Abstract

Preface

The work behind this project report was carried out during the spring semester in 2011 at the Norwegian University of Science and Technology (NTNU), Department of Telematics (ITEM).

Eirik Haver, Eivind Melvold and Pål Ruud

Contents

List of Figures

List of Tables

Listings

Acronyms

ACL Access Control List

AES Advanced Encryption Standard

CA Certification authority

DSA Digital Signature Algorithm

DSS Digital Signature Scheme

FAQ Frequently Asked Questions

IaaS Infrastructure as a Service

LAFS Least Authority File System

MITM Man-in-the-middle

NIST National Institute of Standards and Technology

PBKDF2 Password-Based Key Derivation Function version 2

PaaS Platform as a Service

PGP Pretty Good Privacy

PKI Public Key Infrastructure

RSA Rivest, Shamir and Adleman

SaaS Software as a Service

SHA Secure Hash Algorithm

SSL Secure Socket Layer

TLS Transport Layer Security

VM Virtual Machine

1

Introduction

1.1 Method

1.2 Outline

The work is presented as per the following chapters:

Chapter 2 provides background knowledge of the technologies and software used.

2

Background

2.1 Security Services

This section briefly explains certain security services used in this thesis. A security service is any processing or communication service that enhances the security of the data processing systems and the information transfers of any organization[2, p. 12].

Confidentiality is the art of keeping a message secret from unauthorized parties[2, p. 18]. This can typically be done by either preventing other parties access to the message at all, or making the contents unreadable for instance by the use of encryption.

Integrity in a security perspective deals with detecting, preventing or recovering a message from being changes by an unauthorized party [2].

Authentication is the act for a user, service or similar to prove that he is what he claims to be[2].

2.1.1 Nonrepudiation

prevents either sender or receiver of a message from denying a transmitted message, in other words one party can prove the other parties involvement[2].

2.2 Cryptographic Primitives

This section explains the low level security primitives used in this thesis.

2.2.1 Encryption

Encryption is the process of transforming some information into an unreadable form. Encryption is primarily used to enforce Confidentially, but can also be used for other purposes such as authentication. In a very basic form an encryption scheme consist of an algorithm, the cipher, a key and a message, the plaintext, that is all used to create an encrypted message, a ciphertext. If a good cipher is used, knowledge of the cipher, plaintext and ciphertext should not be enough to obtain the key.

Block-cipher and Stream-cipher are classifications on how a cipher treats data[2, p. 32]. With a block-cipher data will be encrypted in blocks of specific sizes. If the data length is not a multiple of the block size, the data will be padded. A stream cipher on the other hand will encrypt the message co.

Symmetric encryption is an encryption scheme where the same key is used for both encryption and decryption[2, p. 32]. Advanced Encryption Standard (AES) is a block cipher and is the current standard for symmetric encryption. AES works on a block of 128-bit and support keys of 128, 192 and 256-bit.

The mode of operation used for a symmetric encryption scheme enables subsequent safe use of the same key.

Asymmetric encryption is an encryption scheme where a different key is used for encryption than decryption[2, p. 259]. An asymmetric encryption scheme is often called a public-key encryption scheme, where one key is defined as private and the other as public. The public key is shared to allow other parties to encrypt messages for the owner of the private key. The downside of asymmetric encryption compared to symmetric is that it requires a larger key and has a larger computational overhead to obtain the same level of confidentiality. The probably best known asymmetric cipher is Rivest, Shamir and Adleman (RSA).

2.2.2 Cryptographic hash functions

A cryptographic hash function is a deterministic mathematical procedure which takes an arbitrary block of data and outputs a fixed-size bit string. The output is referred to as the hash value, message digest or simply digest. Another property of a cryptographic hash function is that the smallest change in the input data (e.g. one bit) should completely change the output of the hash function. In other words it should be infeasible to find the reverse of a cryptographic hash function [2, p. 335]. It should also be infeasible to find two blocks of data which produce the same hash value (a *collision*).

The standard for cryptographic hash functions today are Secure Hash Algorithm (SHA)-1 and the SHA-2 family.

