P2P File Transfer

Executive Summary

According to MarketsandMarkets research study, “The demand for encryption software is likely to be driven by various factors, such as proliferation in the number of cyber-attacks…”. Taking this into consideration, how can we make sure that any of the information shared across a network is safe?

Our software company, P2P File Transfer, has created a safe way to share information by facilitating encrypted file sharing. Because of the rapid growth for the need of encryption use, our product offers tremendous advantages to company professionals, as well as the thousands of other people who use networks like the internet to share information every day.

Our software was developed in Python. It is built with email delivered download file options that consist of “Receiver” and “Sender”. P2P File Transfer software starts with a Registration PIN, continues with Two-Factor Authentication powered by Google Authenticator, and finishes with Two-Stage Encryption. Our software uses both AES symmetric encryption as well as RSA asymmetric encryption to avoid file size limitations per AES(Brent).

Per Wireshark analysis, files are encrypted before being sent to the other party using TCP. The potential market for the product is very promising, because we solved the problem of how to privately transfer files.

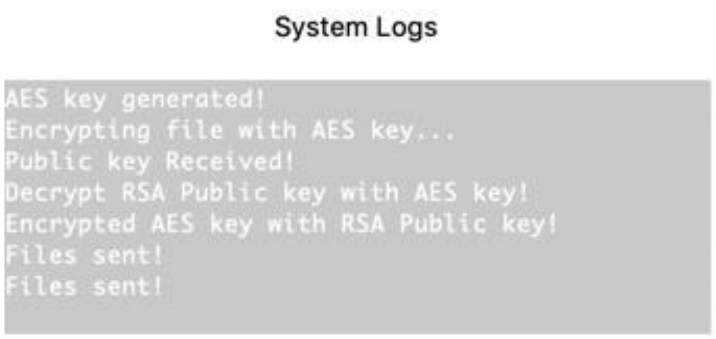
Core Concept

Problem—How can we share files (intellectual property, contracts, employee and/or customer information, medical records, other documents you need to keep confidential) securely, if the Internet is an open space and networks are subject to snooping?

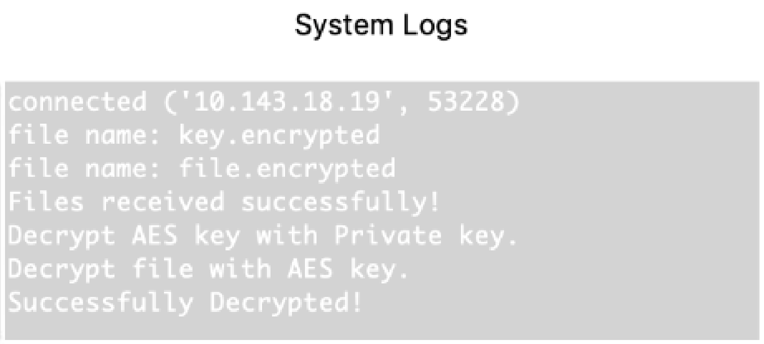
Problem to worsen—"The encryption software market size is expected to grow from USD 3.87 Billion in 2017 to USD 12.96 Billion by 2022, at a Compound Annual Growth Rate (CAGR) of 27.4%. The demand for encryption software is likely to be driven by various factors, such as proliferation in the number of cyber-attacks and the stringent government regulations and compliances that mandate the adoption of encryption among various verticals"(Encryption Software Market by Component, MarketsandMarkets.com).

Solution—Encrypt files so only the intended party has access to see the shared files. Authenticate network users to keep unauthenticated users out.

Our Approach—We created software that incorporates our solution to solve the problem articulated above. Our software enables person-to-person secure file sharing and is called, P2P Transfer. It is built with email delivered download options of "Receiver" or "Sender" files, according to using your computer to either send or receive the file(s) to be shared. P2P File Transfer software uses three core concepts to securely share files: The first is, a Registration PIN; it allows you to establish a connection with our P2P File Transfer software. The second is, Two-Factor Authentication; it verifies you are who you claim to be via the Google Authenticator App, which generates a Verification Code to your cell phone after you scan a QR Code to complete the registration process. The third is, Two-Stage Encryption; our product uses both AES symmetric encryption and RSA asymmetric encryption to avoid file size limitations per AES(Brent) and was created in Python. Below are our P2P File Transfer software System Logs to verify encryption. For complete interface of P2P File Transfer software see Advanced User Guide section. Since the files are encrypted between the "Sender" and "Receiver", the files sent between them are completely private as they move across any network including the open internet. Our private connection is verified by diagnostic testing of P2P File Transfer software via Wireshark diagnostics.


System Logs

```
AES key generated!
Encrypting file with AES key...
Public key Received!
Decrypt RSA Public key with AES key!
Encrypted AES key with RSA Public key!
Files sent!
Files sent!
```


System Logs

```
connected ('10.143.18.19', 53228)
file name: key.encrypted
file name: file.encrypted
Files received successfully!
Decrypt AES key with Private key.
Decrypt file with AES key.
Successfully Decrypted!
```

Our Software—Our P2P File Transfer software is a prototype with the requirement to meet in person to provide/receive the Host Name/IP, Port Name, and Registration PIN—alternatively this information can be mailed. Additionally, a cell phone with the Google Authenticator App and a working email address will be needed to use our software.

Additional P2P File Transfer Software Benefits—P2P File Transfer software can deliver secure file sharing capacity without internet connection required as it functions on a LAN. P2P File Transfer software does encryption internally so you do not have to worry about storing your block encryption key(s). Since P2P File Transfer software encrypts your files, there is no need to worry about snooping on the network any longer. Two-factor authentication can thwart middle-man-attacks as only you will have your cell phone and know its passcode to use it. User friendly interface makes secured file sharing a simple process.

Future Enhancements—Due out next semester, we would make our software accessible via a web page rather than email download. Then we would use Apache Cordova or Ionic Framework(Verma) to mobilize our web page and present our web page as an App to users. It would remove the need for authenticating the user, emailing the software for download, and enable billing of P2P File Transfer on a pay as you go basis available on your device. Additionally, we would move towards strictly using asymmetric encryption to eliminate the need to meet in person as is currently required with our current software prototype. We would also explore delivery of the public key by sharing it after the # in our link to our web page, for example: P2PTransfer.com#public-key-goes-here-for-sender-to-use(Sneddon, Joey) to avoid having to find a place host your public key.

Our Team—Jia Da Wu –Software Development and How to get started Advanced User Guide Lead, Keshawn Gosiengfiao— Wireshark Diagnostic Testing Lead, Lilia Barajas Lopez—Core Concept and How to get started Beginning User Guide Lead, Victor Llano Mariota— Executive Summary Lead (where not listed as Lead, other team members supported Lead).

How to get started: Beginner User & Advanced User guides

Beginner User Guide:

People acting as “Receiver” and “Sender” will need to meet in person to provide/receive the Host Name/IP, Port Name, and Registration PIN. If people acting as “Receiver” and “Sender” cannot meet in person, we recommend the “Sender” request the details in writing so the “Receiver” can physically mail you the needed information via FedEx. Cell phone needed for authentication and must have QR Scanner app present. Email that is working is required to receive P2P File Transfer software download files.

How to Get Started as “Receiver”:

1. Download program file: Open file, select Receiver folder, and download it(“tkinter — Python interface to Tcl/Tk”).
2. Push “Register” button, enter Registration PIN provided(“PyOTP - The Python One-Time Password Library”; “PyQRCode 1.2.1”).
3. Scan QR Code with phone to receive required “Verification Code”(“PyOTP - The Python One-Time Password Library”; “PyQRCode 1.2.1”).
4. Browse for Save File location by pushing “Save File” button.
5. Enter “Verification Code” generated from google authenticator to your cell phone(“PyOTP - The Python One-Time Password Library”).
6. Push “Start Server” button.

How to Get Started as “Sender”:

1. Download program file: Open file, select Sender folder, and download it(“tkinter — Python interface to Tcl/Tk”).
2. Push “Register” button, enter Registration PIN provided(“PyOTP - The Python One-Time Password Library”; “PyQRCode 1.2.1”).
3. Scan QR Code with your phone to receive required “Verification Code”(“PyOTP - The Python One-Time Password Library”; “PyQRCode 1.2.1”).
4. Enter the Host Name/IP and Port Name provided(“tkinter — Python interface to Tcl/Tk”).
5. Push “Select File” button and select the file you want to send(“tkinter — Python Interface to Tcl/Tk”).
6. Enter “Verification Code” generated from google authenticator to your cell phone(“PyOTP - The Python One-Time Password Library”; “PyQRCode 1.2.1”).
7. Push “Send” button; file(s) will be encrypted internally and sent to the “Receiver”(“Cryptography 2.6.1”).

<u>Advanced User Guide:</u>

Step 1

Administrator/Receiver:

Software, 6 digits Passcode – for authentication register, QR code Register, Hostname/IP, Port Number.

