Einrichtung einer verschlüsselten und authentifizierten Verbindung auf der Basis lagebezogener Messdaten

Raspberry Pi-Projekt (WS 2015/2016)

Lehrstuhl für Digitale Kommunikationssysteme
Ruhr-Universität Bochum

Jan Holthuis
Matr.-Nr: 108 009 215 809
jan.holthuis@ruhr-uni-bochum.de

Daniel Peeren
Matr.-Nr: 108 012 210 266
daniel.peeren@ruhr-uni-bochum.de

17. Februar 2016

Zusammenfassung

In diesem Projekt geht es um den Aufbau einer verschlüsselten und authentifizierten Verbindung zweier Kommunikationspartner mittels lagebezogener Daten.

Inhaltsverzeichnis

1 Einleitung

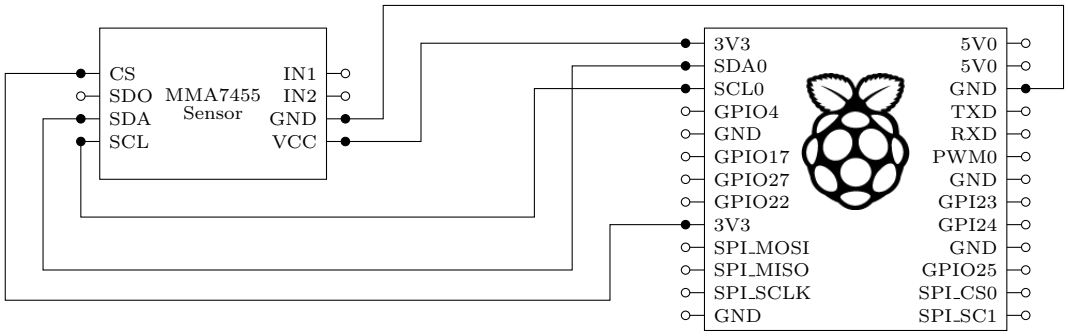
In diesem Projekt geht es um den Aufbau einer verschlüsselten Verbindung und die gegenseitige Authentifizierung zweier Kommunikationspartner anhand von lagebezogener Messwerte. Dies geschieht mit Hilfe von Raspberry Pis und Lagesensoren. Die gegenseitige Authentifizierung (*mutual authentication*) sowie die Schlüsselvereinbarung (*key agreement*) erfolgt mittels eines Protokolls auf Basis eines Challenge-Response-Mechanismus.
Als gemeinsames Geheimnis dienen hierbei die Messwerte des Lagesensors (Tupel der Form (x, y, z), die die Lage im 3-dimensionalen Raum beschreiben).

2 Aufbau

In diesem Projekt werden zwei (2) Raspberry Pis mit je einem Lagesensor verwendet. Diese werden via Ethernet bedient und bauen auch eine gemeinsame Verbindung darüber auf.

2.1 Anschluss der Lagesensoren

Als Lagesensoren wurden *SainSmart MMA7455 Accelerometer* verwendet. Diese wurden jeweils wie folgt an die GPIO-Pins der Raspberry Pis angeschlossen:



2.2 Bibliotheken

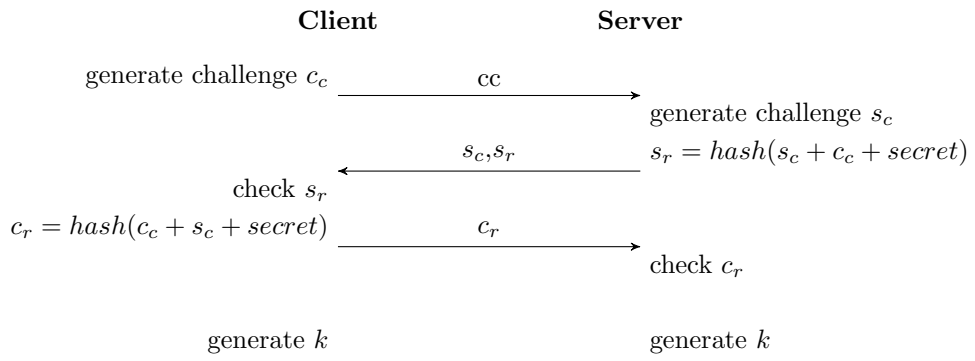
Das Programm verwendet folgende Python-Libraries:

pycrypto Bibliothek mit kryptographischen Primitiven, z.B. Hashfunktionen (z.B. SHA1 und SHA256), Chiffren (z.B. AES und DES) oder Key Derivation Functions (z.B. PBKDF2).

smbus Python-Anbindungen für Linux-SMBus-Zugriff durch i2c-dev.

