POWER EFFICIENT AND SECURE DATA STREAMING FOR MIDDLEWARE

BACKGROUND

• Low power and low cost devices lie at the heart of IoT, Raspberry Pi is a popular low cost computing device.
• Middleware architecture encourages this connectivity for a large number of devices and application layer over a network.
• 24 Billion IoT devices would be connected to the internet by 2020, demanding security from hackers.
• Encrypting connection vs. Encrypting data
  • Encrypt Connection – Data is exposed if connection is hacked into
  • Encrypt Data – Data cannot be accessed unless the correct key is present at the receiver end (end-to-end encryption)
  • Encrypt Data – Even if the connection is hacked, data remains secure

MOTIVATION

• Extensive use of IoT devices in healthcare monitoring
• Numerous articles highlighting search engines for insecure baby monitor setups that expose private data on internet.
• Pressing need for security implementation, particularly device-to-device data encryption rather than just network encryption
RELATED WORK

• Several DIY streaming implementations developed by Raspberry Pi enthusiasts.
• Software solutions like gstreamer, mjpegs for streaming video and audio.
• Libraries like OpenSSL, Libgcrypt already exist.

WHY AES

• AES is the required standard by NSA for exchange of classified information, which indicates the strength of the algorithm being used.
• It is less susceptible to cryptanalysis and supports larger key sizes (as compared to DES).
• It is proven to be faster than DES on both hardware and software.
• While SSL is good as well for encrypting data during transmission, AES is proven better for encrypting data for transmission and storage, particularly for online backups.

SOFTWARE

• TCP protocol used for data transmission between RPis
• AES 128 implementation on ARM-NEON ISA
• AES mode used is Cipher Feedback (CFB), because:
  • It saves space (silicon and/or machine code) in extremely constricted environments, as the encryption cipher is used for decryption as well.
  • Decryption is parallelizable.
  • Transmission errors destroy the following block, but it affects only the wrong bits in the current block
HARDWARE
• 2 Raspberry Pi units (one server and one client)
• Edimax Wi-fi module to create ad-hoc network for data exchange (it is an unencrypted network)
• RaspiCam module to capture video
• Kingwin USB - 3 DSA USB Stereo 3D Sound Adapter for audio input

IMPLEMENTATION
• We are performing Pi-to-Pi (low-power device to low-power device) audio and video data streaming.
• We have created an ad-hoc Wifi network using WiFi modules for Pi-to-Pi interaction and data exchange.
• Raspberry Pi acts as a Middleware, collecting inputs from audio/video sensors, encrypts and forwards it on the network to the receiver Raspberry Pi.
• We are using AES CFB for data encryption
• Current implementation is based on initial key pairing, thus key is already present at sender and receiver Raspberry Pis.
• Received data is decrypted using AES CFB decryption.
• Underlying intention is to expand the number of sensors from the present two sensors for a powerful real-time middleware implementation that involves further processing on this collected data.
• We have implemented AES CFB on ARM v7 NEON engine, rather than a regular CPU implementation of the cipher. Following are the advantages of implementing it on NEON when compared to CPU:
  • It is faster, as the system now works on 16 bytes of data in a single cycle.
  • CPU utilization reduces nearly by 50% when implemented on NEON engine. (Results discussed later)

PROBLEMS FACED AND TROUBLESHOOTING
• Initial plan was to implement AES CFB 128 cipher on GPU. The implementation proved to be difficult due to:
  • Lack of documentation
  • No standardised assembler/language

FURTHER WORK AND ENHANCEMENT
• AES-GCM is the most efficient and secure of all AES modes, which can be looked into for assembly implementation on NEON.
• Current implementation is based on initial key pairing, which can be replaced by public/private key protocols.

RESULT HIGHLIGHTS
Time taken to decrypt one block (per 16 byte block)
NEON - 3.67 us
CPU - 4.14 us
## Performance of the code (Success Metrics)

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>NEON</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Video</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HD (1280x720) 25 fps Avg. Delay: 8.4 sec</td>
<td>HD (1280x720) 25 fps Instantaneous playback</td>
<td>HD (1280x720) 30 fps Instantaneous playback</td>
</tr>
<tr>
<td><strong>Audio</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 bit 44.1kHz Stereo Avg. Delay: 3 sec</td>
<td>16 bit 44.1kHz Stereo Avg. Delay: 2.8 sec</td>
<td></td>
</tr>
<tr>
<td><strong>CPU Utilization</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full HD: ~40%</td>
<td>HD (1280x720): ~27%</td>
<td>Full HD: ~18%</td>
</tr>
</tbody>
</table>

*(Debanjan Chatterjee, Neha Ahlawat)*