channel to communicate to any SMSPKI providers.

In folder views, per particular MSISDN, we show that multiple certificates are associated (registered) with particular MSISDN. In addition all certificates have a revocation certificates per every registered certificate.

4.4.2 SMS Certificate Manager API and Internals

API provided by CM is visualised in UML Class diagram in A.10. As can be seen from a diagram, API provides just three functions addCert(), appendCert() and getCert(). Even though from first look it seems very rudimentary API, but as we will see, these functions allow us to accomplish all required operations to successfully distribute and publish certificates.

Next we overview how and in what scenarios API functions should be used.

4.4.2.1 Function addCert()

Function is to be used only when registering personal new certificates. From function signature in a class diagram in figure A.10 we see, that developer has to pass MSISDN (phone number of the SMSPKI), certificate and it's revocation certificate passed as strings containing ASCII armoured certificates in PEM format.

Flow of certificate registration is visualised in figure A.11 by a UML activity diagram. From a diagram we can see the preliminary steps taken by Certificate

Manager and at which point it communicates with SMSPKI server. To handle cases when we do not receive any reply from a SMSPKI service we introduce a timeout notion in our CM. As we see from a A.11 diagram, if registration fails at any point due to invalid or malformed data, or timeouts when waiting for a reply, client is returned an error message, and any certificates pending for a registration are removed.

4.4.2.2 Function appendCert()

This function can be used to update personal certificates, for example extending their validity date and such. Also, function in question can be used to publish a third-party certificates signed with clients own private-key for building Web-of-trust network amongst the certificates.

Actions taken by a Certificate Manager and SMSPKI when invoking function in question is shown in A.12 figure. From a function's signature in API's class diagram, we see that it requires PKI's address and a certificate to be updated.

4.4.2.3 Function getCert()

From a name we can deduce that this function is for a certificate look-up and retrieval from an SMSPKI. As specified in 4.3.1.3, this function is for retrieving single certificates. Function takes an address of SMSPKI and a full UCK of a certificate to be retrieved. This function can be used to retrieve any existing certificate as well as Root keys of sub-PKIs, given that client knows a precise UCK of a relevant certificate.

Action flow of certificate retrieval is shown in A.13 diagram.

4.4.2.4 CM API Error Handling

In section 4.3.3, we have specified that SMSPKI uses HTTP error codes. For a client applications we define relevant exceptions - these can be inferred from activity diagrams as well. Certificate Manager either raises exception internally or interprets HTTP error code and raises an appropriate exception. We define a following exception types, with their descriptions:

- *InvalidRequestParameters* - raised when parameters to CM's API is invalid

(invalid certificate, invalid MSISDN of the SMSPKI provider or invalid certificate UCK).

- *InvalidApiRequest* - raised when SMSPKI can not handle a request, possibly not supported procedure or malformed text message.
- *CertificateNotFound* - raised, if SMSPKI did not find a certificate that was looked-up.
- *CertificateExists* - raised, if trying to register a certificate with a UCK that is already registered with another certificate (collision of UCKs).
- *RequestTimedOut* - raised, if CM did not receive a response within timeout period.
- *ServiceNotAvailable* - raised, if mobile network is unavailable.

4.4.3 Considerations for CM Implementation on Mobile Devices

iOS. Unfortunately, iPhone operating system would not be able to utilise our SMSMPKI. The main reason is that iPhone operating system does not provide access to SMS interface and therefore channel itself. Moreover, iOS operating system does not have a dedicated IPC mechanism for providing native communications between different apps. Considering, how closed and conservative the iOS platform is, currently only Apple have access and possibility to implement a Certificate Manager providing API services to other apps and accessing SMS interface/channel.

4.4.3.1 Android OS

On the other hand, Android OS provides a very flexible infrastructure with wide controlled access to system resources and a few IPC methods. To be able to utilise system resources it's necessary to request relevant resource permissions.

The only resource that CM requires is to be able to read and write SMS messages, as well as subscribe for incoming message notifications. To accomplish CM resource requirements developer has to include the following code in application manifest file:

```
1 <manifest ...>
2     ...
```

```
3     <application ...>
4     <!-- Subscribe to incoming SMS messages. -->
5     <receiver android:name=".SMSReceiverCM">
6     <!-- Class that will receive incoming SMS -->
7             <intent-filter>
8                 <action android:name="android.provider.Telephony.
      SMS_RECEIVED" />
9             </intent-filter>
10        </receiver>
11    </application>
12    ...
13    <!-- Acquire necessary permissions. -->
14    <uses-permission android:name="android.permission.SEND_SMS">
15    </uses-permission>
16    <uses-permission android:name="android.permission.RECEIVE_SMS">
17    </uses-permission>
18    ...
19 </manifest>
```

To send an SMS, Android OS API provides several different ways, though as we are using ASCII based text messages, we do not require extraneous text message formation and can use the most straightforward method. A sample code of sending a text message:

```
1     SmsManager sm = SmsManager.getDefault();
2     sms.sendTextMessage(SMSPKI_MSISDN, null, requestMsg, ......);
```

A function signature of the sentTextMessage() function from a documentation:

```
1     sendTextMessage(String destinationAddress, String scAddress, String
      text, PendingIntent sentIntent, PendingIntent deliveryIntent);
```

Here, `destinationAddress` in our case is MSISDN of the SMSPKI provider; `scAdress` is SMSC MSISDN, if null system default is used; `text` is SMS body, in our case a formed response to SMSPKI; `sentIntent` and `deilveryIntent` is used as callbacks, for when a message has been sent to network, and when it was delivered correspondingly.

To provide CM service and API through IPC, we highly recommend on using Bound Services [And] method, for the reason that, service implemented with `Service` class an binded to by other activities (apps) and only lives when serving these activities. As Certificate Manager is not needed often and as it is always invoked by clients (activities from client apps) Bound Service is an appropriate method in our situation.

4.4.4 SMS Certificate Manager Clients

4.4.4.1 Binding to service

SMS CM clients must bind to the CM offered service using, using services `bindService()` function. As discussed in prior section clients implement and use `Activity` class to access a service. This activity would be created only when actively using CM's service interface. Multiple clients is supported as well as asynchronous execution of the service code part. Service can use binded `Intent` object from the activity to return any calls - to be used as a callback, when result is available.

4.4.4.2 Key-pair Generation and Storage

For generating key-pairs we require clients to use KeyStore provided by an Android OS. KeyStore provides an abstraction layer for creating symmetric and asymmetric (public-key) keys. KeyStore supports these public-key key-pair generation algorithms: Digital Signature Algorithm (DSA), Elliptic-curves (EC) and RSA type of keys. Key generation is provided by `java.security.KeyPairGenerator` class.

Keystore provides not only an abstraction layer for generating keys, but also provides storage for generated key-pairs, which is often backed by secure hardware, such as Trusted Execution Environment (TEE) or Secure Element (SE). TEE provided by ARM processors, uses hardware virtualization to create a sandboxed environment for storing secrets. On the other hand, SE provides a dedicated hardware element for storing keys, and is likely to have a dedicated crypto-coprocessor for working with said keys.

