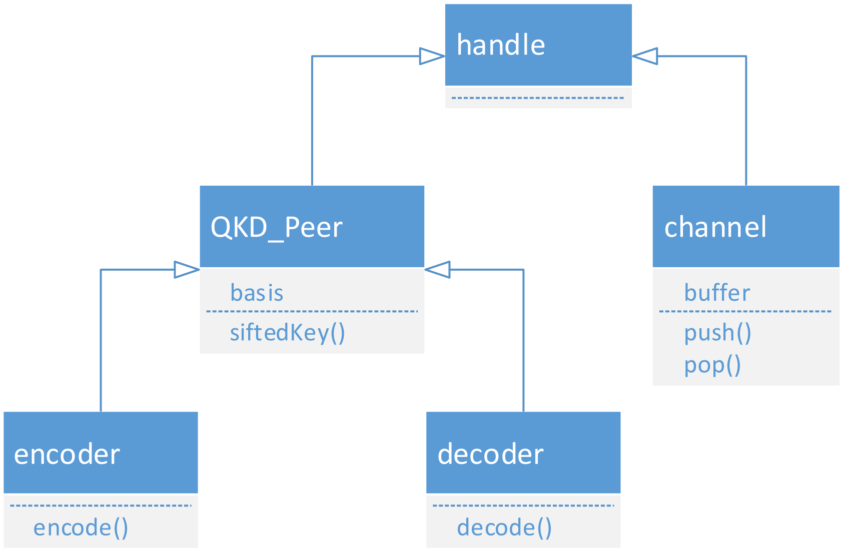
QKD Simulation

Example Main Program



```
% create Alice, channel and Bob
alice=encoder;
c=channel;
bob=decoder;
% encoding by Alice
[S,Alice_bases]=alice.encode(8);
% transport by channel
c.push(S); R=pop(c);
% decoding by Bob
[D,Bob_bases]=bob.decode(R);
% calculation of sifted key
Alice_key = alice.siftedKey( Alice_bases, Bob_bases );
Bob_key = bob.siftedKey( Alice_bases, Bob_bases );
```

QKD_Peer

```
classdef QKD_Peer < handle
    % parent class for Alice and Bob
    properties
        % rectilinear basis and diagonal basis angles
        basis=[['-', '|'] ; ['/', '\']];
        % data bits
        data=[];
    end
    methods
        % calculate and return sifted key
        function [ key ] = siftedKey(obj,
            Alice_bases, Bob_bases )
            key=obj.data(find(Alice_bases==Bob_bases));
        end
    end
end
```

Encoder

```
classdef encoder < QKD_Peer
    % QKP encoder
    methods
        % encoding function
        function [ S,selectedbases ] = encode(obj, n)
            % Generate and array of n random data bits
            obj.data=randi([0 1],1,n);
            % sent photons
            S=[];
            % save basis of each photon
            selectedbases=[];
            for i=1:length(obj.data)
                % randomly select a basis
                b=randi([1 2]);
                selectedbases=[ selectedbases b ];
                % corresponding angles
                A=obj.basis(b,:);
                % map data bit to photon
                S=[S A(obj.data(i)+1) ];
            end
        end
    end
end
```

Example

```
1    1    0    1    1    0    0    1 (Data)

*    *    +    *    *    +    *    + (Bases)

\    \    -    \    \    -    /    | (Photons)
```

Insecure Quantum Channel

(no attack simulated)

```
classdef channel < handle
    % Quantum channel
    properties
        % channel content
        buffer=[];
    end
    methods
        function [ ] = push(obj, S)
            % append bit sequence to buffer
            obj.buffer=[obj.buffer S];
        end
        function [ R ] = pop(obj)
            % return and clear buffer content
            R=obj.buffer;
            obj.buffer=[];
        end
    end
end
```

Decoder

```
classdef decoder < QKD_Peer
    % QKD decoder
    methods
        % decoding function
        function [ D, selectedbases ] = decode(obj,  R)
            % R = received photon angles
            % save basis selected for each photon
            selectedbases=[];
            % Measure the polarity of photons
            obj.data = [];
            for i=1:length(R)
                % randomly select a basis
                b=randi([1 2]);
                selectedbases=[ selectedbases b ];
                % and corresponding angles
                B=obj.basis(b,:);
                if R(i)==B(1)
                    obj.data = [obj.data  0];
                elseif R(i)==B(2)
                    obj.data = [obj.data  1];
                else
                    obj.data = [obj.data randi([0 1])];
                end
            end
            D=obj.data;
        end
    end
end
```

Example

```
|    \    -    -    -    -    /    \ (Alice's Data)

+    *    +    +    +    +    *    * (Alice's bases)

*    +    +    +    +    *    *    + (Bob's bases)

1    0    0    0    0    1    0    0 (Bob's data)
```

Authenticated Channel Handshake

```
find(Alice_bases==Bob_bases)
```

```
3          4          5          7
```

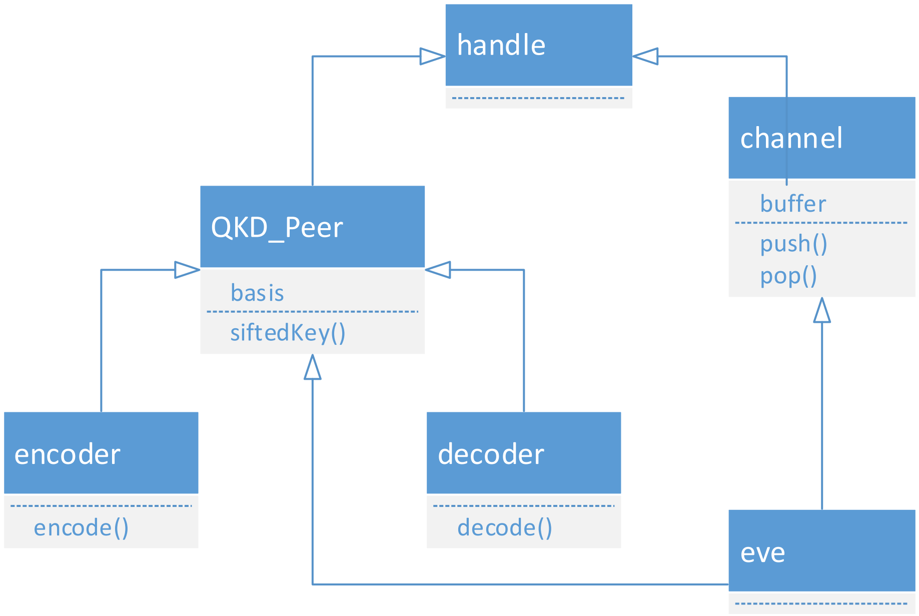
```
% calculation of sifted key
Alice_key = alice.siftedKey( Alice_bases, Bob_bases );

0     0     0     0

Bob_key = bob.siftedKey( Alice_bases, Bob_bases );


0     0     0     0
```

Intercept and Resend Attack Modeling



handle

channel

buffer

push()

pop()

QKD_Peer

basis

siftedKey()

encoder

encode()

decoder

decode()

eve

```
find(Alice_bases==Bob_bases)
1      2      3      6      8

1      1      1      1      0 (Alice's key)

0      1      0      1      0 (Bob's key)

1      1      0      1      0 (Eve's key)
```

Both key establishment and interception success

```
find(Alice_bases==Bob_bases)
3     6     7     8

1     1     1     1 (Alice’s key)

1     1     1     1 (Bob’s key)

1     1     1     1 (Eve’s key)
```