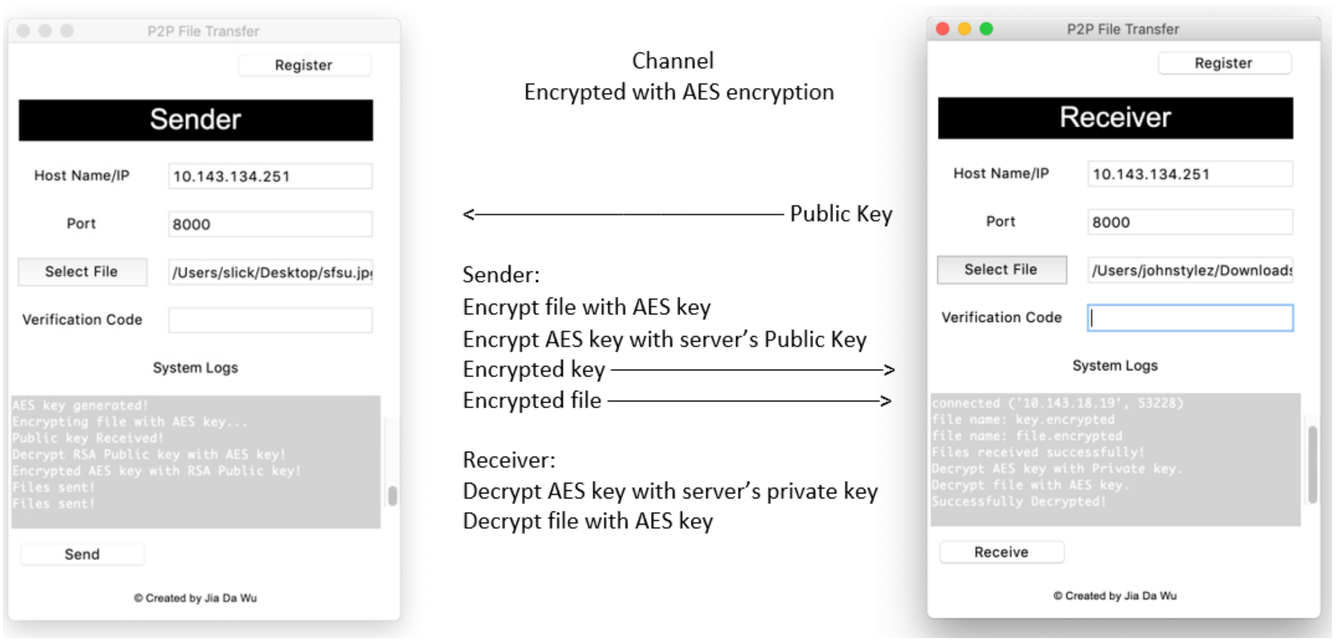
Step 2

1. Download the software.
2. Open software(“tkinter – python interface to Tc;/Tk”).
3. Register – obtain the code from Admin/Receiver(“PyOTP – The Python One-Time Password Library”; PyQRCode 1.2.1”).
4. Scan QR code with google authenticator to complete the registration(PyOTP – The Python One-Time Password Library; PyQRCode 1.2.1).
5. Pick a location to save the file.
6. Enter verification code – obtain the code form Google Authenticator(“PyOTP – The Python One-Time Password Library”).
7. Click “receive”.

Step 3

1. Download the software.
2. Open software(“tkinter – python interface to Tc;/Tk”).
3. Register – obtain the code from Admin(“PyOTP – The Python One-Time Password Library”; “PyQRCode 1.2.1”).
4. Scan QR code with google authenticator to complete the registration(“PyOTP – The Python One-Time Password Library”; “PyQRCode 1.2.1”).

5. Enter hostname/IP – obtain from Admin/Receiver("tkinter – python interface to Tc;/Tk").
6. Enter port number – obtain from Admin/Receiver("tkinter – python interface to Tc;/Tk").
7. Select a file.
8. Enter verification code – obtain the code from google authenticator("PyOTP – The Python One-Time Password Library"; "PyQRCode 1.2.1').
9. Click "Send"



Host Name/IP: 10.143.134.251
Port: 8000
Select File: /Users/slick/Desktop/sfsu.jp
Verification Code:

System Logs

```
AES key generated!
Encrypting file with AES key...
Public key Received!
Decrypt RSA Public key with AES key!
Encrypted AES key with RSA Public key!
Files sent!
Files sent!
```

Sender:
Encrypt file with AES key
Encrypt AES key with server's Public Key
Encrypted key
Encrypted file

Receiver:
Decrypt AES key with server's private key
Decrypt file with AES key

Host Name/IP: 10.143.134.251
Port: 8000
Select File: /Users/johnstylez/Download
Verification Code:

System Logs

```
connected ('10.143.18.19', 53228)
file name: key.encrypted
file name: file.encrypted
Files received successfully!
Decrypt AES key with Private key.
Decrypt file with AES key.
Successfully Decrypted!
```

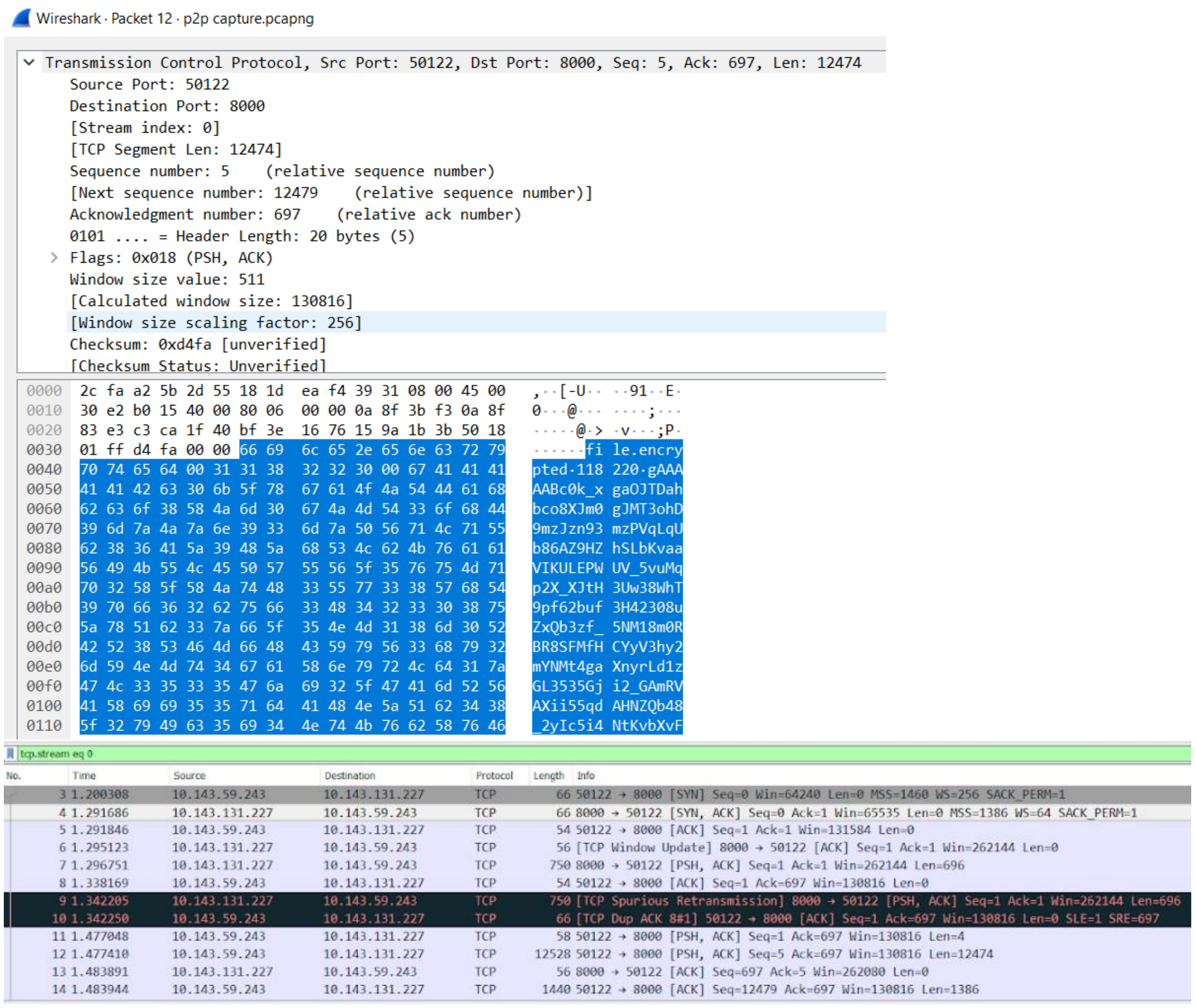
Wireshark Evidence

Application Layer:

- The software itself is the application layer.
- Since the transmission is encrypted, Wireshark evidence for the Application Layer is not available.

TCP Layer:

- TCP layer is shown by using TCP to transmit data from one place to another.
- TCP is used from beginning to end for the File Sharing process.
- Proof in Lines 3-113 in Wireshark.



Wireshark · Packet 12 · p2p capture.pcapng

```
∨ Transmission Control Protocol, Src Port: 50122, Dst Port: 8000, Seq: 5, Ack: 697, Len: 12474
    Source Port: 50122
    Destination Port: 8000
    [Stream index: 0]
    [TCP Segment Len: 12474]
    Sequence number: 5    (relative sequence number)
    [Next sequence number: 12479    (relative sequence number)]
    Acknowledgment number: 697    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    Window size value: 511
    [Calculated window size: 130816]
    [Window size scaling factor: 256]
    Checksum: 0xd4fa [unverified]
    [Checksum Status: Unverified]
```