Additionally, KS provides a flexible interface of protecting generated key-pair. For example, when generating a key-pair developer can set, if key-pair could be exported from a KeyStore or not. KeyStore is strongly associated to system wide locking mechanisms, such as, pin code, pattern lock, password protection and biometric fingerprint protection. Once, device is locked all KeyStore instances are locked as well. After unlocking a device KS, can be configured to be unlocked together with a device, or unlocked separately on demand. Devices that have biometric fingerprint hardware, can be set to request additional authentication step.

4.4.4.3 Certificate Generation

Natively, Android OS supports a narrow range of functions related to X.509 certificate creation and usage. Though, Android OS provides a SpongyCastle cryptographic API suite for any operations related to certificate creation and usage. SpongyCastle is a customized variant of a well known BouncyCastle cryptographic API suite and is dedicated to Android OS. SpongyCastle is able to use KeyStore, for utilizing any stored key-pair.

SpongyCastle supports X.509 certificate, as well as OpenPGP certificate infrastructures. Furthermore, SpongyCastle implements and facilitates many cryptographic protocols and procedures, that assist with communication channel security and cryptographic element interpretation and formation (such as different key-pair formats).

As we can see this cryptographic suite fully covers our needs and requirements, and not only in working with our SMSPKI and SMS CM, but also in applying our cryptographic elements in communications.

Important Considerations

KeyStore was introduced since Android 4.3 version (API level 18 - in Google's API versioning) codenamed JellyBean, released in 2012 summer. As stated earlier, we require SMS CM clients to use KeyStore as it provides additional layer of security for protecting a private-key. To use certificates securely we put highest requirements on protecting a private-key. Using KeyStore allows for securely maintaining passwordless private-key, unlike in most environments, where end-users had to decrypt their private-keys prior to using them.

Underlying functionality and any hardware backing for KeyStore is implemented by a device manufacturers. In past, some early implementations of KeyStore was found to be vulnerable [HD14].

SpongyCastle cryptography suite must be included explicitly as app dependency, therefore every SMS CM client will have to include the library with it's app deployment.

Chapter 5

Usage of Certificates Registered with SMSPKI

As we are considering certificate application in computer network communications, we need to define in what setups certificates could be used in, from the communication service point of view, taking into consideration of an underlying network topology.

Majority of all communications on the network follow client-server communication model. In this model, client (party accessing a service or a resource) initiates communication establishment with a server (party providing a service or a resource). In order for server to be reached, it has to be publicly addressable and actively wait for any communications. This implies that the server must possess a public IP address. Moreover, clients must know or have an ability to find out an IP address of the service. Latter, is usually achieved with domain names and DNS system. Dynamically changing IP address presents a challenge of addressing hosts only by their IP address, as is often the case with typical clients.

In addition, revisiting 1.3.3.1 section, where we have discussed a major differences in IPv4 and IPv6 networks, we recall that on IPv4 network connected devices don't always have a public IP address, for example, when a host is behind IPv4 gateway. It's impossible to reach such hosts, without a prior and

extensive network setup.

In a situation, where two devices behind separate gateways on IPv4 network, wish to establish a communication channel, they must access a dedicated service (server in client-server model) that either will proxy their communications or will help them actively establish a direct communication, for example using hole punching technique to circumvent gateway firewall. Preceding communication described can be seen as client-server-client communication model, where latter one is closer to peer to peer connection.

5.1 SMSPKI and Client-Server Setup

Here we assume that client software is provided by the same service provider providing services on the server, for example a banking app client accessing remote banking services.

Suppose, an end-user using banking app for the first time to access services related to his account. End-user successfully authenticates to the banking services using bank's standard login procedures. Once logged in, banking service could initiate our client app to generate a key-pair and try to register the certificate with certain SMSPKI provider. If certificate is successfully registered, service provider can store UCK of that certificate together with user account information. This achieves relating specific users to specific certificates. Though, service provider before relating certificate UCK to a user should retrieve said certificate (using SMS or HTTPS APIs of SMSPKI) and challenge user to prove that he possess corresponding private key.

Now, the next time user is connecting using the same device, for example using HTTPS protocol, banking service can require cryptographic authentication from a client app using TLS protocol. This can be seen as the case of setting up trusted devices (by service providers) per user.

Furthermore, client certificates could be used, for securing any communications between client and a server, as well as any public-key cryptography functionality, for example signing transactions and such.

5.2 SMSPKI and Client-Server-Client Setup

Let's assume that before attempting communication between two clients, these clients have enrolled with a service provider (server) in the same manner as in the client-server model, discussed in previous section. Therefore, two clients would already possess their individual certificates, of which service provider is aware of, and corresponding private-keys.

Now, if communication is proxied by a service provider, service provider can provide UCK of certificates related to these clients, so they could retrieve them from the SMSPKI service. Thus clients could use each other's certificates to secure data in transport, also protecting from exposing it to a service provider itself.

If service provider is using hole punching technique to setup a direct channel between clients, they can authenticate each other on retrieved certificates and establish a direct secure communication channel through one of the cryptographic protocols employing public-keys.

5.3 SMSPKI and Peer to Peer (P2P) Setup

P2P model, requires that both parties are publicly addressable, so they could reach each other directly. P2P connections is a hard challenge within IPv4 networks as many hosts are behind an internet gateway. On the other hand, as covered in 1.3.3.1, IPv6 networks exposes all hosts publicly. Both versions of the IP protocol presents us with a challenge of finding and addressing remote hosts. Also, to establish a network connection, one of the parties have to take a server and the other client roles. Here, party acting as a serve, creates a service waiting for any incoming connections.

Moreover, in this case we do not have a dedicated and mutually trusted service provider that facilitates exchange of certificate UCKs related to particular users. Therefore, an alternative is required.

Blindly using an arbitrary UCKs, do not guarantee communicating parties any identity properties and can only provide secrecy of the communications, between these two parties.

5.4 Mobile Network Security Implication

As discussed in section 4.1.0.3, attackers could try and perform 'MITM' attack between mobile device and BSS station. If performing 'MITM' for the purpose of sniffing traffic, attacker could possibly observe certificate registration procedure over SMS and make an association between anonymous certificate with certain UCK to a particular mobile device.

If 'MITM' is being performed to find out TMSI or IMSI subscriber identifiers, so attacker could simulate roaming of a device being attacked. Attacker could divert all MT SMS meant for a client to his rogue SMSC, thus intercepting all SMS messages. In such cases SMSPKI usage would be disrupted, though an attacker at best would only acquire a signed certificate - public-key part. On the other hand, Certificate Manager on a client device would timeout and discard that certificate.

Furthermore, having out-of-band communication increases difficulty and costs for a possible attack. For a successful attack, rogue party would have to sabotage any internet communications, likely an HTTPS channel, as well as perform 'MITM', between mobile device and a BSS.