2.3 Applications of cryptographic primitives

2.3.1 Digital Signatures

A digital signature is the digital equivalent of a normal signature, it verifies that an entity approves with or has written a message, the date the signature was made and it should be verifiable by a third party [2, p. 379]. It should logically not be possible or at least unfeasible to fake a digital signature. It is possible to create digital signatures with RSA there is also a standard for digital signatures called Digital Signature Scheme (DSS) which uses Digital Signature Algorithm (DSA) as the actual algorithm.

2.3.2 Digital Certificates and PKI

A digital certificate is the pairing of a digital signature and a public key[2]. By this scheme the services confidentiality, authentication and nonrepudiation can be achieved. Basically a person or other entity has a certificate with some clues about the identity in it, e.g. the e-mail, together with a public key. This certificate can then be signed using digital signatures to verify that some other entity trusts this certificate. In practise the entity which signs certificates is the Certification authority (CA) which all clients have the public key information for, and trusts. The CA will also contain information about which certificates has been revoked, i.e. should not be trusted in use. Such a scheme is usually referred to as a Public Key Infrastructure (PKI).

PGP

Pretty Good Privacy (PGP) is a scheme similar to PKI but with no CA that all users trusts[2]. Instead trust is made between users by somehow verifying their public key, for instance by meeting face to face. A user can then sign another users key, set a trust level for the user and publish this information to a keyserver. Another user can then calculate a trust to an unknown person based on the trust set by peoples that he trusts.

2.3.3 SSL/TLS

Transport Layer Security (TLS) and its predecessor Secure Socket Layer (SSL) are techniques for obtaining confidentiality, integrity for transfer of files over a network[2]. It does so by a combination of different algorithms and primitives, but a digital certificate is required for authentication.

2.3.4 PBKDF2

Password-Based Key Derivation Function version 2 (PBKDF2) is a key derivation function to create an encryption key based on a password. The point of this is that a password is often something that should be memorable to a person, but what is memorable to a person might be a to short phrase to withstand a brute force attack. What PBKDF2 does is make the process of deriving the key from the password an expensive process in terms of computational power, to make it more resistant to brute force attacks.

2.4 Security Attacks

This section briefly list security attacks relevant to this thesis, as defined by Stallings.

Active and passive attacks are classifications of security attacks, where a passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation.

Traffic analysis is the art of capturing communication sent between two parties. This information might contain secrets or might for instance leak enough information about an encryption key to make it breakable.

Masquerade is an active attack where the attacker pretends to be one of the legit parties.

Replay is an active attack where the attacker capture some data in a communication session and subsequently retransmit that information.

Modification of messages is an active attack where the attacker alters some of the contents of a message sent between two communicating parties.

Denial of Service is an active attack where the attacker seeks to make resources unavailable for legit users, i.e. by overloading an application by sending it lots of traffic.

Man-in-the-middle is an attack where an attacker intercepts messages between the communicating parties and then either relay or substitute the intercepted message.

2.4.1 Attacks on cryptographic primitives

Even though cryptographic primitives are designed to be secure, they might have both flaws and be used in an incorrect fashion.

Cryptanalysis attack is an attempt to deduce a specific plaintext or to deduce the key being used in a ciphertext.

Brute-force attack is an attack where you try to obtain a secret by testing the algorithm with up to all possible inputs. The secret might be an encryption key or the data fed into a cryptographic hash function.

2.5 Cloud Computing

In a draft[1] National Institute of Standards and Technology (NIST) defines cloud computing as:

> Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

2.5.1 Service Models

NIST also defines three service models which deals with what kind of service the consumer is able to rent from a provider.

Software as a Service (SaaS) is the capability for a consumer to run the provider's application running on cloud infrastructure, using a thin-client, browser or similar. Gmail[1] can be seen as an example of this.

Platform as a Service (PaaS) is the capability for a consumer to deploy software onto the cloud, but without actually controlling the underlying platform, operating system etc.

Infrastructure as a Service (IaaS) is the capability provided to the consumer to provision processing, storage, networks and other fundamental computing resources where he can run arbitrary software, including operating systems and applications. An example is hiring a Virtual Machine (VM).

2.5.2 Deployment Models

The NIST draft also lists several Deployment Models which deals with how the cloud is organized in terms of where it is hosted and who has access to it.

Private Cloud is a cloud infrastructure operated solely for an organization. Which party manages the cloud and where it is located is not given.