2.3 Protokoll

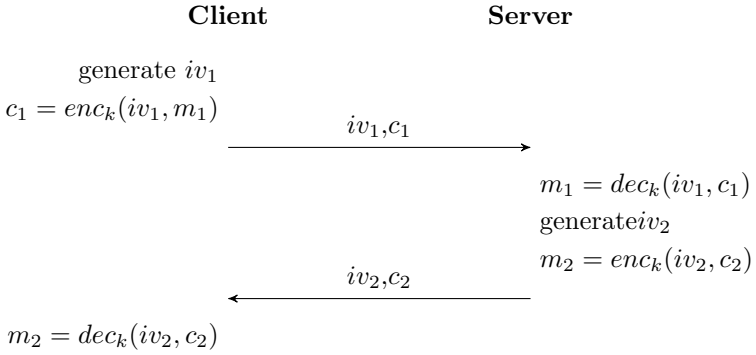
Das Programm ist mit folgendem, auf einem Challenge-Response-Mechanismus basierenden Protokoll ausgestattet:



Nach diesem Schritt sind beide Teilnehmer jeweils

- authentifiziert (*mutual authentication*)
- im Besitz eines gemeinsamen Schlüssels k (*key agreement*)

Anschließend können beide Teilnehmer mithilfe des Schlüssels k verschlüsselt kommunizieren:



3 Bedienung

3.1 Vorbereitungen

1. Programm (und Abhängigkeiten) auf den Raspberry Pis installieren
2. Raspberry Pis in ein Netzwerk einbinden (via Ethernet-Kabel oder USB-WLAN-Adapter)
3. Lagesensoren an die GPIO-Pins der Raspberry Pis anschließen

3.2 Befehle

Das Programm kann mit folgenden Befehlen bedient werden:

Befehl	**Abkürzung**	**Funktion**
`--auth`	`-a`	Festlegen, ob das Protokoll zur Authentifizierung verwendet werden soll.
`--bus <id>`	`-b <id>`	SMBus-ID setzen. default = 1
`--client`	`-c`	Das Programm als Client ausführen.
`--demo`	`-x`	Einen Demo Sensor anstatt eines physischen Sensors verwenden.
`--host <ip>`	`-H <ip>`	IP-Adresse des Hosts setzen.
`--no-quantization`	`-n`	Quantisierung deaktivieren.
`--offset <x y z>`	`-o <x y z>`	Sensoroffsets setzen, um Ungenauigkeiten der Sensoren auszugleichen.
`--port <port>`	`-P <port>`	Port des Servers setzen.
`--server`	`-s`	Das Programm als Server ausführen.
`--test-sensor`	`-t`	Test der Sensoren via Ausgabe der aktuellen Sensordaten.
`--verbose`	`-v`	Zeigt Debug-Informationen an.

3.3 Ausführung

Testweise führen wir das Programm aus, wobei der Server hier auf dem Raspberry Pi mit der IP-Adresse 192.168.178.11 lauscht. Der Client verbindet sich von IP-Adresse 192.168.178.22 aus und versucht, eine authentifizierte und verschlüsselte Verbindung aufzubauen.
Zur Veranschaulichung schlägt hier der erste Authentifizierungsversuch aufgrund von unterschiedlichen Messdaten fehl, während der zweite Versuch gelingt.