Chapter 6

Conclusion

To conclude, we have overviewed pre-modern communication methods, from an information security standpoint, describing where applicable any security implications and familiarity of these methods to humans. In an overview of modern communication methods we demonstrated, that current communication methods are non-intuitive to its users. Further, we described how cryptography, especially public-key cryptography, is important for providing security properties to digital communications and identified that end-users are inadequate at performing certificate management operations. Therefore, we argued for transferring these responsibilities to underlying software developers.

In addition, we defined a Public-Key infrastructure - SMSPKI, that employs SMS channel for OpenPGP and X.509 certificate publishing and retrieval. To utilise SMSPKI we described client-side software setup, including SMS Certificate Manager, which provides API for 3rd party apps that wish to access SMSPKI services. Thanks to this API service we where able to describe how certificate management operations can be relied to underlying app developers.

For future research, it would be compelling to investigate alternatives to SMS channel based solution, for example, publishing certificates in blockchain network in addition to using crypto currency as a limited resource. Furthermore, Certificate Manager could be extended to provide these new types of server-side solutions, though providing a consistent API through all client apps.

Appendix A

Diagrams

A.1 Description Supporting Figures

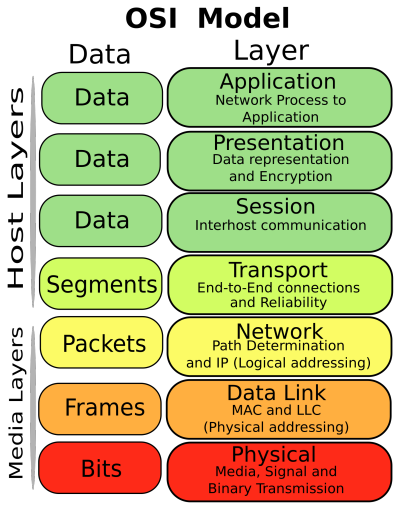

Figure A.1: "OSI RM Model" by Gorivero, used under CC BY 3.0

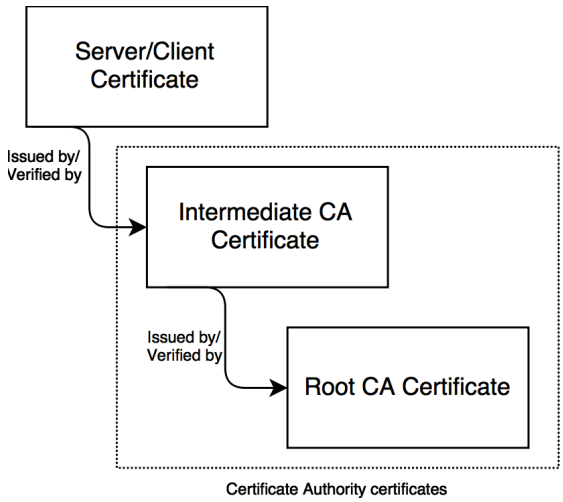

Figure A.2: Validation and Trust chain of X.509 certificates

A.2 SMSPKI Related Figures

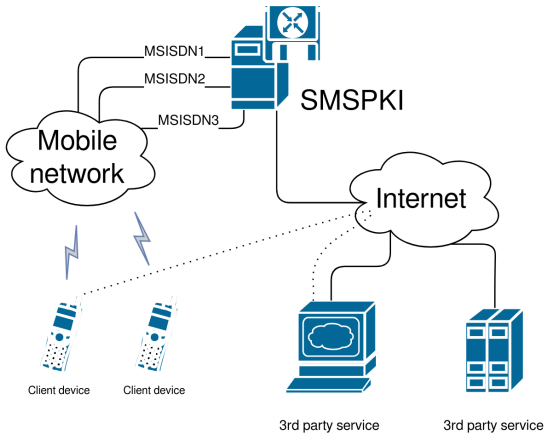

Figure A.3: General overview of SMSPKI infrastructure and its clients.

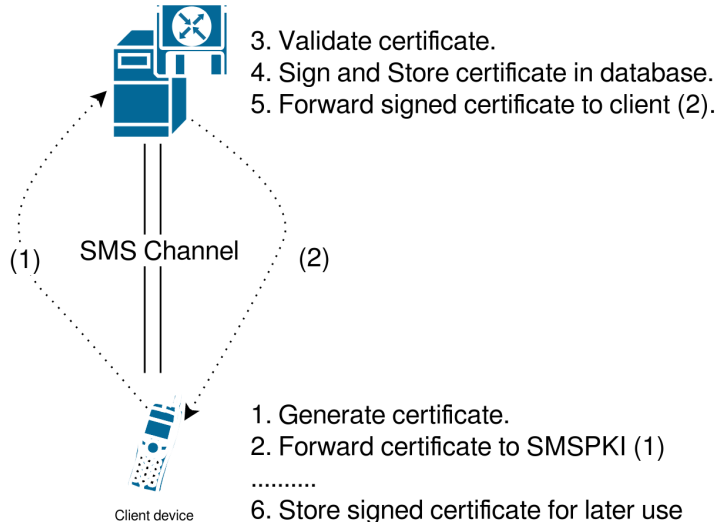

Figure A.4: Simplified certificate registration diagram.

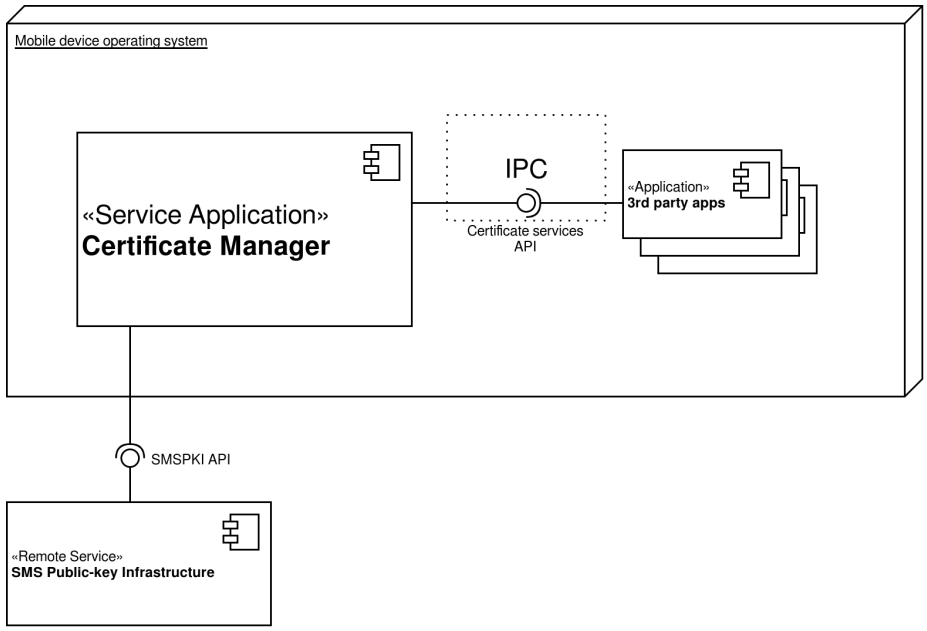

Figure A.5: Client side setup with SMS Certificate Manager its interfaces and 3rd party app as API clients.