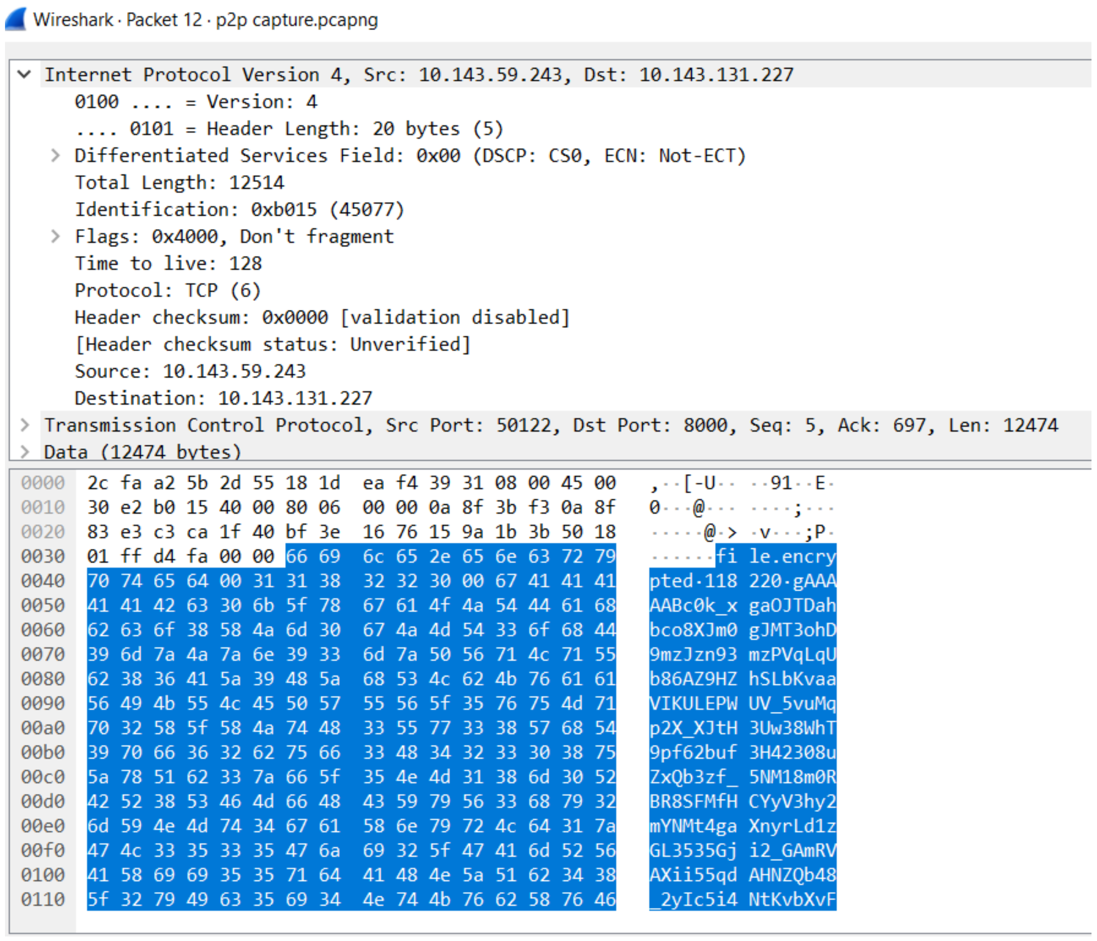
```
0000  2c fa a2 5b 2d 55 18 1d  ea f4 39 31 08 00 45 00   ,··[-U·· ··91··E·
0010  30 e2 b0 15 40 00 80 06  00 00 0a 8f 3b f3 0a 8f   0···@··· ····;···
0020  83 e3 c3 ca 1f 40 bf 3e  16 76 15 9a 1b 3b 50 18   ·····@·> ·v···;P·
0030  01 ff d4 fa 00 00 66 69  6c 65 2e 65 6e 63 72 79   ······fi le.encry
0040  70 74 65 64 00 31 31 38  32 32 30 00 67 41 41 41   pted·118 220·gAAA
0050  41 41 42 63 30 6b 5f 78  67 61 4f 4a 54 44 61 68   AABc0k_x gaOJTDah
0060  62 63 6f 38 58 4a 6d 30  67 4a 4d 54 33 6f 68 44   bco8XJm0 gJMT3ohD
0070  39 6d 7a 4a 7a 6e 39 33  6d 7a 50 56 71 4c 71 55   9mzJzn93 mzPVqLqU
0080  62 38 36 41 5a 39 48 5a  68 53 4c 62 4b 76 61 61   b86AZ9HZ hSLbKvaa
0090  56 49 4b 55 4c 45 50 57  55 56 5f 35 76 75 4d 71   VIKULEPW UV_5vuMq
00a0  70 32 58 5f 58 4a 74 48  33 55 77 33 38 57 68 54   p2X_XJtH 3Uw38WhT
00b0  39 70 66 36 32 62 75 66  33 48 34 32 33 30 38 75   9pf62buf 3H42308u
00c0  5a 78 51 62 33 7a 66 5f  35 4e 4d 31 38 6d 30 52   ZxQb3zf_ 5NM18m0R
00d0  42 52 38 53 46 4d 66 48  43 59 79 56 33 68 79 32   BR8SFMfH CYyV3hy2
00e0  6d 59 4e 4d 74 34 67 61  58 6e 79 72 4c 64 31 7a   mYNMt4ga XnyrLd1z
00f0  47 4c 33 35 33 35 47 6a  69 32 5f 47 41 6d 52 56   GL3535Gj i2_GAmRV
0100  41 58 69 69 35 35 71 64  41 48 4e 5a 51 62 34 38   AXii55qd AHNZQb48
0110  5f 32 79 49 63 35 69 34  4e 74 4b 76 62 58 76 46   _2yIc5i4 NtKvbXvF
```

tcp.stream eq 0

```
No.  Time      Source          Destination     Protocol Length Info
   3 1.200308  10.143.59.243   10.143.131.227  TCP         66 50122 → 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   4 1.291686  10.143.131.227  10.143.59.243   TCP         66 8000 → 50122 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1386 WS=64 SACK_PERM=1
   5 1.291846  10.143.59.243   10.143.131.227  TCP         54 50122 → 8000 [ACK] Seq=1 Ack=1 Win=131584 Len=0
   6 1.295123  10.143.131.227  10.143.59.243   TCP         56 [TCP Window Update] 8000 → 50122 [ACK] Seq=1 Ack=1 Win=262144 Len=0
   7 1.296751  10.143.131.227  10.143.59.243   TCP        750 8000 → 50122 [PSH, ACK] Seq=1 Ack=1 Win=262144 Len=696
   8 1.338169  10.143.59.243   10.143.131.227  TCP         54 50122 → 8000 [ACK] Seq=1 Ack=697 Win=130816 Len=0
   9 1.342205  10.143.131.227  10.143.59.243   TCP        750 [TCP Spurious Retransmission] 8000 → 50122 [PSH, ACK] Seq=1 Ack=1 Win=262144 Len=696
  10 1.342250  10.143.59.243   10.143.131.227  TCP         66 [TCP Dup ACK 8#1] 50122 → 8000 [ACK] Seq=1 Ack=697 Win=130816 Len=0 SLE=1 SRE=697
  11 1.477048  10.143.59.243   10.143.131.227  TCP         58 50122 → 8000 [PSH, ACK] Seq=1 Ack=697 Win=130816 Len=4
  12 1.477410  10.143.59.243   10.143.131.227  TCP      12528 50122 → 8000 [PSH, ACK] Seq=5 Ack=697 Win=130816 Len=12474
  13 1.483891  10.143.131.227  10.143.59.243   TCP         56 8000 → 50122 [ACK] Seq=697 Ack=5 Win=262080 Len=0
  14 1.483944  10.143.59.243   10.143.131.227  TCP       1440 50122 → 8000 [ACK] Seq=12479 Ack=697 Win=130816 Len=1386
```

IP Layer:

- Sender IP: 10.143.59.243.
- Destination IP: 10.143.131.227.
- (Evidence in capture. Sender and Destination address are found in multiple lines).



