```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using Newtonsoft.Json;
using RabbitMQ.Client;
using RabbitMQ.Client.Framing;
using RabbitMQ.Client.MessagePatterns;
using RabbitRx.Advanced.Subscription;
using RabbitRx.Core.Message;
using RabbitRx.Core.Subscription;
using RabbitRx.Json.Subscription;

namespace Authentication
{
    class Program
    {
        /// <summary>
        /// Connecting to a Broker
        /// </summary>
        static readonly ConnectionFactory Factory = new ConnectionFactory
          { HostName = "66.128.60.46", UserName = "dev", Password = "dev",
          VirtualHost = "/" };
        static readonly IConnection Connection = Factory.CreateConnection();

        static string exchangeName = "deviceTopic";
        static string authenticationQueue = "authentication";
        static string authenticationResponseQueue = "authenticationResponse";

        static void Main(string[] args)
        {
            Start();
        }

        private static CancellationTokenSource _tokenSource;

        /// <summary>
        /// Title: RabbitRx
        /// Author: Ben Johnson
        /// Date: Jan 27, 2015
        /// Availability: https://github.com/bensmind/RabbitRx
        /// </summary>
        private static void Start()
        {
            _tokenSource = new CancellationTokenSource();

            Console.WriteLine("Authentication Service: Press Enter to Start");
            Console.ReadLine();
            Task.Run(() => ConsumeThrottle());
            Console.WriteLine("Press Any Key to Stop");
            Console.ReadLine();
            _tokenSource.Cancel();
            Start();
        }

        static void ConsumeThrottle()
        {
            var channel = Connection.CreateModel();

            channel.BasicQos(0, 50, false);
            channel.ExchangeDeclare(exchangeName, "topic");

            var settings = new BasicProperties()
            {
                ContentType = "application/json",
            };

            var consumer = new JsonObservableSubscription<object>(channel, authenticationQueue, true);

            var throttlingConsumer = new ThrottlingConsumer<RabbitMessage<object>>(consumer, 4);

            throttlingConsumer.Subscribe(message =>
            {
                var session = JsonConvert.DeserializeObject<Model.Authentication>(message.Payload.ToString());

                if (session.Password == Model.Common.Encrypt(session.BaseString + session.Salt))
                {
                    var authenticationResponse = new Model.AuthenticationResponse()
                    {
                        Id = session.Id,
                        Token = session.Token
                    };
                    var bytes = Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(authenticationResponse));
                    channel.BasicPublish(exchangeName, authenticationResponseQueue, settings, bytes);
                    Console.WriteLine("Received:\n");
                    Console.WriteLine("Device: {0}\n", session.Id);
                    Console.WriteLine("Base String: {0}\n", session.BaseString);
                    Console.WriteLine("Share Secret: {0}\n", session.Password);
                    Console.WriteLine("Thread: {0}\n\n", Thread.CurrentThread.GetHashCode());
                }

            }, _tokenSource.Token);

            var start = throttlingConsumer.Start(_tokenSource.Token,
                  TimeSpan.FromSeconds(5));

            start.ContinueWith(t =>
            {
                consumer.Close();
                channel.Dispose();
            });
        }
    }
}

```