Figure A.6: High-level overview of SMSPKI server-side with its sub-PKIs associated with MSISDN(s) and their HTTP interfaces.

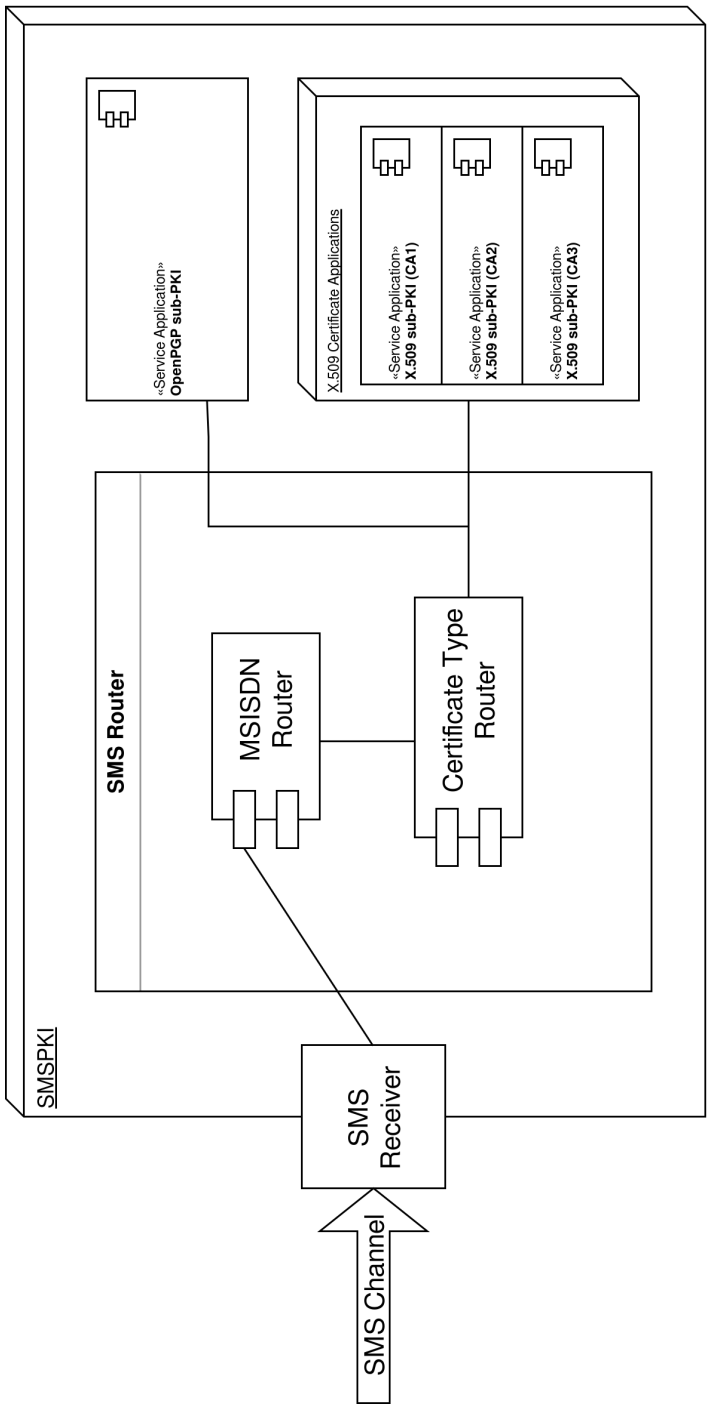

Figure A.7: SMSPKI SMS receiving and routing components connected to sub-PKIs.

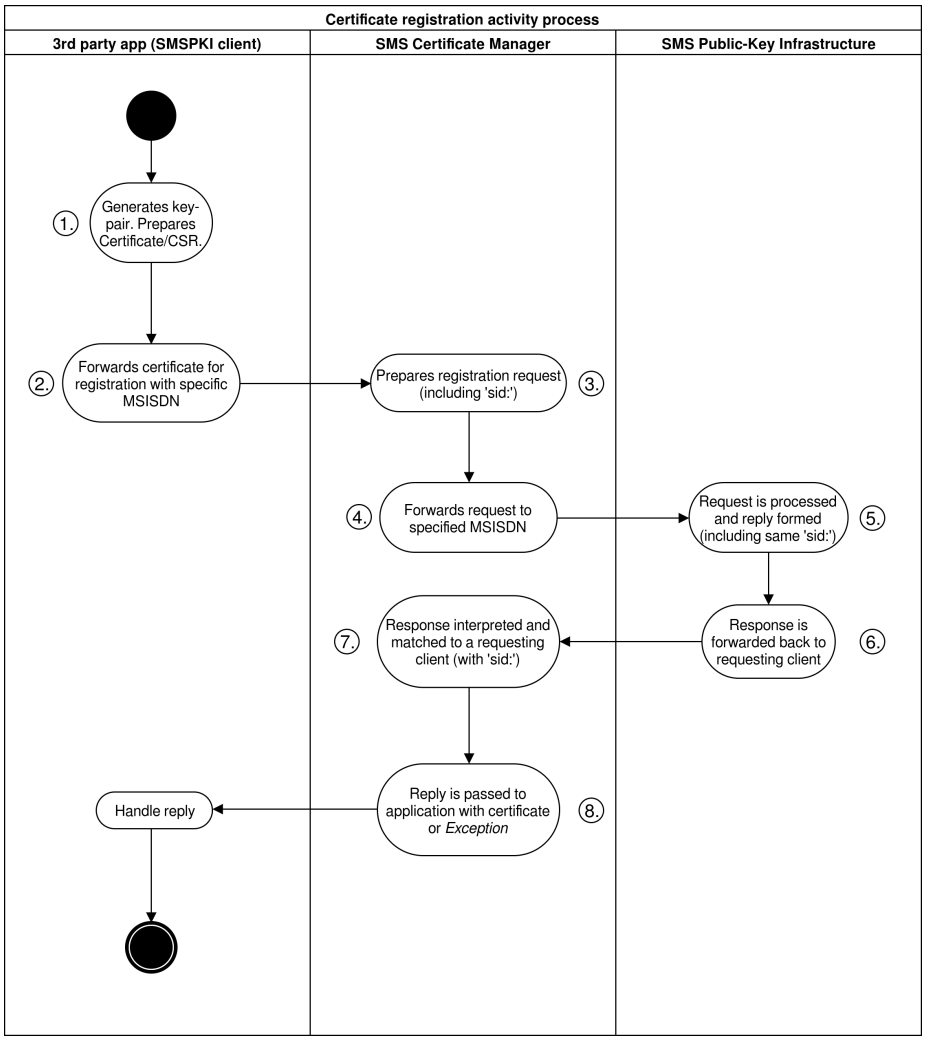

Figure A.8: Interaction between 3rd party client apps, SMS Certificate Manager and SMSPKI server-side.

