Google Cloud: Secure Communication

Vojtěch Horký Petr Tůma

2010 – 2022

Contents

1 Technology Overview **1**

2 Assignment Part I **2**

3 Authorization **3**

4 Google Cloud Platform Services **4**

5 Assignment Part II **5**

1 Technology Overview

RSA Refresher

Public Key Cryptography
A key pair where data encrypted with one key (private or public) can be decrypted with the other one (public or private).
- Public key available, private key kept secret
- Encrypting with public key, signing with private key

$x^{(p-1)(q-1)} = 1 \ (modulo\ pq)$ ... for p, q prime and x not commensurable with pq pick p, q have $n = pq$ and $\phi = (p-1)(q-1)$ pick e, d such that $ed = 1 \ (modulo\ \phi)$ then $(m^e)^d = m^{1+k(p-1)(q-1)} = m \cdot m^{k(p-1)(q-1)} = m \ (all\ modulo\ n)$

... Martin Ouwehand: The (simple) Mathematics of RSA

DH Refresher

Shared Secret Agreement
A process through which parties can agree on a shared secret without actually transmitting the shared secret itself.

have p and g where g is a generator of multiplicative integer group modulo p

Alice: pick a and publish $g^a \ (modulo\ p)$ Bob: pick b and publish $g^b \ (modulo\ p)$
then $(g^a)^b = (g^b)^a$ is a shared secret

TLS Technology Overview

Goals

Provide privacy and integrity guarantees in network communication.

Features

- Ciper suite negotiation
  - Key exchange (~~RSA~~, DHE, PSK ...)
  - Encryption (AES GCM, ~~AES CCM~~, ~~AES CBC~~ ...)
  - Message authentication (~~MD5~~, ~~SHA1~~, SHA256 ...)
- Secure session key exchange
- Server authentication
- Data encryption
- Data integrity

... TLS 1.3 RFC 8446

TLS 1.2 RSA Handshake Sketch

```
[CLT] Hello, I support these cipher suites,
      and here is my CLIENT RANDOM number

[SRV] Hello, I have picked cipher suite AES256-SHA256,
      here is my SIGNED SERVER CERTIFICATE
      and here is my SERVER RANDOM number

[CLT] Here is a random PRE MASTER SECRET encrypted with your RSA key

MASTER SECRET = function (PRE MASTER SECRET, CLIENT RANDOM, SERVER RANDOM)
various session keys = function (MASTER SECRET)

[CLT] I am finished and here is encrypted hash of exchanged messages

[SRV] I am finished and here is encrypted hash of exchanged messages
```

TLS 1.3 ECDH Handshake Sketch

```
[CLT] Hello, I support these cipher suites,
      I expect you to use these DH handshake parameters,
      here is my CLIENT DH KEY SHARE
      and my CLIENT RANDOM number

PRE MASTER SECRET = function (CLIENT DH KEY SHARE, SERVER DH KEY SHARE)
MASTER SECRET = function (PRE MASTER SECRET, CLIENT RANDOM, SERVER RANDOM)
various session keys = function (MASTER SECRET)

[SRV] Hello, I have picked this cipher suite,
      here is my SERVER DH KEY SHARE
      and my SERVER RANDOM number,
      and I am finished and here is encrypted hash of exchanged messages
      and my encrypted SIGNED SERVER CERTIFICATE

[CLT] I am finished and here is encrypted hash of exchanged messages
```

2 Assignment Part I

Assignment

Server

Implement a server that will provide information on current time.

- The server should accept a spec of what fields to return.
- Fields should be standard YYYY-MM-DD HH:MM:SS.

Client

Implement a client that will query server time:

- Pick a random combination of fields.
- Query information on current time.
- Print the time.

Security

The connection between the client and the server should be encrypted.

Python Secure Connection Basics

Server

```
key_data = open ('server.key', 'rb').read ()
crt_data = open ('server.crt', 'rb').read ()
credentials = grpc.ssl_server_credentials ([( key_data, crt_data )])

server = grpc.server (...)
server.add_secure_port (SERVER_ADDR, credentials)
```

Client

```
crt_data = open ('server.crt', 'rb').read ()
credentials = grpc.ssl_channel_credentials (root_certificates = crt_data)
channel = grpc.secure_channel (SERVER_ADDR, credentials)

stub = AnExampleServiceStub (channel)
```

Certificate Generation

Self Signed

Good for limited testing but nothing else !

```
> openssl req -newkey rsa -nodes -keyout server.key -x509 -out server.crt -days 666
> openssl x509 -in server.crt -text
> openssl rsa -in server.key -text
```

- Create both a key and a certificate
- Create RSA key with default size
- Do not encrypt the RSA key file
- Make the certificate self signed
- Make the certificate valid for 666 days

For Real Use

See `https://www.letsencrypt.org` ...