Community Cloud is a cloud infrastructure is shared by several organizations to serve a common concern. Where it is located and who manages it is not given.

Public Cloud is a cloud infrastructure where basically everyone or at least a large group can have access, and is owned by a external provider of cloud services.

Hybrid Cloud is a cloud infrastructure composed of two or more clouds of any other model.

[1] `http://www.gmail.com`

2.5.3 Security considerations

There are some considerations when using cloud services from an external provider as opposed to self controlled hardware, software and plattforms. Most notably is that you loose the control of selecting the people which will have physical and digital access to the infrastructure. In essence this means that the provider can read every data sent to and from the cloud as well as the data saved in the cloud.

Another risk is that information might be leaked to other users of the same cloud. For instance it might be able possible for a VM to leak information to other VMs on the same host.

2.6 Existing Solutions

There are a number of existing storage solutions for storing data in the cloud, with more or less of the functionality required to fulfill the problem description for this thesis. The section highlights some of them.

2.6.1 Dropbox

Dropbox[2] is a commercial application for storing data in the cloud, more specific using Amazons S3 storage service. It claims that files are stored encrypted with AES-256 which can only be decrypted with the users username and password, and that Dropbox employees are not able to access the files of the user[3]. However Dropbox also has a Forgot password feature which means that Dropbox can read the users files if they really want to. Their Frequently Asked Questions (FAQ) does however say that some people have been successfull in putting truecrypt containers in Dropbox which effectivly makes dropbox secure[4].

2.6.2 Tahoe-LAFS

Tahoe-Least Authority File System (LAFS)[5] is an open source cloud storage file system which does fulfill the requirements set by our problem description in regards to security. In Tahoe-LAFS files are exclusivly encrypted client-side, before beeing uploaded into the cloud. Tahoe-LAFS also uses erasure-coding to obtain redundancy across multiple storage servers.

[2] `http://www.dropbox.com`
[3] `http://www.dropbox.com/help/27`
[4] `http://www.dropbox.com/help/179`
[5] http://www.tahoe-lafs.org

3

Architectural Solution

The architectural solution of a secure cloud file sharing system has to convince its users that the functions indeed are secure, and that the concepts are easy to understand and accept.

3.1 Introduction

The architecture has to support various user functionality. Figure 3.1 exhibits Alice uploading files to the cloud, and thereafter transferring the necessary information to Bob so that he also can gain access to the files.

In the following sections, we will describe how the file storage is organized, and take a closer look at how the different functionality are solved.

3.2 File Storage

The solution proposed in this thesis, is that only a simple key-value store is needed on the server side. This may be extended with an Access Control List (ACL) layer to support user access and other features.

Two types of files exist: immutable and mutable files. The mutable files are used as directories, in the sense that they contain the information needed to access other directories or files in the form of capabilities. A capability is a short alphanumeric string containing all information needed to find, get, read and write a file or folder. This includes an identifier and cryptographic keys.