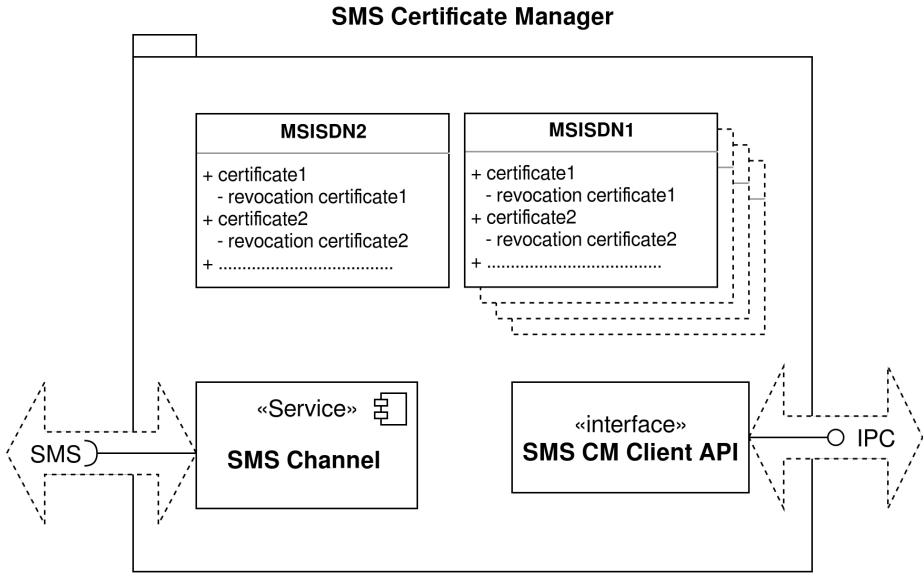

Figure A.9: Certificate Manager high-level overview. MSISDN(1/2) separate folders for different SMSPKI providers with its provided and implemented interfaces.

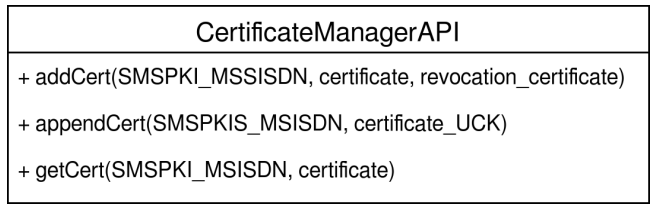

Figure A.10: SMS Certificate Manager API service interface definition.

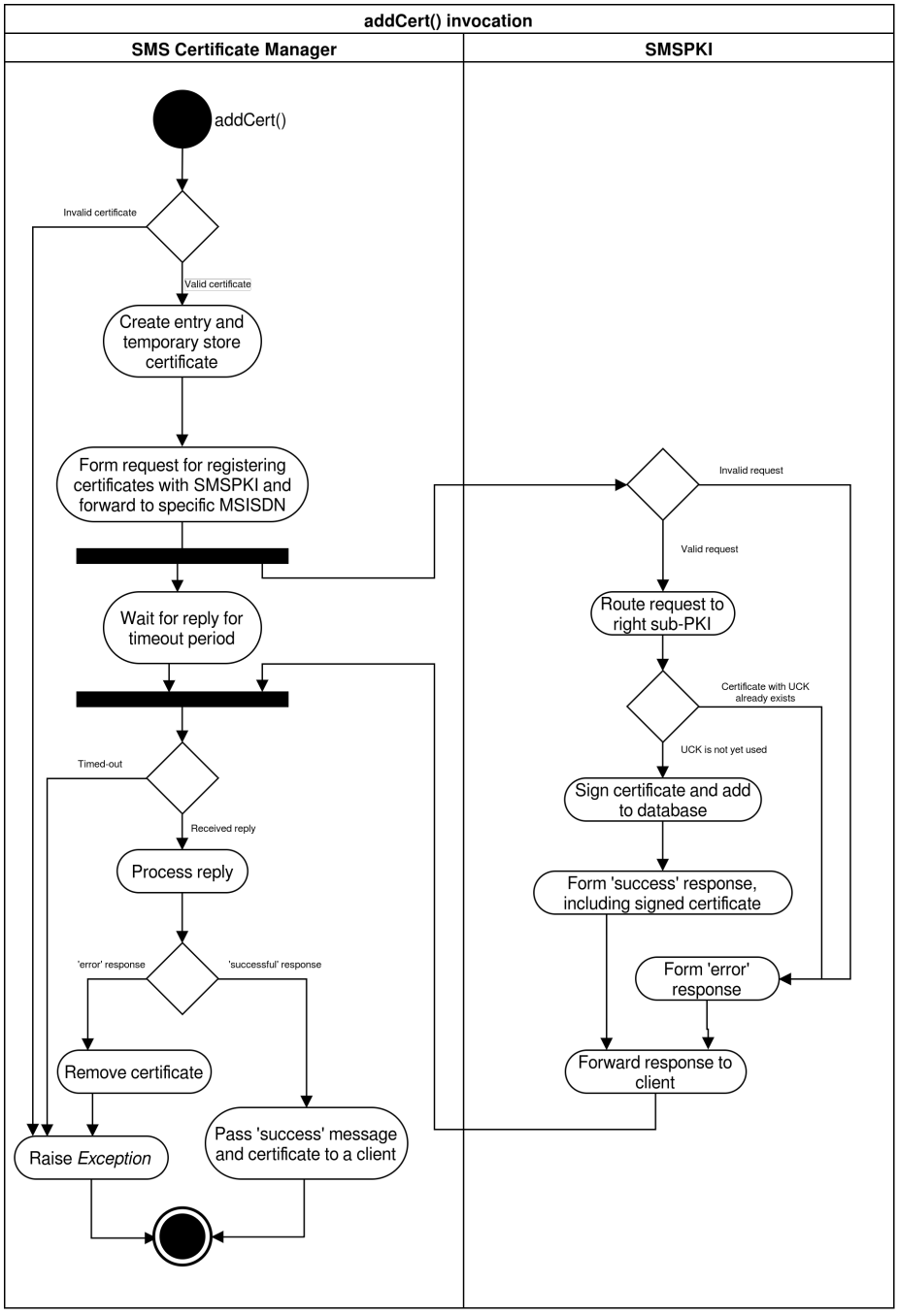

Figure A.11: Certificate registration activity flow and interaction between Certificate Manager and SMSPKI, in context.