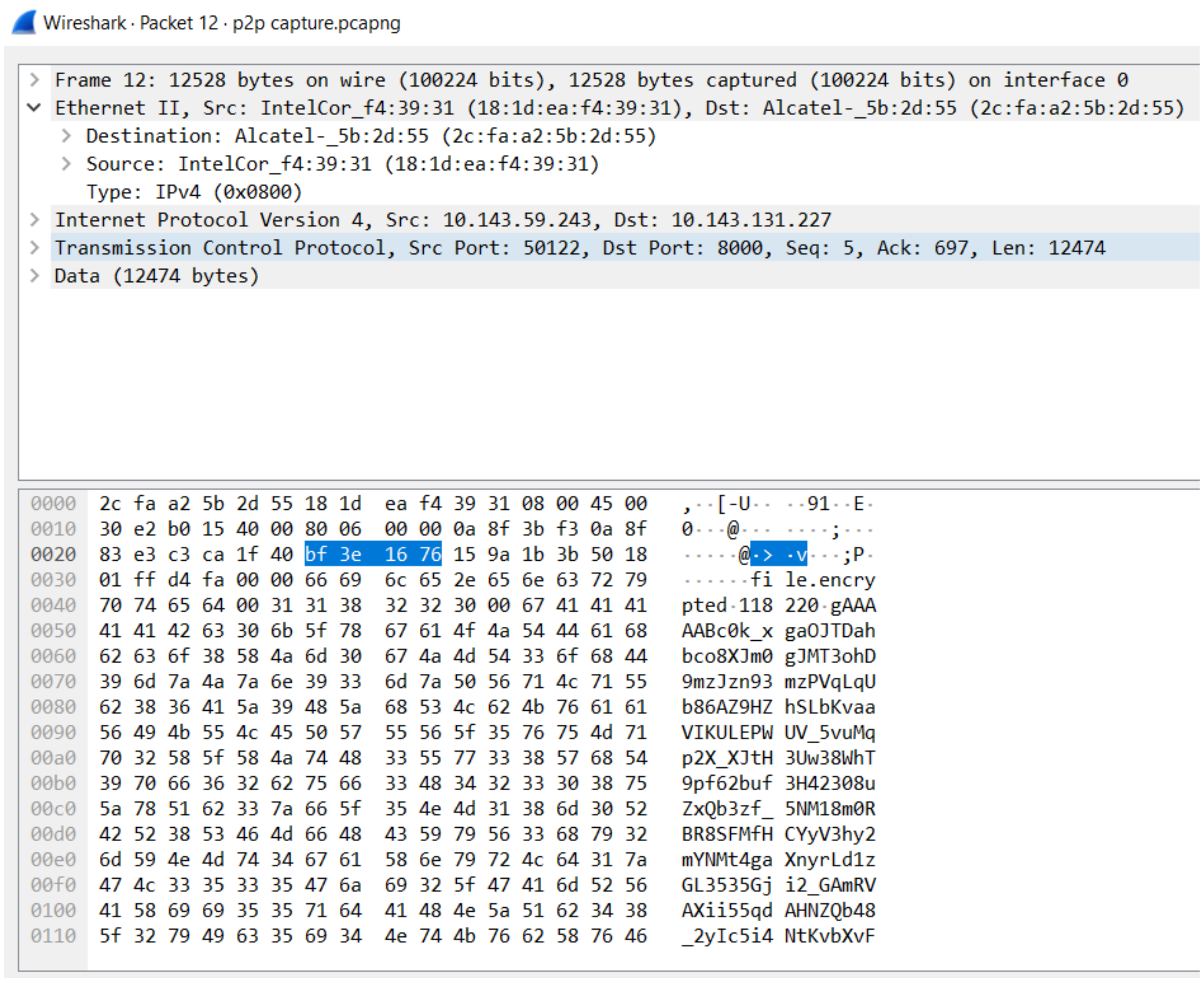
Wireshark · Packet 12 · p2p capture.pcapng

```
v Internet Protocol Version 4, Src: 10.143.59.243, Dst: 10.143.131.227
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 12514
    Identification: 0xb015 (45077)
  > Flags: 0x4000, Don't fragment
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.143.59.243
    Destination: 10.143.131.227
> Transmission Control Protocol, Src Port: 50122, Dst Port: 8000, Seq: 5, Ack: 697, Len: 12474
> Data (12474 bytes)
```

```
0000  2c fa a2 5b 2d 55 18 1d  ea f4 39 31 08 00 45 00   ,··[-U·· ··91··E·
0010  30 e2 b0 15 40 00 80 06  00 00 0a 8f 3b f3 0a 8f   0···@··· ····;···
0020  83 e3 c3 ca 1f 40 bf 3e  16 76 15 9a 1b 3b 50 18   ·····@·> ·v···;P·
0030  01 ff d4 fa 00 00 66 69  6c 65 2e 65 6e 63 72 79   ······fi le.encry
0040  70 74 65 64 00 31 31 38  32 32 30 00 67 41 41 41   pted·118 220·gAAA
0050  41 41 42 63 30 6b 5f 78  67 61 4f 4a 54 44 61 68   AABc0k_x gaOJTDah
0060  62 63 6f 38 58 4a 6d 30  67 4a 4d 54 33 6f 68 44   bco8XJm0 gJMT3ohD
0070  39 6d 7a 4a 7a 6e 39 33  6d 7a 50 56 71 4c 71 55   9mzJzn93 mzPVqLqU
0080  62 38 36 41 5a 39 48 5a  68 53 4c 62 4b 76 61 61   b86AZ9HZ hSLbKvaa
0090  56 49 4b 55 4c 45 50 57  55 56 5f 35 76 75 4d 71   VIKULEPW UV_5vuMq
00a0  70 32 58 5f 58 4a 74 48  33 55 77 33 38 57 68 54   p2X_XJtH 3Uw38WhT
00b0  39 70 66 36 32 62 75 66  33 48 34 32 33 30 38 75   9pf62buf 3H42308u
00c0  5a 78 51 62 33 7a 66 5f  35 4e 4d 31 38 6d 30 52   ZxQb3zf_ 5NM18m0R
00d0  42 52 38 53 46 4d 66 48  43 59 79 56 33 68 79 32   BR8SFMfH CYyV3hy2
00e0  6d 59 4e 4d 74 34 67 61  58 6e 79 72 4c 64 31 7a   mYNMt4ga XnyrLd1z
00f0  47 4c 33 35 33 35 47 6a  69 32 5f 47 41 6d 52 56   GL3535Gj i2_GAmRV
0100  41 58 69 69 35 35 71 64  41 48 4e 5a 51 62 34 38   AXii55qd AHNZQb48
0110  5f 32 79 49 63 35 69 34  4e 74 4b 76 62 58 76 46   _2yIc5i4 NtKvbXvF
```

Data Link:

- Wi-Fi.
- Ethernet.
- Ethernet 2.



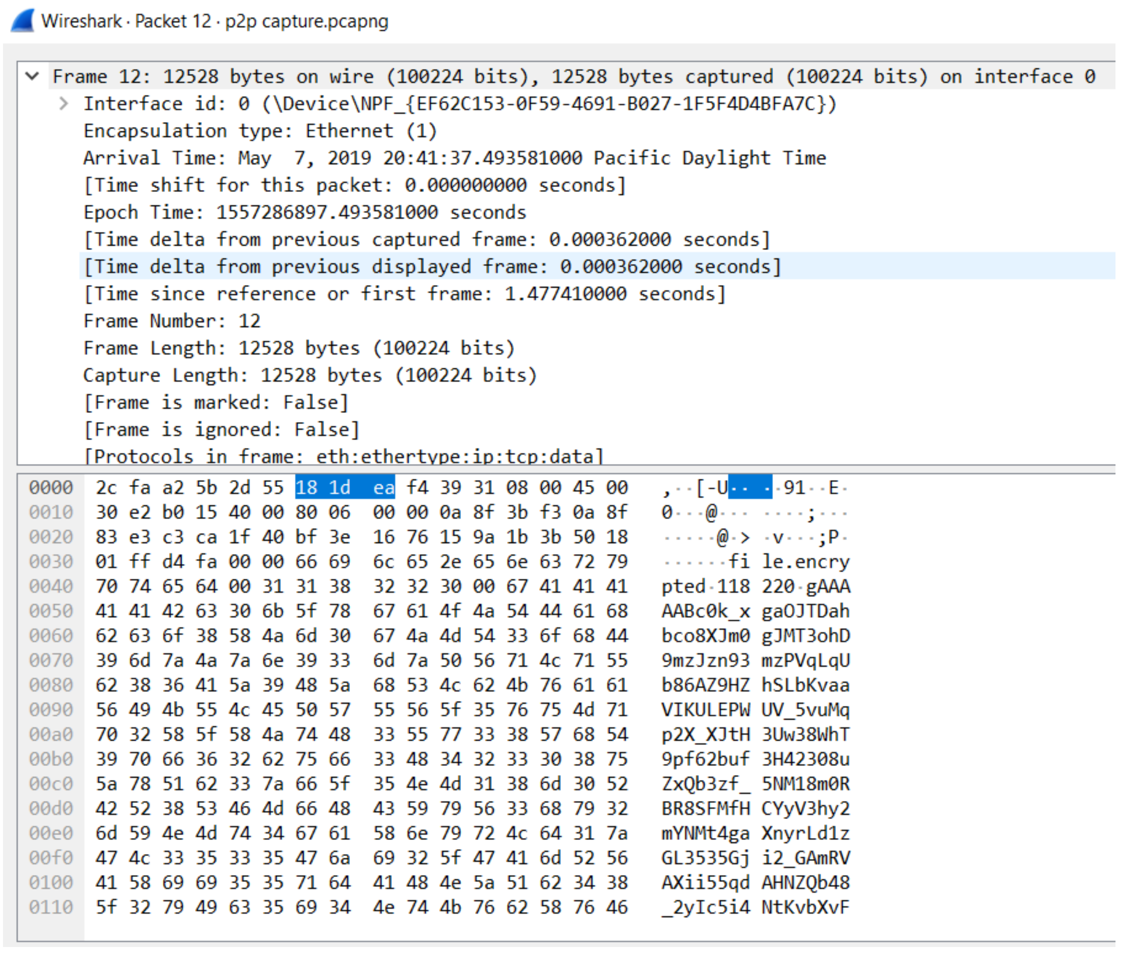
Wireshark · Packet 12 · p2p capture.pcapng

```
> Frame 12: 12528 bytes on wire (100224 bits), 12528 bytes captured (100224 bits) on interface 0
v Ethernet II, Src: IntelCor_f4:39:31 (18:1d:ea:f4:39:31), Dst: Alcatel-_5b:2d:55 (2c:fa:a2:5b:2d:55)
  > Destination: Alcatel-_5b:2d:55 (2c:fa:a2:5b:2d:55)
  > Source: IntelCor_f4:39:31 (18:1d:ea:f4:39:31)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 10.143.59.243, Dst: 10.143.131.227
> Transmission Control Protocol, Src Port: 50122, Dst Port: 8000, Seq: 5, Ack: 697, Len: 12474
> Data (12474 bytes)
```

```
0000  2c fa a2 5b 2d 55 18 1d  ea f4 39 31 08 00 45 00   ,··[-U·· ··91··E·
0010  30 e2 b0 15 40 00 80 06  00 00 0a 8f 3b f3 0a 8f   0···@··· ····;···
0020  83 e3 c3 ca 1f 40 bf 3e  16 76 15 9a 1b 3b 50 18   ·····@·> ·v···;P·
0030  01 ff d4 fa 00 00 66 69  6c 65 2e 65 6e 63 72 79   ······fi le.encry
0040  70 74 65 64 00 31 31 38  32 32 30 00 67 41 41 41   pted·118 220·gAAA
0050  41 41 42 63 30 6b 5f 78  67 61 4f 4a 54 44 61 68   AABc0k_x gaOJTDah
0060  62 63 6f 38 58 4a 6d 30  67 4a 4d 54 33 6f 68 44   bco8XJm0 gJMT3ohD
0070  39 6d 7a 4a 7a 6e 39 33  6d 7a 50 56 71 4c 71 55   9mzJzn93 mzPVqLqU
0080  62 38 36 41 5a 39 48 5a  68 53 4c 62 4b 76 61 61   b86AZ9HZ hSLbKvaa
0090  56 49 4b 55 4c 45 50 57  55 56 5f 35 76 75 4d 71   VIKULEPW UV_5vuMq
00a0  70 32 58 5f 58 4a 74 48  33 55 77 33 38 57 68 54   p2X_XJtH 3Uw38WhT
00b0  39 70 66 36 32 62 75 66  33 48 34 32 33 30 38 75   9pf62buf 3H42308u
00c0  5a 78 51 62 33 7a 66 5f  35 4e 4d 31 38 6d 30 52   ZxQb3zf_ 5NM18m0R
00d0  42 52 38 53 46 4d 66 48  43 59 79 56 33 68 79 32   BR8SFMfH CYyV3hy2
00e0  6d 59 4e 4d 74 34 67 61  58 6e 79 72 4c 64 31 7a   mYNMt4ga XnyrLd1z
00f0  47 4c 33 35 33 35 47 6a  69 32 5f 47 41 6d 52 56   GL3535Gj i2_GAmRV
0100  41 58 69 69 35 35 71 64  41 48 4e 5a 51 62 34 38   AXii55qd AHNZQb48
0110  5f 32 79 49 63 35 69 34  4e 74 4b 76 62 58 76 46   _2yIc5i4 NtKvbXvF
```

Physical Layer:

- Mac or Windows Computer with network capabilities (Wi-Fi or Ethernet).
- Mac Address/Physical Evidence is shown in capture.