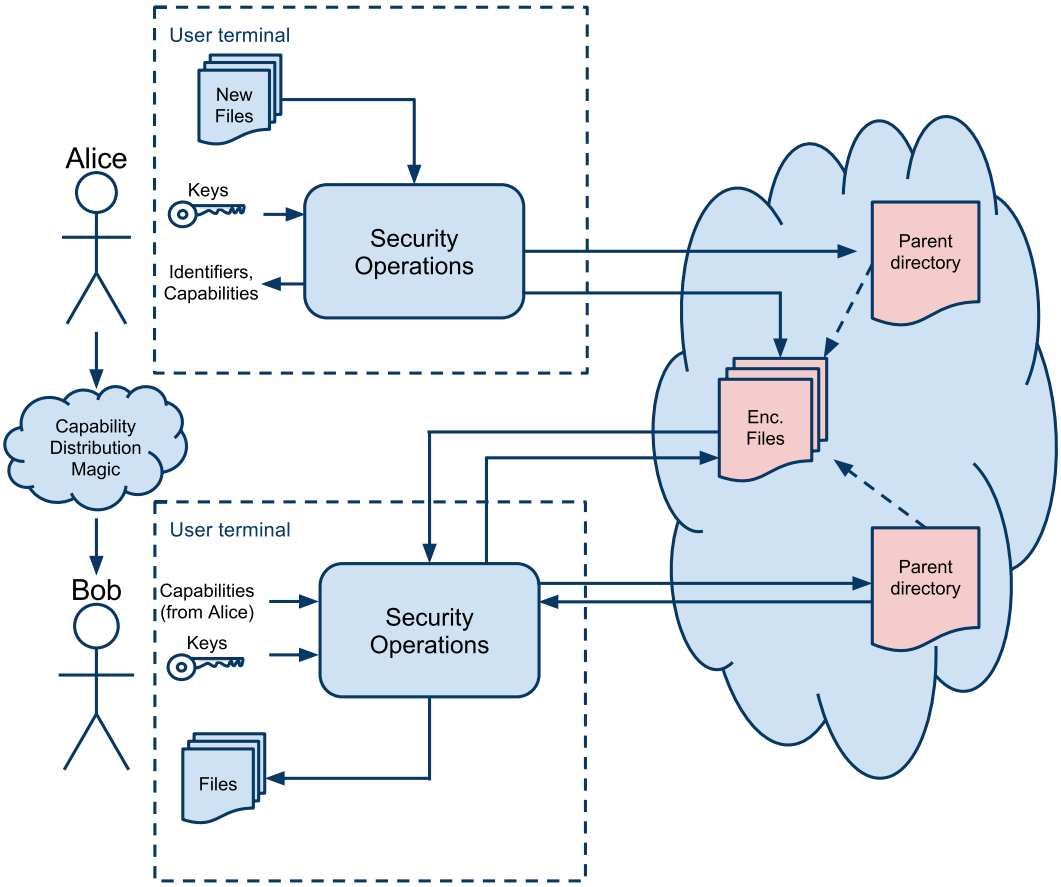

Figure 3.1: Overview of user functionality

In principle, the only operations the key-value store need to support are **PUT**, **GET** and **UPDATE**. The latter one is required to make changes to the "directory" files.

3.3 User scenarios

The various user scenarios the software has to support, provides a logic way to describe the external properties of the system. The fundamental operations are *downloading*, *uploading* and *sharing* of files.

3.3.1 Download file

When a user wishes to download a file or directory, all that is needed is the password to unlock the local keyring on the user terminal, as depicted in

Figure 3.2. The client sends a GET request with the identifier in question, and the server responds with the encrypted folder. This contains the capabilities needed to locate and decrypt the underlying files and folders. After decrypting the contents, the client again queries the server with the identifier of the wanted file, and there after decrypts it, before displaying it to the user.

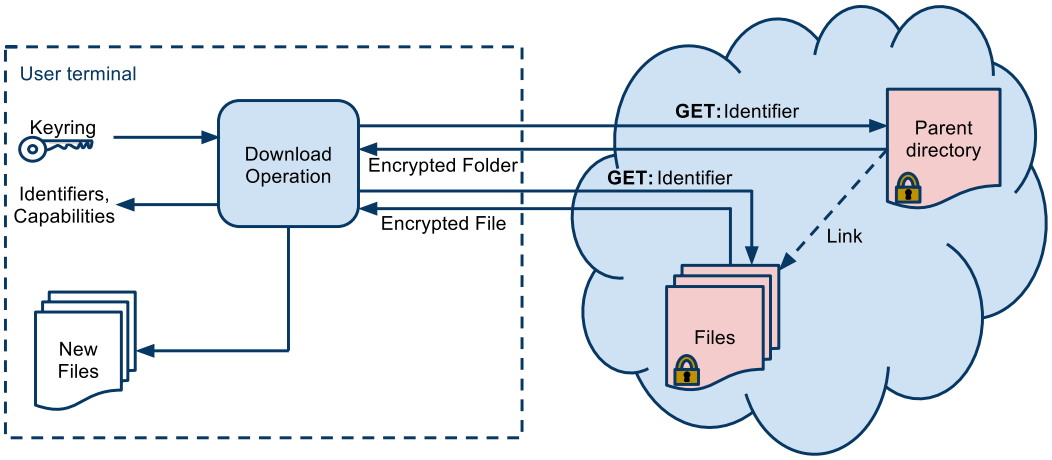

Figure 3.2: Scenario: Downloading of files

3.3.2 Upload Files

Figure 3.3 shows the process of uploading a new file. The only information the server in the cloud receives, are identifiers and encrypted containers. The user is also given the opportunity to transfer the corresponding identifiers and capabilities to users.