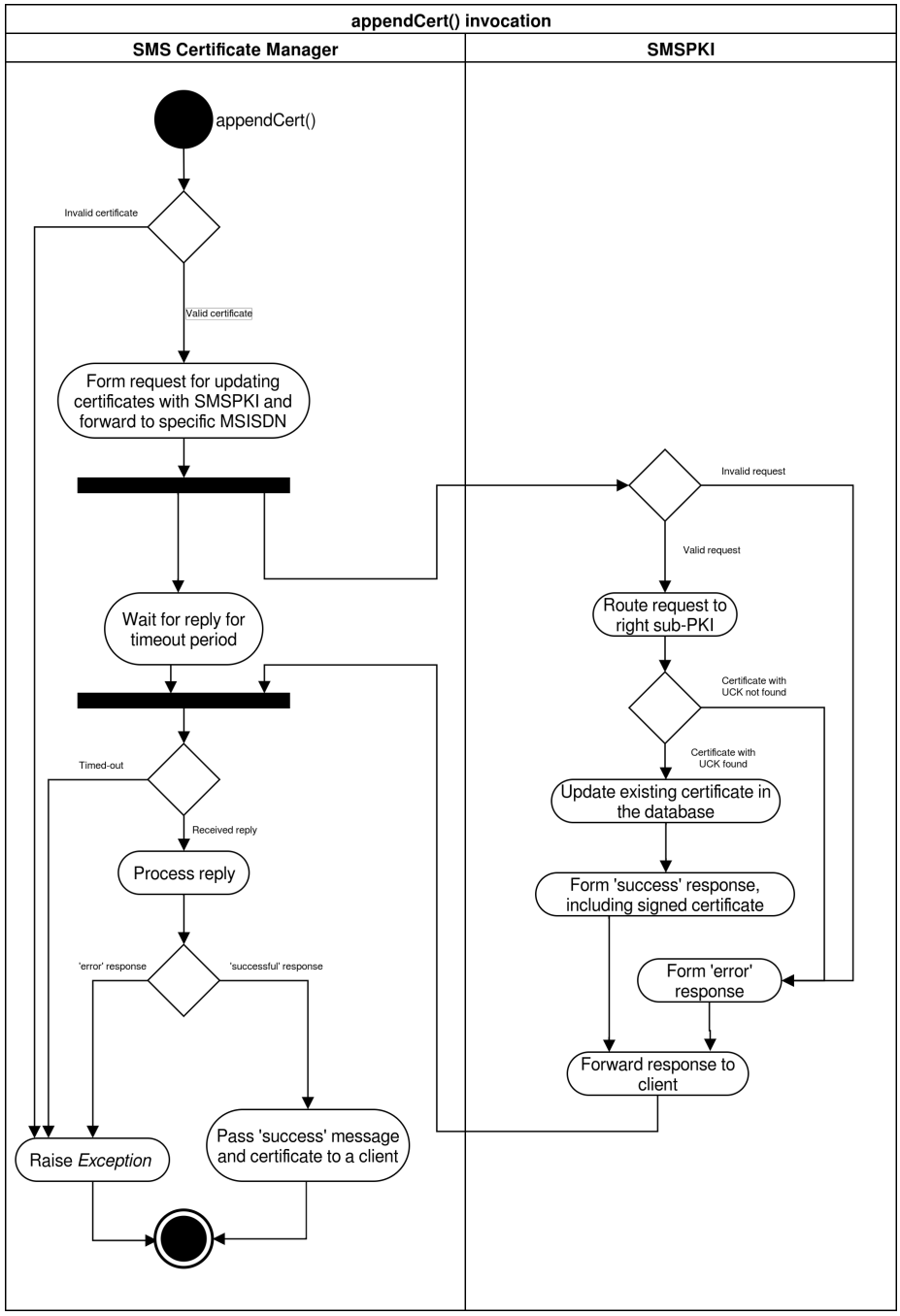

Figure A.12: Certificate updating activity flow and interaction between Certificate Manager and SMSPKI, in context.

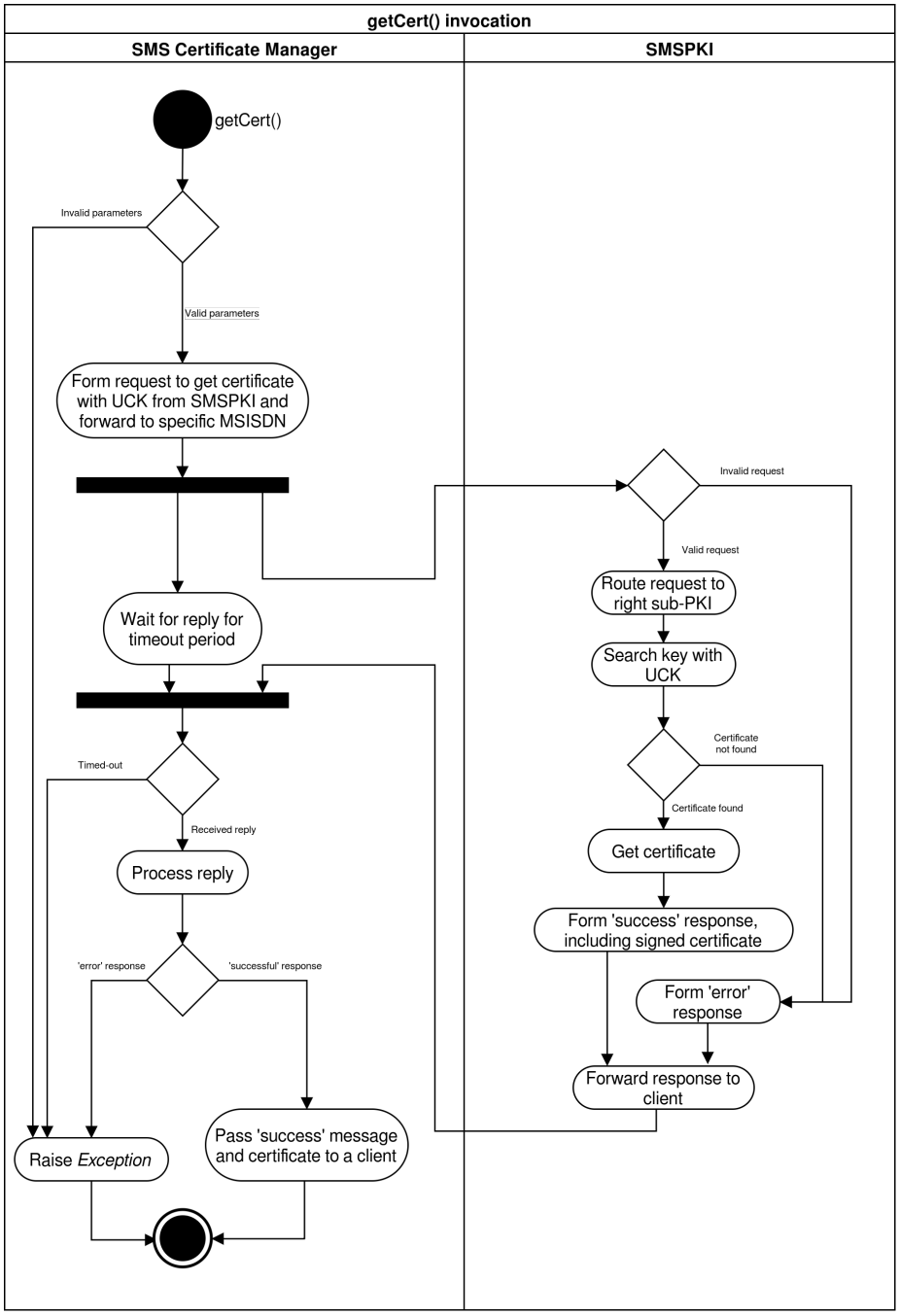

Figure A.13: Certificate retrieval activity flow and interaction between Certificate Manager and SMSPKI, in context.