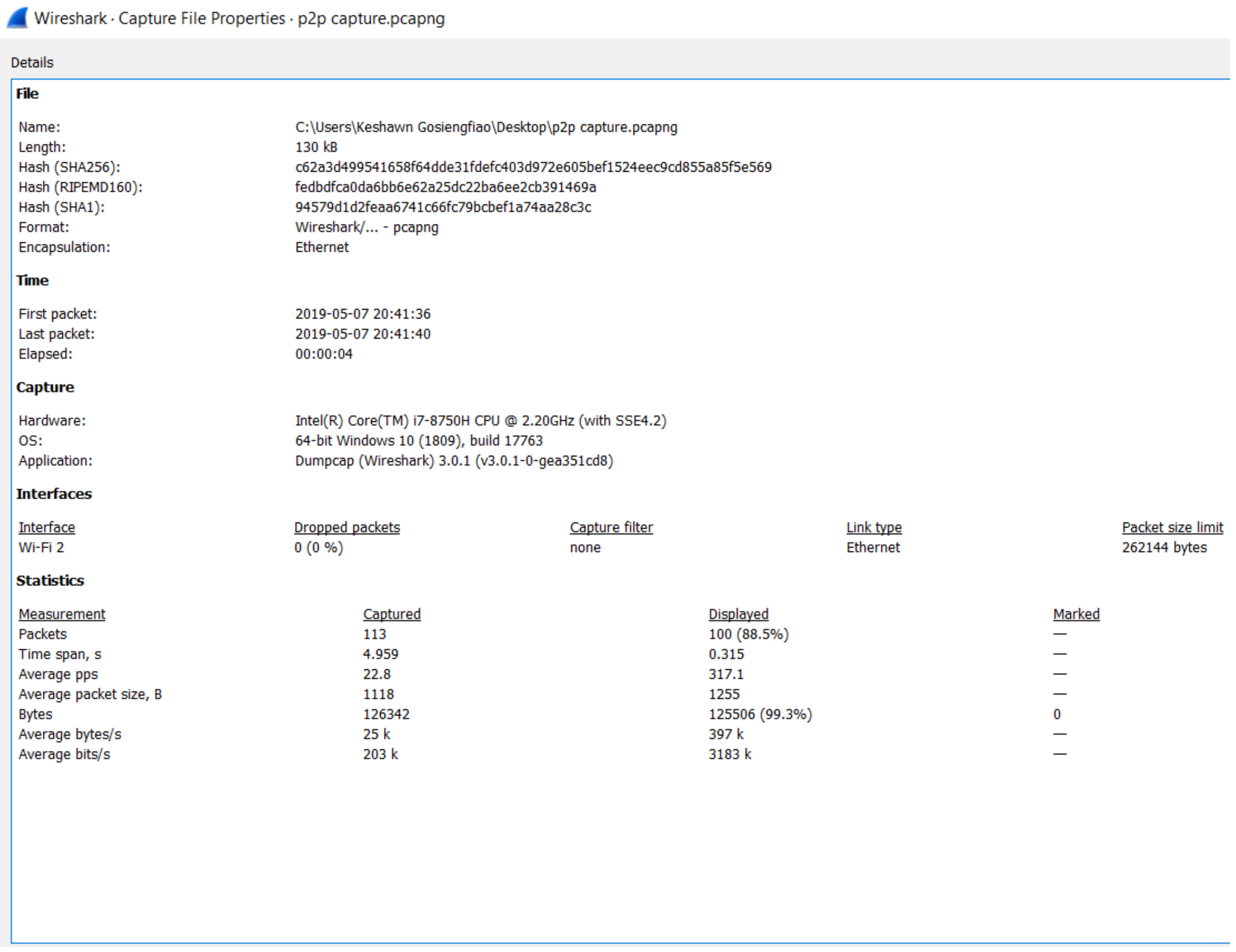


Wireshark · Packet 12 · p2p capture.pcapng

```
Frame 12: 12528 bytes on wire (100224 bits), 12528 bytes captured (100224 bits) on interface 0
  > Interface id: 0 (\Device\NPF_{EF62C153-0F59-4691-B027-1F5F4D4BFA7C})
    Encapsulation type: Ethernet (1)
    Arrival Time: May  7, 2019 20:41:37.493581000 Pacific Daylight Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1557286897.493581000 seconds
    [Time delta from previous captured frame: 0.000362000 seconds]
    [Time delta from previous displayed frame: 0.000362000 seconds]
    [Time since reference or first frame: 1.477410000 seconds]
    Frame Number: 12
    Frame Length: 12528 bytes (100224 bits)
    Capture Length: 12528 bytes (100224 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp:data]
```

```
0000  2c fa a2 5b 2d 55 18 1d  ea f4 39 31 08 00 45 00   ,··[-U·· ··91··E·
0010  30 e2 b0 15 40 00 80 06  00 00 0a 8f 3b f3 0a 8f   0···@··· ····;···
0020  83 e3 c3 ca 1f 40 bf 3e  16 76 15 9a 1b 3b 50 18   ·····@·> ·v···;P·
0030  01 ff d4 fa 00 00 66 69  6c 65 2e 65 6e 63 72 79   ······fi le.encry
0040  70 74 65 64 00 31 31 38  32 32 30 00 67 41 41 41   pted·118 220·gAAA
0050  41 41 42 63 30 6b 5f 78  67 61 4f 4a 54 44 61 68   AABc0k_x gaOJTDah
0060  62 63 6f 38 58 4a 6d 30  67 4a 4d 54 33 6f 68 44   bco8XJm0 gJMT3ohD
0070  39 6d 7a 4a 7a 6e 39 33  6d 7a 50 56 71 4c 71 55   9mzJzn93 mzPVqLqU
0080  62 38 36 41 5a 39 48 5a  68 53 4c 62 4b 76 61 61   b86AZ9HZ hSLbKvaa
0090  56 49 4b 55 4c 45 50 57  55 56 5f 35 76 75 4d 71   VIKULEPW UV_5vuMq
00a0  70 32 58 5f 58 4a 74 48  33 55 77 33 38 57 68 54   p2X_XJtH 3Uw38WhT
00b0  39 70 66 36 32 62 75 66  33 48 34 32 33 30 38 75   9pf62buf 3H42308u
00c0  5a 78 51 62 33 7a 66 5f  35 4e 4d 31 38 6d 30 52   ZxQb3zf_ 5NM18m0R
00d0  42 52 38 53 46 4d 66 48  43 59 79 56 33 68 79 32   BR8SFMfH CYyV3hy2
00e0  6d 59 4e 4d 74 34 67 61  58 6e 79 72 4c 64 31 7a   mYNMt4ga XnyrLd1z
00f0  47 4c 33 35 33 35 47 6a  69 32 5f 47 41 6d 52 56   GL3535Gj i2_GAmRV
0100  41 58 69 69 35 35 71 64  41 48 4e 5a 51 62 34 38   AXii55qd AHNZQb48
0110  5f 32 79 49 63 35 69 34  4e 74 4b 76 62 58 76 46   _2yIc5i4 NtKvbXvF
```

Overview:



Wireshark · Capture File Properties · p2p capture.pcapng

Details

File

Name:	C:\Users\Keshawn Gosiengfiao\Desktop\p2p capture.pcapng
Length:	130 kB
Hash (SHA256):	c62a3d499541658f64dde31fdefc403d972e605bef1524eec9cd855a85f5e569
Hash (RIPEMD160):	fedbdfca0da6bb6e62a25dc22ba6ee2cb391469a
Hash (SHA1):	94579d1d2feaa6741c66fc79bcbef1a74aa28c3c
Format:	Wireshark/... - pcapng
Encapsulation:	Ethernet

Time

First packet:	2019-05-07 20:41:36
Last packet:	2019-05-07 20:41:40
Elapsed:	00:00:04

Capture

Hardware:	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz (with SSE4.2)
OS:	64-bit Windows 10 (1809), build 17763
Application:	Dumpcap (Wireshark) 3.0.1 (v3.0.1-0-gea351cd8)