Before uploading, the client has to download and decrypt the directory the files are to be placed in. This process was described in the previous section. The cryptographic details of the Upload Operation can be found in Section 4.2.

3.3.3 Share files

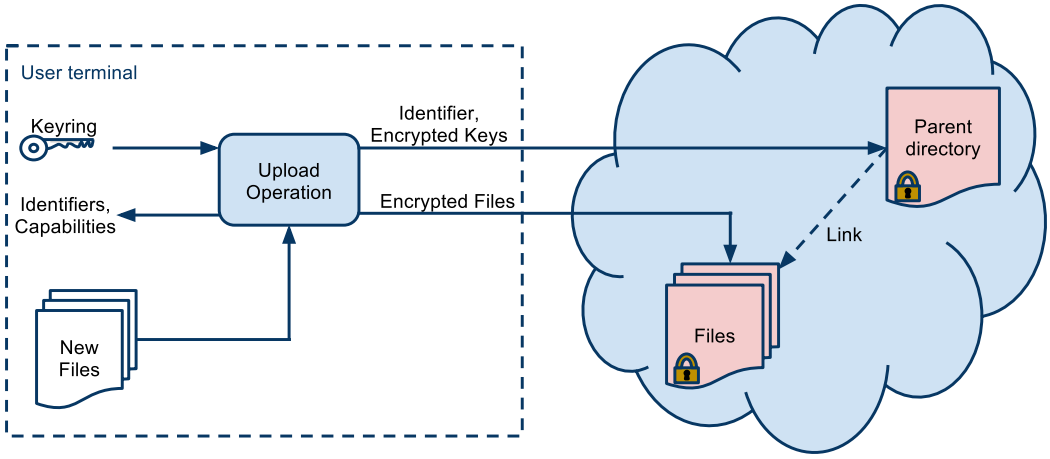

Figure 3.3: Scenario: Uploading of files

4

Cryptographic Solutions

This chapter will elaborate on the cryptographic solutions applied to the architectural scheme in Chapter 3. We will take a closer look at how confidentiality, integrity, authentication and access control can be integrated into the proposed architecture.

A fundamental scheme for key distribution is needed to realize the desired security features, hence an appropriate solution for key distribution will also be given.

In the first section, we will give a brief introduction explaining the basic security scheme used in the application. We will further look at each file operation exhibited in Chapter 3 and describe our corresponding cryptographic solution. The following section will give a detailed overview of the cryptographic keys used in the mentioned file operations. The chapter will end with an explanation of the chosen scheme for key distribution.

4.1 Basic Security

The basic security concept of our application is to keep a users remote storage confidential to a third-party storage provider. To solve this, we will create an application that encrypt files locally at the users terminal before they are uploaded to the third-party storage provider. To further access files, the user will have to decrypt files locally when downloading them. To enable this simple encryption scheme, the user is required to possess at least one cryptographic

key.

By initially knowing that files are encrypted on a remote server and that the local user possess one or more cryptographic keys, we can continue with a more comprehensive description of the complete cryptographic solution. The details of the complete solution will be explained in the manner of 3 different file operations, namely upload, download and share file.

4.2 Upload File

The process of uploading a file onto a remote server was illustrated in Figure **??**. This section will provide a more detailed description of the process and specify the functionality of the Upload operation at the user terminal.

When uploading a file to the remote file structure, described in Section 3.2, two operations are needed. First of all, it is important to add the desired file's meta data into the remote parent directory. This meta data will serve as a link to the uploaded file. The second operation is to upload the desired file.

These two operations can be broken into the following five steps:

1. Download and decrypt parent directory to target file
2. Generate file keys and capabilities
3. Write file meta data to parent directory
4. Encrypt and upload directory
5. Encrypt and upload file

Each step is described below.

4.2.1 Download and Decrypt Directory

To insert the file's meta data into the remote directory, it is necessary to download and decrypt the chosen directory. We will combine the download and decrypt procedures into the term "open directory". The procedure of opening a directory is illustrated in Figure **??** and described as follows.