A.3 Request and Response Examples

```
req:/pks/add
sid:3213
-----BEGIN CERTIFICATE REQUEST-----
MIICwTCCAakCAQAwfDELMAkGA1UEBhMCREsxDDAKBgNVBAgMA2tiaDEMMAoGA1UE
BwwDa2JoMQwwCgYDVQQKDANEVFUxDDAKBgNVBAsMA0RUVTEZMBcGA1UEAwwQMDAw
MDAwMDAwMDAwMDAwMDEaMBgGCSqGSIb3DQEJARYLc21zQHBraS5vcmcwggEiMA0G
CSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDTakmZZHNRIlGUsGD8g8CSISHrH6mw
1e/l9Pj3ujeybU+uYxh2Sd+808J6us4grZcsxeUEw9VHxVPcotQ0Kt7V561jYJeW
3ybPyCemoi9gMZzMRhIZETPU1OThpJyl+23MAa46CHjRNEPKEzy7RO/SAJXPWOlF
BzzCzsxaWFsSUL4goQHDpg50LOPAjCeJXpf/nPMMY3dUwgW6RcEwa63UQsWD+jXP
V8DLllP/AXWhipmf+S01WYD2GvqUMwSoIdJ1MxjZ9/Ajrp4lsubD1jo4iK7U/P/o
wsDbORjrTzVASAMTFErVL0LkpV9Gl6Ve2UbeR/l7gXVktOGNdDIwDp2fAgMBAAGg
ADANBgkqhkiG9w0BAQsFAAOCAQEAdD/+tMuW1x66aouNDwlYdPdkKTP6IjLvEp2Y
ZLMS/FSCm38nx2PJyJ/OjRd4H1lQ8FuIUWFtYePqCt/WrOzdul5zizVNpCS3saZa
79fJqfdadCoo7+yZa9aqBUcXyPt7npaWIlE6tIS93EaCDNH8dgjZa3pXPyD+T4sP
PHCD73XaKIlbLqAPgMGQjneb6n7cP59L8XO8H99KxypmEP1NdMriznTEdE+zjP+f
dsNDsp0A7VKokHmN61xjbiwJc4FwP1q22/f1fqLmaBRJtKKkNBMz1PV74udsVmBn
SJuKzw6qO2rtjcyyNXCjFTiUBB/wVZULs9ZWEajaKzqum8hLgQ==
-----END CERTIFICATE REQUEST-----
```

Figure A.14: An example of certificate registration request. Included is X.509 CSR certificate in PEM format. *req* : - requested operation, *sid* : - arbitrary number to track session, between requests and responses.

res:200
sid:3213
-----BEGIN PGP PUBLIC KEY BLOCK-----
mQENBEsBUPIBCADCHItsBbQ//q32FAS5tzRLr23rulqQe0WpssnQM6cC0qvVwfuqEmI9IncT
wtyVzF33HJ7quREw79Y4E01AOlvmP0iUxr7ppfAdecKiPLR5bwVdZR1fuVUs42d8aIhBt9N8
/dn/k+CbZCJ89KO4C9ArOa8mSe3mCnNKdEVo1/qbjCGOAwmJqcNXGfja71LxG1veTwpCiiR2
A6FBT0BZXTkTHf2VOLrYDHrwQLEJDqOeWzxJqGnK0l7McLKUpUZ2NVJ6fVz4EadhvPpelo7x
FO9RZnZiATXKNiKXfawrXO0n6SraV9Re84ZTQS1H2FCO4Dib6GabCvytW15ZiMZ/vREZABEB
AAGJAR8EIAECAAkFAksBUPwCHQIACgkQAAWlgoAJ277wqAf/Ur+9VikF4Pd9/5nH3yxWEIUt
4zmVxCds/ghapx8XcwpsEhSXKrUVaqDmTg1C32CLjc3265J5b9BhpWaYFcvFv29A+11sU6JS
/GKFGClrO2c5CpATSs911j4XXNRRMUMhLU95JfMKwhWHyzeQ21DxgM0n3V+XwR+oxSAVi7u0
qvdSXwl/LT9n3rSkJaQvXbrTLdoSRN0NIIO9fN0Q3JqfEPrI4RBmQmycCek4rPUbSghioGRo
ZZdyTb25PMKlkO8DZ7yeGk9RUcAYhWNT9kiTXLHwjWtdwLsS3+w8Ed9pwyB8Npl02nGMWqcd
qPILm84enq9ph6JFPcmDqkl2Mhs4iLQ4R2VybWFuYXMgU2t1cmljaGluYXMgKG5vIHBhc3N3
b3JkKSA8c2dlcm1hbmFzQGdtYWlsLmNvbT6JATwEEwECACYFAksBUPICGyMFCQAGl4AGCwkI
BwMCBBUCCAMEFgIDAQIeAQIXgAAKCRAABaWCgAnbvmSQB/9JOMm3REXHHmHJBlhxQ9kWNLwt
ic2wWtOA+WN6lwqMoiSEEmjSInCOpOQGeGgIdFBYXTGEGolPAkpnbvn+lxiJLs0VImx16ZJV
fCSNuTJy7R+Wci4wkpye62DAZciCGhEWSSMNiKUgR1h8N41molupGswR932SZ4fvGav3zwpS
2/vcI1DqXdQTS/qxjbVg6URfQtwmYmhIXSLepQ3sK1PEmmcXxLjnziUCtmogzNWnHbLEIfEO
2c5t0SGIiatxvTO2uxEod6ufeXTiUVuX0VK0abkQ8s44xfVK29yEo84Tygpc/U0G9n2F6kq/
I98nM8Pb/DdR4rKDuW+hEbVlz6X7uQENBEsBUPIBCACyjNWcB0CT6wuWSjwzB8AFV+B4lzJU
/2WfVy9MjFUoDqvW8Oo/JBi9zI0XC4s2D1eZclIWuIVEw+sZu4cs2TjL2v1z5IpjNKiP79cO
Y3+pUmnkHEs2sB/v3/q9bExrcgvWAKy/45W7BUYgd85VSy7ank/VYFoBG6qnz0M1r9K232Aj
RoBj1V1A4HJdXHmgN/TwxQyI32xT2WFESEqxUO/TpHmtETutKJyd0hY0qiuyf+IKf4EDWbs5
gcqTjqN8O4a4Uv1tXHYg6Gm10iZz7hX3T8H2/3oLO+an9KbfcA3ffZl9qxwwp3WXmFwhfueR
XVY5BN8y2FEQKasfr4hv9Z7pABEBAAGJASUEGAECAA8FAksBUPICGwwFCQAGl4AACgkQAAWl
goAJ277Sugf/Zw1wg3X7tjnOkuP3ScEvUzMrJxPA1IDvZnFYlgYPmDb+uakQyQmZuUtTFvY6
6mHpqH9qNJPIZGxiTUpziXsRF4kM2BceRv+TEyd0aKz/dc0aMeFtWaMu6Jm0rfXIkpKbLlca
PSrDtiFK+mA4l6r6FNSkJe1eI+fF7toqso2FY2qB4vRTWw8HGTraPDLj3ikCRP+oqT6DhQ4K
dFXwAFk45vSDssqeIFPuwWZW57SS3kAxD4jxXnUptJJlEGqbxpD0M08aOCVQYgkxwNoaF1LE
D9Yb/XMIL4D+AW5PZOGYzkOio2hYhiMHUq36uEdTyWQhkGZOZRU74asmE2ZOxSiiTw==
=mxyR
-----END PGP PUBLIC KEY BLOCK-----