Interfaces

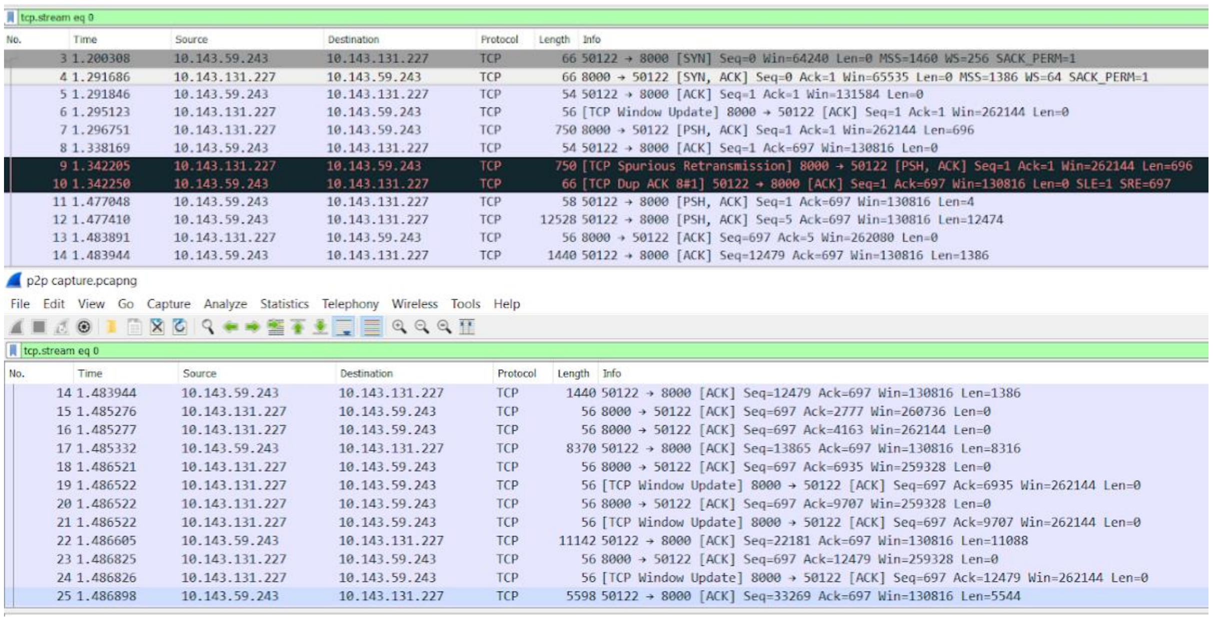
Interface	Dropped packets	Capture filter	Link type	Packet size limit
Wi-Fi 2	0 (0 %)	none	Ethernet	262144 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	113	100 (88.5%)	—
Time span, s	4.959	0.315	—
Average pps	22.8	317.1	—
Average packet size, B	1118	1255	—
Bytes	126342	125506 (99.3%)	0
Average bytes/s	25 k	397 k	—
Average bits/s	203 k	3183 k	—

Expected Protocol:

- Expected Protocol is TCP for File Transmission (Lines 3-113).


p2p capture.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.stream eq 0

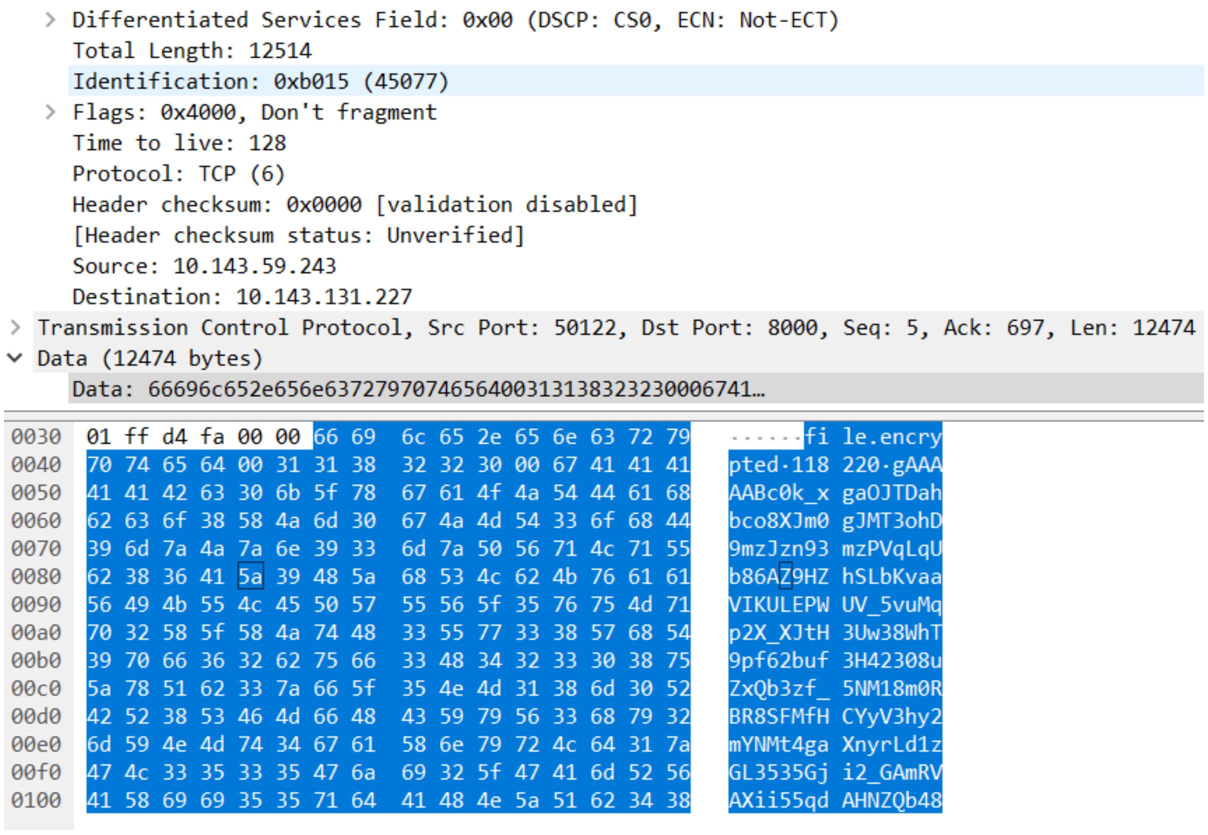
No.	Time	Source	Destination	Protocol	Length	Info
3	1.200308	10.143.59.243	10.143.131.227	TCP	66	50122 → 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
4	1.291686	10.143.131.227	10.143.59.243	TCP	66	8000 → 50122 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1386 WS=64 SACK_PERM=1
5	1.291846	10.143.59.243	10.143.131.227	TCP	54	50122 → 8000 [ACK] Seq=1 Ack=1 Win=131584 Len=0
6	1.295123	10.143.131.227	10.143.59.243	TCP	56	[TCP Window Update] 8000 → 50122 [ACK] Seq=1 Ack=1 Win=262144 Len=0
7	1.296751	10.143.131.227	10.143.59.243	TCP	750	8000 → 50122 [PSH, ACK] Seq=1 Ack=1 Win=262144 Len=696
8	1.338169	10.143.59.243	10.143.131.227	TCP	54	50122 → 8000 [ACK] Seq=1 Ack=697 Win=130816 Len=0
9	1.342205	10.143.131.227	10.143.59.243	TCP	750	[TCP Spurious Retransmission] 8000 → 50122 [PSH, ACK] Seq=1 Ack=1 Win=262144 Len=696
10	1.342250	10.143.59.243	10.143.131.227	TCP	66	[TCP Dup ACK 8#1] 50122 → 8000 [ACK] Seq=1 Ack=697 Win=130816 Len=0 SLE=1 SRE=697
11	1.477048	10.143.59.243	10.143.131.227	TCP	58	50122 → 8000 [PSH, ACK] Seq=1 Ack=697 Win=130816 Len=4
12	1.477410	10.143.59.243	10.143.131.227	TCP	12528	50122 → 8000 [PSH, ACK] Seq=5 Ack=697 Win=130816 Len=12474
13	1.483891	10.143.131.227	10.143.59.243	TCP	56	8000 → 50122 [ACK] Seq=697 Ack=5 Win=262080 Len=0
14	1.483944	10.143.59.243	10.143.131.227	TCP	1440	50122 → 8000 [ACK] Seq=12479 Ack=697 Win=130816 Len=1386

tcp.stream eq 0

No.	Time	Source	Destination	Protocol	Length	Info
14	1.483944	10.143.59.243	10.143.131.227	TCP	1440	50122 → 8000 [ACK] Seq=12479 Ack=697 Win=130816 Len=1386
15	1.485276	10.143.131.227	10.143.59.243	TCP	56	8000 → 50122 [ACK] Seq=697 Ack=2777 Win=260736 Len=0
16	1.485277	10.143.131.227	10.143.59.243	TCP	56	8000 → 50122 [ACK] Seq=697 Ack=4163 Win=262144 Len=0
17	1.485332	10.143.59.243	10.143.131.227	TCP	8370	50122 → 8000 [ACK] Seq=13865 Ack=697 Win=130816 Len=8316
18	1.486521	10.143.131.227	10.143.59.243	TCP	56	8000 → 50122 [ACK] Seq=697 Ack=6935 Win=259328 Len=0
19	1.486522	10.143.131.227	10.143.59.243	TCP	56	[TCP Window Update] 8000 → 50122 [ACK] Seq=697 Ack=6935 Win=262144 Len=0
20	1.486522	10.143.131.227	10.143.59.243	TCP	56	8000 → 50122 [ACK] Seq=697 Ack=9707 Win=259328 Len=0
21	1.486522	10.143.131.227	10.143.59.243	TCP	56	[TCP Window Update] 8000 → 50122 [ACK] Seq=697 Ack=9707 Win=262144 Len=0
22	1.486605	10.143.59.243	10.143.131.227	TCP	11142	50122 → 8000 [ACK] Seq=22181 Ack=697 Win=130816 Len=11088
23	1.486825	10.143.131.227	10.143.59.243	TCP	56	8000 → 50122 [ACK] Seq=697 Ack=12479 Win=259328 Len=0
24	1.486826	10.143.131.227	10.143.59.243	TCP	56	[TCP Window Update] 8000 → 50122 [ACK] Seq=697 Ack=12479 Win=262144 Len=0
25	1.486898	10.143.59.243	10.143.131.227	TCP	5598	50122 → 8000 [ACK] Seq=33269 Ack=697 Win=130816 Len=5544