To download and open the desired parent directory, the user must be in possession of a read key specific to the directory. The read key is created from a hash value of the directory's write key. How keys are obtained by the user will be described in Section **??**. The distribution of keys will be exhibited in Section **??**.

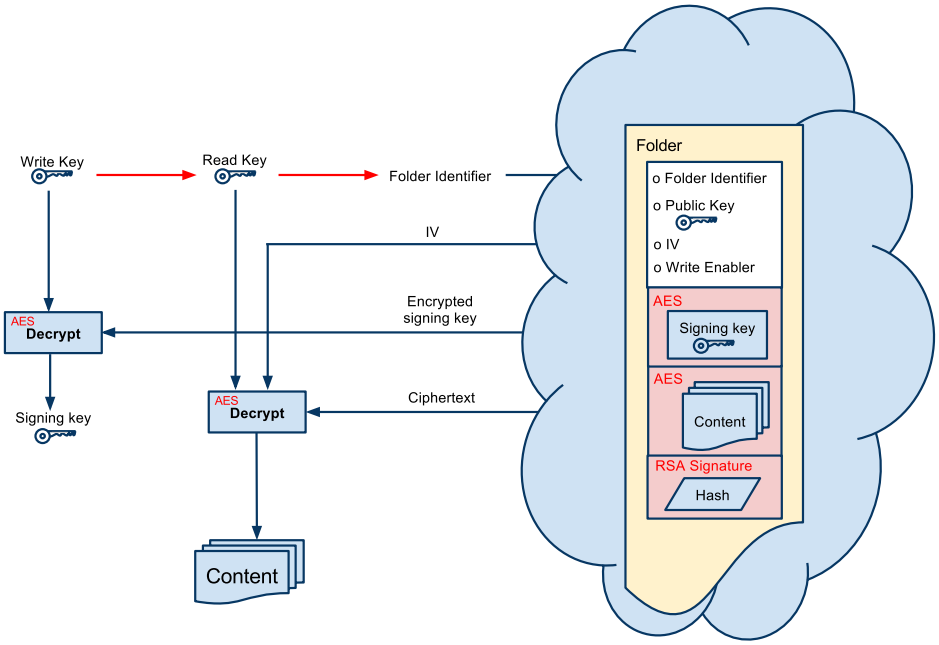

Figure 4.1: Opening a remote directory.

The directory specific read key is used for two purposes. The first purpose is to identify and localize the corresponding directory on the remote server. This is done by hashing the read key to create a "Folder Identifier" value. This value is used to localize which directory the user should download. The structure of a remote directory is depicted in Figure **??**. The second purpose of the read key is to decrypt the encrypted content in the directory. An Initialization Value (IV) is used together with the read key to carry out the decryption procedure. The IV is easily fetched from the localized directory prior to decryption.

After decryption, the user has to obtain a signing key that is specific to the directory. The signing key is an RSA private key corresponding to the directory's public key. It is needed by the user to later modify the directory. A modified directory must contain a valid signature to proof that it has been modified by a user with write permissions. The signature consists of an encrypted hash value of the directory content. The hash value is encrypted with the signing key.

The signing key is located within a directory and is encrypted with AES in CBC mode. The directory's write key is used to encrypt the signing key.

The user simply downloads the encrypted signing key and decrypts it with the write key. This is illustrated in Figure **??**.

4.2.2 Generating Keys and Capabilities

4.2.3 Write File to Folder

4.2.4 Encrypt and Upload Folder

4.2.5 Encrypt and Upload File

4.3 Download File

4.4 Share File

4.5 Key Structure

-Describe the different keys -Describe where the keys come from

4.6 Key Distribution

-Confidentiality: AES-CBC, RSA
-Integrity: SHA-256
-Authentication: RSA PrK signature
-Key hierarchy
-Key distribution

5

Implementation

6

Results

7

Discussion

- Cascading deletes? Loops.

8

Conclusion and Future Work

Bibliography

[1] P. Mell and T. Grance. The nist definition of cloud computing (draft). Technical report, NIST, 2011. From `http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf`.

[2] William Stallings. *Cryptography and Network Security – Principles and Practices*. Pearson Education Inc., fourth edition, 2006.

Appendices