Figure A.15: Example of a response to a successful certificate registartion request.*res* : - HTTP status code, *sid* : - arbitrary number to track session and registered PGP type certificate in PEM format.

Bibliography

[And] Android™. Bound services. https://developer.android.com/guide/components/bound-services.html. Online; accessed: 2017-04-19.

[Arc] Internet Traffic Statistics Archive. Internet statistics. report for week 1, 2017. URL: http://stats.simpleweb.org/statistics.php?l=2&w=1&y=2017.

[BGB04] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use pgp. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 77–84. ACM, 2004.

[CDF+07] Jon Callas, Lutz Donnerhacke, Hal Finney, David Shaw, and Rodney Thayer. RFC 4880: OpenPGP Message Format. URL: https://tools.ietf.org/html/rfc4880, November 2007.

[CDH+05] Matt Cooper, Yuriy Dzambasow, Peter Hesse, Susan Joseph, and Richard Nicholas. RFC 4158: Internet X.509 Public Key Infrastructure: Certification Path Building. URL: https://tools.ietf.org/html/rfc4158, September 2005.

[CSF+08] D Cooper, S Santesson, S Farrell, S Boeyen, R Housley, and W Polk. Internet x. 509 public key infrastructure certificate and crl profile. *Internet Engineering Task Force. URL: http://tools.ietf.org/html/rfc5280*, 2008.

[DH14] S Deering and R Hinden. Rfc 2460-internet protocol, version 6 (ipv6) specification, 1998. *URL: http://www.ietf.org/rfc/rfc2460.txt*, 2014.

[EPS15] Chris Evans, Chris Palmer, and Ryan Sleevi. Rfc 7469: public key pinning extension for http. *Internet Engineering Task Force. URL: http://tools.ietf.org/html/rfc7469*, 2015.

[ETS] TS ETSI. 123 040 3gpp ts 23.040 version 4.2. 0.

[ETS95] EN ETSI. 300 356-1" integrated services digital network (isdn). *Signalling System*, 1995.

[HD14] Roee Hay and Avi Dayan. Android keystore stack buffer overflow, 2014.

[Inc] Google Inc. IPv6 statistics. URL: `https://www.google.com/intl/en/ipv6/statistics.html`. Online; accessed: 2017-03-10.

[Kal93] Burton Kaliski. Privacy enhancement for internet electronic mail: Part 4: Notary, co-issuer, crl-storing and crl-retrieving services. In *RFC 1424*. RSA Laboratories, 1993.

[KL93] S Kent and J Linn. Rfc 1422: Privacy enhancement for internet electronic mail: Part ii: Certificate-based key management, february 1993. *Internet Engineering Task Force. URL: http://tools.ietf.org/html/rfc1422*, 1993.

[KS06] Stephen Kent and Karen Seo. Rfc 4301: Security architecture for the internet protocol. 2005. *Internet Engineering Task Force. URL: http://tools.ietf.org/html/rfc4301*, 2006.

[LY06] C Lonvick and T Ylönen. The secure shell (ssh) authentication protocol. *Internet Engineering Task Force. URL: http://tools.ietf.org/html/rfc4252*, 2006.

[P+81] Jon Postel et al. Rfc 791: Internet protocol. *Internet Engineering Task Force. URL: http://tools.ietf.org/html/rfc791*, 1981.

[Pos80] Jon Postel. Rfc 768: User datagram protocol, august 1980. *Internet Engineering Task Force. URL: http://tools.ietf.org/html/rfc768*, 1980.

[Pos03] Jon Postel. Rfc 793: Transmission control protocol, september 1981. *Internet Engineering Task Force. URL: http://tools.ietf.org/html/rfc793*, 88, 2003.

[Res00] Eric Rescorla. Rfc 2818: Http over tls. *Internet Engineering Task Force. URL: http://tools.ietf.org/html/rfc2818*, 2000.

[SA11] Peter Saint-Andre. Extensible messaging and presence protocol (xmpp): Core. *Internet Engineering Task Force. URL: http://tools.ietf.org/html/rfc6120*, 2011.

[SAH11] Peter Saint-Andre and Jeff Hodges. RFC 6125: Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS). URL: `https://tools.ietf.org/html/rfc6125`, March 2011.

[SBKH06] Steve Sheng, Levi Broderick, Colleen Alison Koranda, and Jeremy J Hyland. Why johnny still can't encrypt: evaluating the usability of email encryption software. In *Symposium On Usable Privacy and Security*, pages 3–4, 2006.

[SCH+02] Eve Schooler, Gonzalo Camarillo, Mark Handley, Jon Peterson, Jonathan Rosenberg, Alan Johnston, Henning Schulzrinne, and Robert Sparks. Sip: Session initiation protocol. *Internet Engineering Task Force. URL: http://tools.ietf.org/html/rfc6120*, 2002.

[Sha03] David Shaw. The openpgp http keyserver protocol (hkp). *Internet Engineering Task Force. URL: http://tools.ietf.org/html/draft-shaw-openpgp-hkp-00*, 2003.

[Sta] StatCounter. Device type and operating system market share worldwide. `http://gs.statcounter.com/os-market-share`. Online; accessed: 2017-04-12.

[Wel17] Bill Welch. Exploiting the weaknesses of ss7. *Network Security*, 2017(1):17–19, 2017.

[Zyr08] J Zyren. Long term evolution protocol overview. *White Paper, Freescale Semiconductor*, 2008.