Client

```
pi@192.168.178.22 ~ $ ./run.py -ac -H 192.168.178.11 -vvv
Demo sensor: No (SMBus ID 1)
Mode: Client
Host: 192.168.178.11
Port: 9876
Auth: Yes
INFO:accelauth.client:Connecting to URL http://192.168.178.11:9876 ...
INFO:accelauth.client:Trying to authenticate...
DEBUG:accelauth.client:secret = (39, 5, 10)
INFO:accelauth.client:Generating client challenge...
DEBUG:accelauth.client:cc = '\x19e\xc6%b&\x8b\x8eV\x11\x1c\xf6\xec\xc6\xd2\xf3'
INFO:accelauth.client:Sending client challenge to server...
INFO:accelauth.client:Received server challenge/response...
DEBUG:accelauth.client:sc = '\x17\xd7o\xca\xa1\xeb\x95\x83\x08\xe7\xc5\x17\xaf#\
    ↪xd0\x9a'
INFO:accelauth.client:Checking if server response is valid...
DEBUG:accelauth.client:sr1 = 'd50643e86f9a72da388f0a8566dbd460272d6014'
DEBUG:accelauth.client:sr2 = 'ae7d7df4f7505fa4820643aa0c861900e0137281' (expected)
WARNING:accelauth.client:Server response is invalid for secret (39, 5, 10)!
DEBUG:accelauth.client:cr = 'a4871bbf1e4dda07efc2bd4b5b04f256f008e519'
INFO:accelauth.client:Sending client response....
INFO:accelauth.client:Trying to authenticate...
DEBUG:accelauth.client:secret = (38, 35, 19)
INFO:accelauth.client:Generating client challenge...
DEBUG:accelauth.client:cc = "\xd4\x112\x95,7\xc0\xec\x02\x10@\x10#'\xdc\xbf"
INFO:accelauth.client:Sending client challenge to server...
INFO:accelauth.client:Received server challenge/response...
DEBUG:accelauth.client:sc = '\x9d4\x82\x9aW\x7fdI\x98\x81\x7fS\x04G\xed4'
INFO:accelauth.client:Checking if server response is valid...
DEBUG:accelauth.client:sr1 = '646846269339b6f744f4e1dc7424f8efdd6c359e'
DEBUG:accelauth.client:sr2 = '646846269339b6f744f4e1dc7424f8efdd6c359e' (expected)
INFO:accelauth.client:Server response is valid for secret (38, 35, 19)!
DEBUG:accelauth.client:cr = 'e492aa8dfbb59b5b9c9318f7ae4d479a26158e88'
INFO:accelauth.client:Sending client response....
INFO:accelauth.client:Generating key...
INFO:accelauth.client:Key generated!
DEBUG:accelauth.client:key = "Ls\xd52\xdc\xae\xa2\x90?'(\xf4z\xa1R\x9eY4\x04\x88\
    ↪xb9\xbc/\x16\x89\x12\xfbH\x0f\x08\x9a\xed"
Type 'logout' to end conversation.
YOU: hello
DEBUG:accelauth.client:iv = '\x85\xd1s5Y\xd9-\x82\xd2\xac\x1bNs\xfd\xbf\xfb'
DEBUG:accelauth.client:ciphertext = '\xdcL\xbb,.\xa8\xf0\xcfv\xdc6(O\xa1bS'
INFO:accelauth.client:Sending message to server...
DEBUG:accelauth.client:s_iv = '\xfe\x0c\nSm$\xdb\x92\xf2\x19\x02/(\xbc\xff?'
DEBUG:accelauth.client:s_ciphertext = '\xb2=\xf5a\x99\xf46\xdfB\xe4\xa5\xc2\xaei\
    ↪xc7\xcaT\xdc_\xb1jM\xad\x05J\xc4%m.\\\t\xf6P\x1b7\x1c\x97)iz\x15\n\xf0\xcf
    ↪}\xf3ku'
SERVER: I received your message, thanks!
YOU: logout
DEBUG:accelauth.client:iv = '\x83\xed[Ot\x81\xd1\xbb\xb3\x0e\xbd$\xb9U\xfa\xab'
DEBUG:accelauth.client:ciphertext = '\x0c\xef\xcc\x8e\xbb\x8c\x90\xbf\x83+X\xbeQ\
    ↪xc5]J'
INFO:accelauth.client:Sending message to server...
DEBUG:accelauth.client:s_iv = 'n\x7f\xd0\xb6\x0b\xc5K\x8a\xfd\xe2\xea\x03J\x00\x8e
    ↪\xc4'
DEBUG:accelauth.client:s_ciphertext = "F\xc0\xa5r\xea\xbao\x18'k\x06\x03(\xca\x8e\
    ↪x8b"
SERVER: Goodbye!
INFO:accelauth.client:Client exited.
```