Encryption:

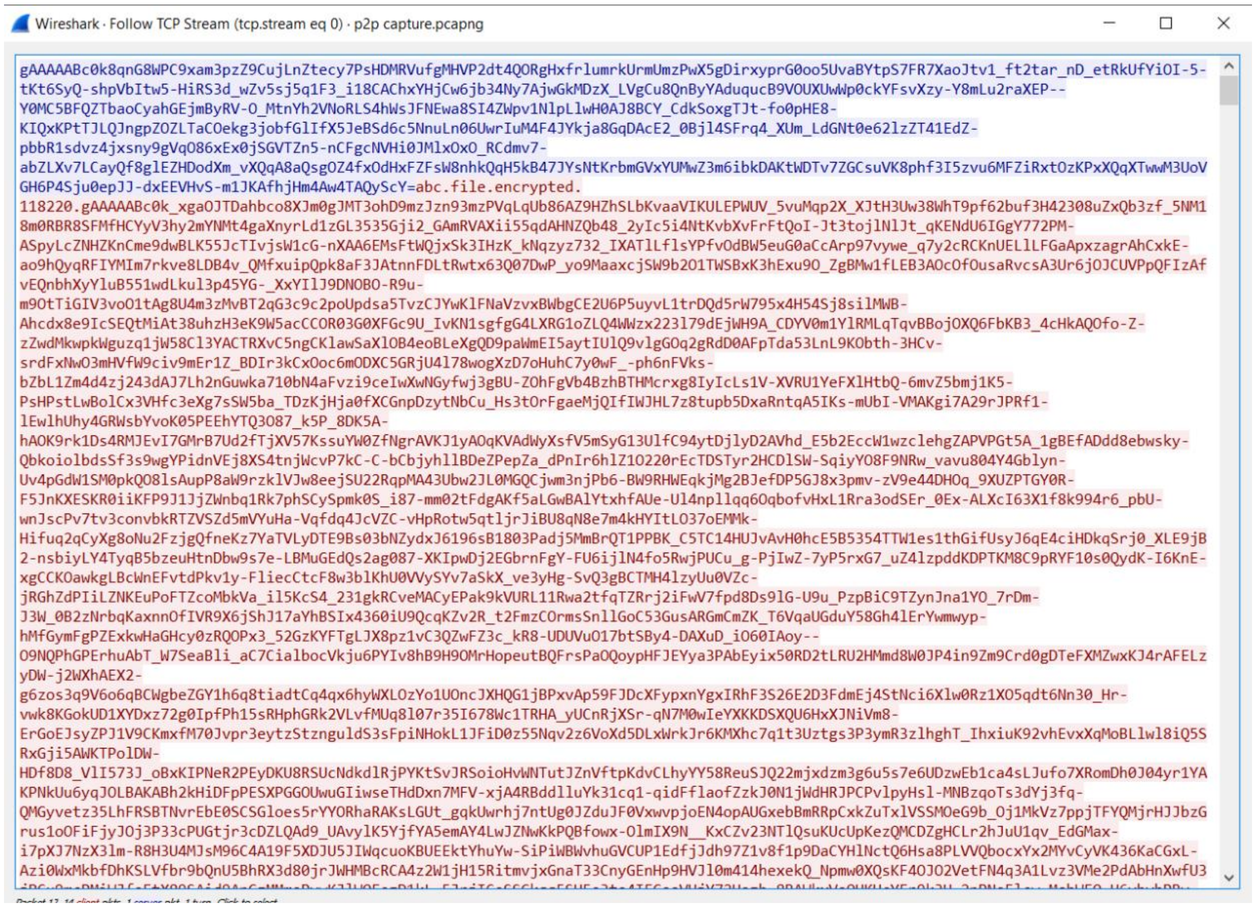
- AES symmetric encryption and RSA asymmetric encryption:



```
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 12514
    Identification: 0xb015 (45077)
  > Flags: 0x4000, Don't fragment
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.143.59.243
    Destination: 10.143.131.227
> Transmission Control Protocol, Src Port: 50122, Dst Port: 8000, Seq: 5, Ack: 697, Len: 12474
v Data (12474 bytes)
    Data: 66696c652e656e63727970746564003131383232300006741…

0030  01 ff d4 fa 00 00 66 69  6c 65 2e 65 6e 63 72 79   ······fi le.encry
0040  70 74 65 64 00 31 31 38  32 32 30 00 67 41 41 41   pted·118 220·gAAA
0050  41 41 42 63 30 6b 5f 78  67 61 4f 4a 54 44 61 68   AABc0k_x gaOJTDah
0060  62 63 6f 38 58 4a 6d 30  67 4a 4d 54 33 6f 68 44   bco8XJm0 gJMT3ohD
0070  39 6d 7a 4a 7a 6e 39 33  6d 7a 50 56 71 4c 71 55   9mzJzn93 mzPVqLqU
0080  62 38 36 41 5a 39 48 5a  68 53 4c 62 4b 76 61 61   b86AZ9HZ hSLbKvaa
0090  56 49 4b 55 4c 45 50 57  55 56 5f 35 76 75 4d 71   VIKULEPW UV_5vuMq
00a0  70 32 58 5f 58 4a 74 48  33 55 77 33 38 57 68 54   p2X_XJtH 3Uw38WhT
00b0  39 70 66 36 32 62 75 66  33 48 34 32 33 30 38 75   9pf62buf 3H42308u
00c0  5a 78 51 62 33 7a 66 5f  35 4e 4d 31 38 6d 30 52   ZxQb3zf_ 5NM18m0R
00d0  42 52 38 53 46 4d 66 48  43 59 79 56 33 68 79 32   BR8SFMfH CYyV3hy2
00e0  6d 59 4e 4d 74 34 67 61  58 6e 79 72 4c 64 31 7a   mYNMt4ga XnyrLd1z
00f0  47 4c 33 35 33 35 47 6a  69 32 5f 47 41 6d 52 56   GL3535Gj i2_GAmRV
0100  41 58 69 69 35 35 71 64  41 48 4e 5a 51 62 34 38   AXii55qd AHNZQb48
```

Encryption Continued:



Wireshark · Follow TCP Stream (tcp.stream eq 0) · p2p capture.pcapng

```
gAAAAABc0k8qnG8WPC9xam3pzZ9CujLnZtecy7PsHDMRVufgMHVP2dt4QORgHxfrlumrkUrmUmzPwX5gDirxyprG0oo5UvaBYtpS7FR7XaoJtv1_ft2tar_nD_etRkUfYiOI-5-
tKt6SyQ-shpVbItw5-HiRS3d_wZv5sj5q1F3_i18CAChxYHjCw6jb34Ny7AjwGkMDzX_LVgCu8QnByYAduqucB9VOUXUwWp0ckYFsvXzy-Y8mLu2raXEP--
Y0MC5BFQZTbaoCyahGEjmByRV-O_MtnYh2VNoRLS4hWsJFNEwa8SI4ZWpv1NlpLlwH0AJ8BCY_CdkSoxgTJt-fo0pHE8-
KIQxKPtTJLQJngpZOZLTaCOekg3jobfGlIfX5JeBSd6c5NnuLn06UwrIuM4F4JYkja8GqDAcE2_0Bjl4SFrq4_XUm_LdGNt0e62lzZT41EdZ-
pbbR1sdvz4jxsny9gVqO86xEx0jSGVTZn5-nCFgcNVHi0JMlxOxO_RCdmv7-
abZLXv7LCayQf8glEZHDodXm_vXQqA8aQsgOZ4fxOdHxFZFsW8nhkQqH5kB47JYsNtKrbmGVxYUMwZ3m6ibkDAKtWDTv7ZGCsuVK8phf3I5zvu6MFZiRxtOzKPxXQqXTwwM3UoV
GH6P4Sju0epJJ-dxEEVHvS-m1JKAfhjHm4Aw4TAQyScY=abc.file.encrypted.
118220.gAAAAABc0k_xgaOJTDahbco8XJm0gJMT3ohD9mzJzn93mzPVqLqUb86AZ9HZhSLbKvaaVIKULEPWUV_5vuMqp2X_XJtH3Uw38WhT9pf62buf3H42308uZxQb3zf_5NM1
8m0RBR8SFMfHCYyV3hy2mYNMt4gaXnyrLd1zGL3535Gji2_GAmRVAXii55qdAHNZQb48_2yIc5i4NtKvbXvFrFtQoI-Jt3tojlNlJt_qKENdU6IGgY772PM-
ASpyLcZNHZKnCme9dwBLK55JcTIvjsW1cG-nXAA6EMsFtWQjxSk3IHzK_kNqzyz732_IXATlLflsYPfvOdBW5euG0aCcArp97vywe_q7y2cRCKnUELlLFGaApxzagrAhCxkE-
ao9hQyqRFIYMIm7rkve8LDB4v_QMfxuipQpk8aF3JAtnnFDLtRwtx63Q07DwP_yo9MaaxcjSW9b2O1TWSBxK3hExu9O_ZgBMw1fLEB3AOcOfOusaRvcsA3Ur6jOJCUVPpQFIzAf
vEQnbhXyYluB551wdLkul3p45YG-_XxYIlJ9DNOBO-R9u-
m9OtTiGIV3voO1tAg8U4m3zMvBT2qG3c9c2poUpdsa5TvzCJYwKlFNaVzvxBWbgCE2U6P5uyvL1trDQd5rW795x4H54Sj8silMWB-
Ahcdx8e9IcSEQtMiAt38uhzH3eK9W5acCCOR03G0XFGc9U_IvKN1sgfgG4LXRG1oZLQ4WWzx223l79dEjWH9A_CDYV0m1YlRMLqTqvBBojOXQ6FbKB3_4cHkAQOfo-Z-
zZwdMkwpkWguzq1jW58Cl3YACTRXvC5ngCKlawSaXlOB4eoBLeXgQD9paWmEI5aytIUlQ9vlgGOq2gRdD0AFpTda53LnL9KObth-3HCv-
srdFxNwO3mHVfW9civ9mEr1Z_BDIr3kCxOoc6mODXC5GRjU4l78wogXzD7oHuhC7y0wF_-ph6nFVks-
bZbL1Zm4d4zj243dAJ7Lh2nGuwka710bN4aFvzi9ceIwXwNGyfwj3gBU-ZOhFgVb4BzhBTHMcrxg8IyIcLs1V-XVRU1YeFXlHtbQ-6mvZ5bmj1K5-
PsHPstLwBolCx3VHfc3eXg7sSW5ba_TDzKjHja0fXCGnpDzytNbCu_Hs3tOrFgaeMjQIfIWJHL7z8tupb5DxaRntqA5IKs-mUbI-VMAKgi7A29rJPRf1-
lEwlhUhy4GRWsbYvoK05PEEhYTQ3O87_k5P_8DK5A-
hAOK9rk1Ds4RMJEvI7GMrB7Ud2fTjXV57KssuYW0ZfNgrAVKJ1yAOqKVAdWyXsfV5mSyG13UlfC94ytDjlyD2AVhd_E5b2EccW1wzclehgZAPVPGt5A_1gBEfADdd8ebwsky-
QbkoiolbdsSf3s9wgYPidnVEj8XS4tnjWcvP7kC-C-bCbjyhllBDeZPepZa_dPnIr6hlZ1O220rEcTDSTyr2HCDlSW-SqiyYO8F9NRw_vavu804Y4Gblyn-
Uv4pGdW1SM0pkQO8lsAupP8aW9rzklVJw8eejSU22RqpMA43Ubw2JL0MGQCjwm3njPb6-BW9RHWEqkjMg2BJefDP5GJ8x3pmv-zV9e44DHOq_9XUZPTGY0R-
F5JnKXESKR0iiKFP9J1JjZWnbq1Rk7phSCySpmk0S_i87-mm02tFdgAKf5aLGwBAlYtxhfAUe-Ul4npllqq6OqbofvHxL1Rra3odSEr_0Ex-ALXcI63X1f8k994r6_pbU-
wnJscPv7tv3convbkRTZVSZd5mVYuHa-Vqfdq4JcVZC-vHpRotw5qtljrJiBU8qN8e7m4kHYItLO37oEMMk-
Hifuq2qCyXg8oNu2FzjgQfneKz7YaTVLyDTE9Bs03bNZydxJ6196sB1803Padj5MmBrQT1PPBK_C5TC14HUJvAvH0hcE5B5354TTW1es1thGifUsyJ6qE4ciHDkqSrj0_XLE9jB
2-nsbiyLY4TyqB5bzeuHtnDbw9s7e-LBMuGEdQs2ag087-XKIpwDj2EGbrnFgY-FU6ijlN4fo5RwjPUCu_g-PjIwZ-7yP5rxG7_uZ4lzpddKDPTKM8C9pRYF10s0QydK-I6KnE-
xgCCKOawkgLBcWnEFvtdPkv1y-FliecCtcF8w3blKhU0VVySYv7aSkX_ve3yHg-SvQ3gBCTMH4lzyUu0VZc-
jRGhZdPIiLZNKEuPoFTZcoMbkVa_il5KcS4_231gkRCveMACyEPak9kVURL11Rwa2tfqTZRrj2iFwV7fpd8Ds9lG-U9u_PzpBiC9TZynJna1YO_7rDm-
J3W_0B2zNrbqKaxnnOfIVR9X6jShJ17aYhBSIx4360iU9QcqKZv2R_t2FmzCOrmsSnllGoC53GusARGmCmZK_T6VqaUGduY58Gh4lErYwmwyp-
hMfGymFgPZExkwHaGHcy0zRQOPx3_52GzKYFTgLJX8pz1vC3QZwFZ3c_kR8-UDUVuO17btSBy4-DAXuD_iO60IAoy--
O9NQPhGPErhuAbT_W7SeaBli_aC7CialbocVkju6PYIv8hB9H9OMrHopeutBQFrsPaOQoypHFJEYya3PAbEyix50RD2tLRU2HMmd8W0JP4in9Zm9Crd0gDTeFXMZwxKJ4rAFELz
yDW-j2WXhAEX2-
g6zos3q9V6o6qBCWgbeZGY1h6q8tiadtCq4qx6hyWXLOzYo1UOncJXHQG1jBPxvAp59FJDcXFypxnYgxIRhF3S26E2D3FdmEj4StNci6Xlw0Rz1XO5qdt6Nn30_Hr-
vwk8KGokUD1XYDxz72g0IpfPh15sRHphGRk2VLvfMUq8l07r35I678Wc1TRHA_yUCnRjXSr-qN7M0wIeYXKKDSXQU6HxXJNiVm8-
ErGoEJsyZPJ1V9CKmxfM70Jvpr3eytzStznguldS3sFpiNHokL1JFiD0z55Nqv2z6VoXd5DLxWrkJr6KMXhc7q1t3Uztgs3P3ymR3zlhghT_IhxiuK92vhEvxXqMoBLlwl8iQ5S
RxGji5AWKTPolDW-
HDf8D8_VlI573J_oBxKIPNeR2PEyDKU8RSUcNdkdlRjPYKtSvJRSoioHvWNTutJZnVftpKdvCLhyYY58ReuSJQ22mjxdzm3g6u5s7e6UDzwEb1ca4sLJufo7XRomDh0J04yr1YA
KPNkUu6yqJOLBAKABh2kHiDFpPESXPGGOUwuGIiwseTHdDxn7MFV-xjA4RBddlluYk31cq1-qidFflaofZzkJ0N1jWdHRJPCPvlpyHsl-MNBzqoTs3dYj3fq-
QMGyvetz35LhFRSBTNvrEbE0SCSGloes5rYYORhaRAKsLGUt_gqkUwrhj7ntUg0JZduJF0VxwvpjoEN4opAUGxebBmRRpCxkZuTxlVSSMOeG9b_Oj1MkVz7ppjTFYQMjrHJJbzG
rus1oOFiFjyJOj3P33cPUGtjr3cDZLQAd9_UAvylK5YjfYA5emAY4LwJZNwKkPQBfowx-OlmIX9N__KxCZv23NTlQsuKUcUpKezQMCDZgHCLr2hJuU1qv_EdGMax-
i7pXJ7NzX3lm-R8H3U4MJsM96C4A19F5XDJU5JIWqcuoKBUEEktYhuYw-SiPiWBWvhuGVCUP1EdfjJdh97Z1v8f1p9DaCYHlNctQ6Hsa8PLVVQbocxYx2MYvCyVK436KaCGxL-
Azi0WxMkbfDhKSLVfbr9bQnU5BhRX3d80jrJWHMBcRCA4z2W1jH15RitmvjxGnaT33CnyGEnHp9HVJl0m414hexekQ_Npmw0XQsKF4OJO2VetFN4q3A1Lvz3VMe2PdAbHnXwfU3
```

References

"A Deep Dive on End-to-End Encryption: How Do Public Key Encryption Systems Work?." *Surveillance Self-Defense*. 29 November 2018. https://ssd.eff.org/en/module/deep-dive-end-end-encryption-how-do-public-key-encryption-systems-work. Accessed on April 12, 2019.

Brent. "Encryption and Decryption in Python." *Nitratine*. 18 December 2018. https://nitratine.net/blog/post/asymmetric-encryption-and-decryption-in-python/. Accessed on April 12, 2019, used Brent comments to resolve file limit size of AES.

"Cryptography 2.6.1." 27 February 2019. *Python Software Foundation (US).* *https://pypi.org/project/cryptography/*. Accessed on April 13, 2019.

"Encryption Software Market by Component (Solution and Services), Application (Disk Encryption, File/Folder Encryption, Communication Encryption, and Cloud Encryption), Deployment Type, Organization Size, Vertical, and Region - Global Forecast to 2022." *MarketsandMarkets.* October 2017. https://www.marketsandmarkets.com/Market-Reports/encryption-software-market-227254588.html. Accessed on April 9, 2019.

“How to: Use Signal for Android.” *Surveillance Self-Defense*. 9 May 2018. https://ssd.eff.org/en/module/how-use-signal-android. Accessed on April 12, 2019.

“How to: Use Signal on iOS.” *Surveillance Self-Defense*. 9 May 2018. https://ssd.eff.org/en/module/how-use-signal-ios. Accessed on April 12, 2019.

“PyOTP - The Python One-Time Password Library.” 2015. *PyOTP*. PyOTP contributors Revision 425a0ec4. https://pyotp.readthedocs.io/en/latest/. Accessed on April 15, 2019.

“PyQRCode 1.2.1.” 19 June 2016. *Python Software Foundation (US).* https://pypi.org/project/PyQRCode/. Accessed on April 17, 2019.

Sneddon, Joey. “Firefox Send is a Free, Encrypted File Sharing Service.” 15 March 2019. *omg! Ubuntu!*. https://www.omgubuntu.co.uk/2019/03/firefox-send-encrypted-file-sharing. Accessed on April 10, 2019.

“tkinter — Python interface to Tcl/Tk.” 10 May 2019. *Python Software Foundation (US).* https://docs.python.org/3/library/tkinter.html. Accessed on April 12, 2019.

Verma, Samer. ISYS 565-Managing Enterprise Networks: “Cloud, Mobile, IT.” 7 May 2019. San Francisco State University, College of Business. Lecture.