**如何为Azure Service Bus和Azure IoT Hub生成SharedAccessSignature**

很多服务在做验证的时候都会用到SharedAccessSignature，例如Azure Service Bus, Azure IoT Hub等。这篇文章主要来讨论一下不同服务下生成的SharedAccessSignature的区别。

首先让我们先看下SharedAccessSignature的格式：

*SharedAccessSignature sig={signature-string}&se={expiry}&skn={policyName}&sr={URL-encoded-resourceURI}*

从上面的格式来看，直观上来看针对不同的服务只要正确指定policyname,resourceURI 就可以计算出相应的SAS。但实际上，对于不同的服务，他们对如何利用秘钥生成signature会存在细微的差别。

接下来我们以Azure Service Bus和Azure IoT Hub进行举例，首先是Azure IoT Hub, 针对如何生成signature [官方文档](https://azure.microsoft.com/en-us/documentation/articles/service-bus-shared-access-signature-authentication/)说明如下：

*{signature} ：An HMAC-SHA256 signature string of the form: {URL-encoded-resourceURI} + "\n" + expiry. Important:* ***The key is decoded from >base64 and used as key*** *to perform the HMAC-SHA256 computation.*

再来看Azure Service Bus 的[官方文档](https://azure.microsoft.com/en-us/documentation/articles/service-bus-shared-access-signature-authentication/)说明：

*The signature for the SAS token is computed using the HMAC-SHA256 hash of a string-to-sign* ***with the PrimaryKey property of an authorization rule****.*

从上面的描述来看，两者的区别在于是否需要对秘钥进行decode。代码分别如下：

*HMACSHA256 hmac = new HMACSHA256(Convert.FromBase64String(key)); // decode the key*

*HMACSHA256 hmac = new HMACSHA256(Encoding.UTF8.GetBytes(key)); // don't decode the key*

如果您是使用C#做开发，可以通过以下方法生成SAS Token:

*class SASTokenGenerator*

*{*

*public static string GetSASToken(string resourceUri, string keyName, string key, TimeSpan ttl)*

*{*

*var expiry = GetExpiry(ttl);*

*string stringToSign = HttpUtility.UrlEncode(resourceUri) + "\n" + expiry;*

*// HMACSHA256 hmac = new HMACSHA256(Convert.FromBase64String(key));* ***for IoT Hub Service***

*HMACSHA256 hmac = new HMACSHA256(Encoding.UTF8.GetBytes(key));* ***for service bus Service***

*var signature = Convert.ToBase64String(hmac.ComputeHash(Encoding.UTF8.GetBytes(stringToSign)));*

*var sasToken = String.Format(CultureInfo.InvariantCulture, "SharedAccessSignature sr={0}&sig={1}&se={2}&skn={3}",*

*HttpUtility.UrlEncode(resourceUri), HttpUtility.UrlEncode(signature), expiry, keyName);*

*return sasToken;*

*}*

*private static string GetExpiry(TimeSpan ttl)*

*{*

*TimeSpan expirySinceEpoch = DateTime.UtcNow - new DateTime(1970, 1, 1) + ttl;*

*return Convert.ToString((int)expirySinceEpoch.TotalSeconds);*

*}*

*}*