/proc/locks Covert Channel Operation
Written by: Benjamin Steenkamer, December 2017

Connection Algorithm

M: Numbers bits of data per transmission

Sender Operations:

```
for HANDSHAKE_TIME seconds do
        Set locks on all M data files and 1 transmission signal file;
        Sleep for HOLD_TIME microseconds;
        Release locks on all M data files and 1 transmission signal file;
        Sleep for HOLD_TIME microseconds;
end for
for HANDSHAKE_TIME seconds do
        Record entries in /proc/locks;
        Sleep for time smaller than HOLD_TIME microseconds;
end for
for each recorded entry of /proc/locks do
        if lock entry's number of toggles >= THRESHOLD then
                Receiver has ACKed = true;
        end if
end for
if receiver has ACKed then
        Send data;
end if
End program;
```

Receiver Operations:

```
for HANDSHAKE_TIME seconds do
        Record entries in /proc/locks;
        Sleep for time smaller than HOLD_TIME microseconds;
end for
for each recorded entry of /proc/locks do
        if lock entry's number of toggles >= THRESHOLD then
                Remember lock's device number;
        end if
end for
if number of sender locks found != M + 1 then
        End program;
end if
for HANDSHAKE_TIME seconds do
        Set lock on ACK data file;
        Sleep for HOLD_TIME microseconds;
        Release lock on ACK data file;
        Sleep for HOLD_TIME microseconds;
end for
```

Receive data;
Print out received data;
End program;

Transmission Algorithm (connection has been established between sender and receiver)

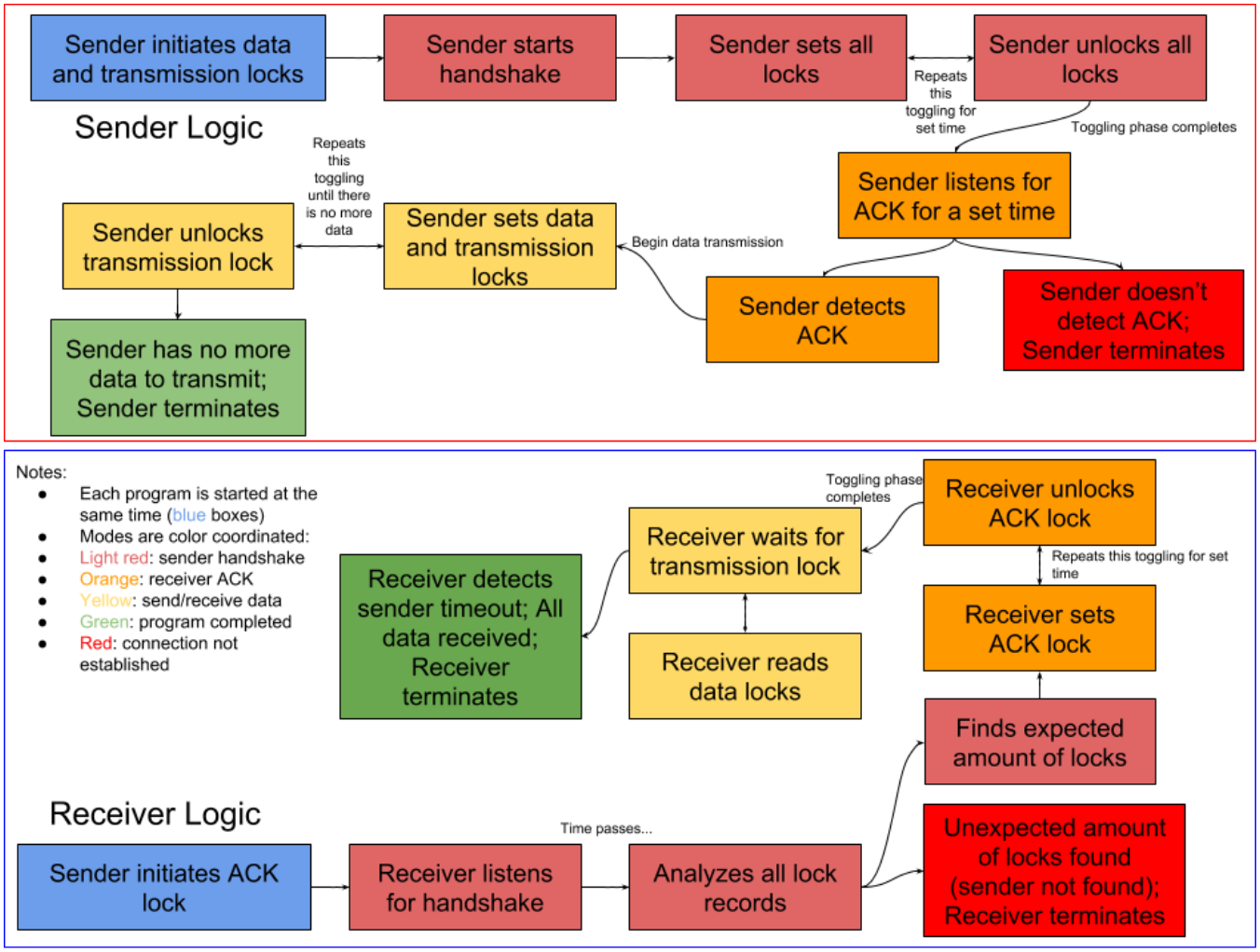
$D_{Send}[N]$, $D_{Recv}[N]$: N transmissions to send and receive; One transmission consists of M bits of data.

Sender Operations (Send data**)**:

```
for i = 0 to N - l do
        for j = 0 to M - 1 do
                if D_Send[i][j] == 1 then
                        Set lock on data file j;
                else then
                        Release lock on data file j;
                end if
        end for
        Set lock on transmission signal file;
        Sleep for TRANSMISSION_TIME microseconds;
        Release lock on transmission signal file;
        Sleep for BETWEEN_TRANS_TIME microseconds;
end for
End connection;
```

Receiver Operations (Receive data):

```
while connection is active do
        Record entries in /proc/locks;
        if transmission lock is present in /proc/locks and haven't read data then
                for j = 0 to M - 1 do
                        if data lock j is present in /proc/locks then
                                D_Recv[i][j] = 1;
                        else then
                                D_Recv[i][j] = 0;
                        end if
                end for
                Data has been read = true;
        else if transmission lock not present in /proc/locks and have read data then
                Data has been read = false;
        else if last transmission time >= TIME_OUT seconds then
                End connection;
        end if
end while
```


Sender initiates data and transmission locks
Sender starts handshake
Sender sets all locks
Repeats this toggling for set time
Sender unlocks all locks
Toggling phase completes
Sender Logic
Repeats this toggling until there is no more data
Sender listens for ACK for a set time
Sender unlocks transmission lock
Sender sets data and transmission locks
Begin data transmission
Sender detects ACK
Sender doesn't detect ACK; Sender terminates
Sender has no more data to transmit; Sender terminates
Notes:
Each program is started at the same time (blue boxes)
Modes are color coordinated:
Light red: sender handshake
Orange: receiver ACK
Yellow: send/receive data
Green: program completed
Red: connection not established
Toggling phase completes
Receiver unlocks ACK lock
Receiver waits for transmission lock
Repeats this toggling for set time
Receiver detects sender timeout; All data received; Receiver terminates
Receiver reads data locks
Receiver sets ACK lock
Finds expected amount of locks
Receiver Logic
Time passes...
Sender initiates ACK lock
Receiver listens for handshake
Analyzes all lock records
Unexpected amount of locks found (sender not found); Receiver terminates