Server

```
pi@192.168.178.11 ~ $ ./run.py -as -H 0.0.0.0 -vvv
Demo sensor: No (SMBus ID 1)
Mode: Server
Host: 0.0.0.0
Port: 9876
Auth: Yes
INFO:accelauth.server:Started listening on 0.0.0.0:9876 ...
INFO:accelauth.server:New authentication request from '192.168.178.22'!
DEBUG:accelauth.server:secret = (29, 40, 7)
DEBUG:accelauth.server:cc = '\x19e\xc6%b&\x8b\x8eV\x11\x1c\xf6\xec\xc6\xd2\xf3'
INFO:accelauth.server:Generating server challenge...
DEBUG:accelauth.server:sc = '\x17\xd7o\xca\xa1\xeb\x95\x83\x08\xe7\xc5\x17\xaf#\
    ↪xd0\x9a'
INFO:accelauth.server:Generating server response...
DEBUG:accelauth.server:sr = 'd50643e86f9a72da388f0a8566dbd460272d6014'
INFO:accelauth.server:Sending server challenge/response to '192.168.178.22' and
    ↪wait...
INFO:accelauth.server:Continuing authentication with '192.168.178.22'!
INFO:accelauth.server:Checking if client response is valid...
DEBUG:accelauth.server:cr1 = 'a4871bbf1e4dda07efc2bd4b5b04f256f008e519'
DEBUG:accelauth.server:cr2 = 'c5a734ea4d8a8c0d02e8f61964d0345308684cde' (expected)
WARNING:accelauth.server:Client response is invalid for secret (29, 40, 7)!
INFO:accelauth.server:New authentication request from '192.168.178.22'!
DEBUG:accelauth.server:secret = (38, 35, 19)
DEBUG:accelauth.server:cc = "\xd4\x112\x95,7\xc0\xec\x02\x10@\x10#'\xdc\xbf"
INFO:accelauth.server:Generating server challenge...
DEBUG:accelauth.server:sc = '\x9d4\x82\x9aW\x7fdI\x98\x81\x7fS\x04G\xed4'
INFO:accelauth.server:Generating server response...
DEBUG:accelauth.server:sr = '646846269339b6f744f4e1dc7424f8efdd6c359e'
INFO:accelauth.server:Sending server challenge/response to '192.168.178.22' and
    ↪wait...
INFO:accelauth.server:Continuing authentication with '192.168.178.22'!
INFO:accelauth.server:Checking if client response is valid...
DEBUG:accelauth.server:cr1 = 'e492aa8dfbb59b5b9c9318f7ae4d479a26158e88'
DEBUG:accelauth.server:cr2 = 'e492aa8dfbb59b5b9c9318f7ae4d479a26158e88' (expected)
INFO:accelauth.server:Client response is valid for secret (38, 35, 19)!
INFO:accelauth.server:Generating key...
INFO:accelauth.server:Key generated!
DEBUG:accelauth.server:key = "Ls\xd52\xdc\xae\xa2\x90?'(\xf4z\xa1R\x9eY4\x04\x88\
    ↪xb9\xbc/\x16\x89\x12\xfbH\x0f\x08\x9a\xed"
DEBUG:accelauth.server:c_iv = '\x85\xd1s5Y\xd9-\x82\xd2\xac\x1bNs\xfd\xbf\xfb'
DEBUG:accelauth.server:c_ciphertext = '\xdcL\xbb,.\xa8\xf0\xcfv\xdc6(O\xa1bS'
Received message: u'hello'
DEBUG:accelauth.server:s_message = 'I received your message, thanks!'
DEBUG:accelauth.server:s_iv = '\xfe\x0c\nSm$\xdb\x92\xf2\x19\x02/(\xbc\xff?'
DEBUG:accelauth.server:s_ciphertext = '\xb2=\xf5a\x99\xf46\xdfB\xe4\xa5\xc2\xaei\
    ↪xc7\xcaT\xdc_\xb1jM\xad\x05J\xc4%m.\\\t\xf6P\x1b7\x1c\x97)iz\x15\n\xf0\xcf
    ↪}\xf3ku'
DEBUG:accelauth.server:c_iv = '\x83\xed[Ot\x81\xd1\xbb\xb3\x0e\xbd$\xb9U\xfa\xab'
DEBUG:accelauth.server:c_ciphertext = '\x0c\xef\xcc\x8e\xbb\x8c\x90\xbf\x83+X\xbeQ
    ↪\xc5]J'
Received message: u'logout'
DEBUG:accelauth.server:s_message = 'Goodbye!'
DEBUG:accelauth.server:s_iv = 'n\x7f\xd0\xb6\x0b\xc5K\x8a\xfd\xe2\xea\x03J\x00\x8e
    ↪\xc4'
DEBUG:accelauth.server:s_ciphertext = "F\xc0\xa5r\xea\xbao\x18'k\x06\x03(\xca\x8e\
    ↪x8b"
```