3 Authorization

OAuth Technology Overview

Goals

Standard protocol for granting third party applications limited access to HTTP accessible resources.

Features

- Considers multiple client types

    - Applications running in browser
    - Server hosted applications acting on own behalf
    - Server hosted applications acting on user behalf
- Heavily uses browser request redirection
- Requires (mostly) encrypted communication
- Authentication represented by (secret) access token

... OAuth 2.0 RFC 6749

Authorization Process Participants

Resource Owner
This is the end user who authorizes third party clients to access resources. The resource owner accesses the third party client through a browser.

Resource Server
This is the server that provides access to resources when shown authorization in the form of access token.

Third Party Client
This is the application that needs to access resources on behalf of resource owner.

Authorization Server
This is the server that can authenticate the resource owner and issues access tokens as directed by the resource owner.

Authorization Process Sketch

```
[OWN] Accesses an application link that needs authorization.

[APP] Responds with REDIRECT sending the browser to authorization server.
      The link includes CLIENT ID and SCOPE and arbitrary STATE.

[OWN] The browser follows the link to the authorization server.

[AUT] The server authenticates the user behind the browser.
      The user is then asked to grant authorization for SCOPE.
      The server concludes with REDIRECT back to the application.
      The link includes AUTHORIZATION CODE and associated application STATE.

[OWN] The browser follows the link to the application.

[APP] The application gets the AUTHORIZATION CODE from the link.
      The application asks the authorization server to convert
      the AUTHORIZATION CODE into an ACCESS TOKEN.

[AUT] The server generates the ACCESS TOKEN as requested.

[APP] The application accesses the resource server
      with the ACCESS TOKEN included in request header.
```

4 Google Cloud Platform Services

Google Cloud Platform Technology Overview

Goals
Computing platform build on Google infrastructure resources and services.

Features

- Tons of services
  - Compute services (IaaS and PaaS and FaaS)
  - Storage services (SQL, tables, documents, raw block storage)
  - Networking (private networks, load balancing, content delivery)
  - Big data processing
  - Machine learning
  - Management
- Accessible through public interfaces
- Libraries for multiple languages

... http://cloud.google.com

Installation

Browser

- Register for free trial at http://cloud.google.com
- Log in to console at http://console.cloud.google.com
- Create a new project
- Enable required libraries
- Create and download a service account key

Shell

```
> export GOOGLE_APPLICATION_CREDENTIALS=/path/to/service-account-key.json
```

Cloud Speech API

```
from google.cloud import speech as google_cloud_speech
from google.cloud.speech import enums as google_cloud_speech_enums
from google.cloud.speech import types as google_cloud_speech_types

client = google_cloud_speech.SpeechClient ()

content = read_data_from_file (...)
audio = google_cloud_speech_types.RecognitionAudio (content = content)
config = google_cloud_speech_types.RecognitionConfig (language_code = 'en-US')

result = client.recognize (config, audio)
```

... http://cloud.google.com/speech/docs

Cloud Translate API

```
from google.cloud import translate_v2 as google_cloud_translate

client = google_cloud_translate.Client ()

# Get a list of all supported languages.
languages = client.get_languages ()

# Translate a sentence.
result = client.translate ('some_text', target_language = 'en')
```

... http://cloud.google.com/translate/docs

5 Assignment Part II

Assignment

Goal

Create a client that translates input speech.

- An audio file with speech in English on input
- A text with speech translated into Czech on output

… or some other interesting service combination.

Implementation

Use the client libraries rather than generated stub code.

Submission

GitLab

Use your personal GitLab repository under `https://gitlab.mff.cuni.cz/teaching/nswi163/2022`.

Requirements

- Use the assignment subdirectory.
- Write brief report in `SOLUTION.md`.
- Include build scripts with instructions.
- Do not commit binaries or temporary build artifacts.
- Tag your solution with `task-05` and push the tag.

Credentials

- Do commit your self signed server certificate and key for Part I.
- Do not commit your Google Cloud Access Token for Part II.