Base64编码和AES加密

版本:2018/9/18-1(20:18)

Base64

1、Base64是什么?

> 1. 一种编码方式
> 2. 由于某些系统中只能使用ASCII字符。
> 3. 可以把二进制数据编码为可见的字符数据。

2、Base64末尾采用=结束

> 1. 解析Base64编码时，遇到 = 就知道字符结束了

3、Base64每76个字符增加一个换行符

4、标准的Base64包含64个字符

> 1. A-Z
> 2. a-z
> 3. 0-9
> 4.
>   -

> 5. /
> 6. =

5、Url Safe的Base64编码

> 1. 将 + 替换为 -
> 2. 将 / 替换为 _

Java实现

6、Java中的base64

```
java.util.Base64;
```

7、Java标准版本的base64和Url安全版本的base64

> 1-标准Base64

```
// java 标准Base64编码
String encodeResult = Base64.getEncoder().encodeToString(bytes);
byte[] decodeResult = Base64.getDecoder().decode(encodeResult);
```

> 2-url安全的Base64

```
// java url安全的Base64编码
String urlSafeEncodeResult = Base64.getUrlEncoder().encodeToString(bytes);
byte[] urlSafeDecodeResult = Base64.getUrlDecoder().decode(urlSafeEncodeResult);
```

Android实现

8、Android中的base64

```
android.util.Base64;
```

9、 Android中String的编码和解码

```
// 编码成String
String encodeResult = Base64.encodeToString(bytes, DEFAULT | NO_PADDING | NO_WRAP | CRLF | URL_
// 解码
byte[] decodeResult = Base64.decode(encodeResult, DEFAULT | NO_PADDING | NO_WRAP | CRLF | URL_S
```

10、Android中Base64和java的差异

> 1. Android的Base64编码是默认换行

2. 进行服务器验证的时候，会出现验证失败的情况，这是由于服务器那边的解码不支持换行符模式，所以编码时需要增加flag标志.

11、Android的Base64具有哪些flag?

1. DEFAULT : 默认模式
2. NO_PADDING :过滤结束符=
3. NO_WRAP : 过滤换行符，和CRLF互斥。
4. CRLF : 采用CRLF而不是LF作为换行符，也就是采用Window中的换行符，而不是unix中的换行符。
5. URL_SAFE: 将+,/换成-,_
6. NO_CLOSE

文件

12、文件进行编码

```
File file = new File("/storage/emulated/0/pimsecure_debug.txt");
FileInputStream inputFile  = new FileInputStream(file);
byte[] buffer = new byte[(int) file.length()];
// 1、从文件写入到byte数组中
inputFile.read(buffer);
inputFile.close();
// 2、对byte数组进行编码
encodedString = Base64.encodeToString(buffer, Base64.DEFAULT);
```

13、文件进行解码

```
File desFile = new File("/storage/emulated/0/pimsecure_debug_1.txt");
FileOutputStream fos = new FileOutputStream(desFile);

byte[] decodeBytes = Base64.decode(encodedString.getBytes(), Base64.DEFAULT);
// 将解码而成的byte数组中的数据，写入到文件中。
fos.write(decodeBytes);
fos.close();
```

OkHttp

14、OkHttp中要使用 `NO_WRAP | URL_SAFE` 模式

1. 传输的参数使用android base64进行编码，而字符刚刚好超过76，导致参数增加换行符，换行符为不可见字符.
2. 在OKhttp的参数中为不合法，会直接报错，
3. 需要指定模式为 `NO_WRAP | URL_SAFE` ，过滤换行符，采用URL安全模式。

AES

1、AES是什么?

> 1. 一种高级加密标准（英语：Advanced Encryption Standard，缩写：AES）在密码学中又称Rijndael加密法。
> 2. 美国联邦政府采用的一种区块加密标准。
> 3. 这个标准用来替代原先的DES
> 4. 是 对称密钥加密 中最流行的算法之一。

2、AES工具类:AesUtils

```
import java.security.Provider;
import java.security.SecureRandom;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

/**
 * @function: AES加密算法
 */
public class AesUtils {
    /**
     * @param
     * @return AES加密算法加密
     * @throws Exception
     */
    public static String encrypt(String seed, String key)
            throws Exception {
        byte[] rawKey = getRawKey(key.getBytes());
        byte[] result = encrypt(seed.getBytes("utf-8"), rawKey);
        return toHex(result);
    }

    public static byte[] encryptByte(String seed, String key)
            throws Exception {
        byte[] rawKey = getRawKey(key.getBytes());
        return encrypt(seed.getBytes("utf-8"), rawKey);
    }


    public static String decryptString(byte[] byteData, byte[] byteKey) throws Exception {
        byte[] rawKey = getRawKey(byteKey);
        byte[] result = decrypt(byteData, rawKey);
        return new String(result, "UTF8");
    }


    /***
     * AES加密算法加密
     * @param byteData 数据
     * @param byteKey key
     * @return
     * @throws Exception
     */
    private static byte[] encrypt(byte[] byteData, byte[] byteKey) throws Exception {
        return Ase(byteData, byteKey, Cipher.ENCRYPT_MODE);
    }

    /***
     * AES加密算法解密
     * @param byteData 数据
     * @param byteKey key
     * @return
```

```
     * @throws Exception
     */
    private static byte[] decrypt(byte[] byteData, byte[] byteKey) throws Exception {
        return Ase(byteData, byteKey, Cipher.DECRYPT_MODE);
    }


    /***
     *
     * @param byteData
     * @param byteKey
     * @param opmode
     * @return
     * @throws Exception
     */
    private static byte[] Ase(byte[] byteData, byte[] byteKey, int opmode) throws Exception {
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        SecretKeySpec skeySpec = new SecretKeySpec(byteKey, "AES");
        cipher.init(opmode, skeySpec);
        byte[] decrypted = cipher.doFinal(byteData);
        return decrypted;
    }

    private static byte[] getRawKey(byte[] seed) throws Exception {
        KeyGenerator kgen = KeyGenerator.getInstance("AES");
        SecureRandom sr = null;
        int sdk_version = android.os.Build.VERSION.SDK_INT;
        if (sdk_version > 23) {  // Android  6.0 以上
            sr = SecureRandom.getInstance("SHA1PRNG", new CryptoProvider());
        } else if (sdk_version >= 17) {
            sr = SecureRandom.getInstance("SHA1PRNG", "Crypto");
        } else {
            sr = SecureRandom.getInstance("SHA1PRNG");
        }
        sr.setSeed(seed);
        kgen.init(128, sr); // 192 and 256 bits may not be available
        SecretKey skey = kgen.generateKey();
        byte[] raw = skey.getEncoded();
        return raw;
    }


    public static class CryptoProvider extends Provider {
        /**
         * Creates a Provider and puts parameters
         */
        public CryptoProvider() {
            super("Crypto", 1.0, "HARMONY (SHA1 digest; SecureRandom; SHA1withDSA signature)");
            put("SecureRandom.SHA1PRNG",
                    "org.apache.harmony.security.provider.crypto.SHA1PRNG_SecureRandomImpl");
            put("SecureRandom.SHA1PRNG ImplementedIn", "Software");
        }
    }
```

```
    private static String toHex(byte[] buf) {
        final String HEX = "0123456789ABCDEF";
        if (buf == null)
            return "";
        StringBuffer result = new StringBuffer(2 * buf.length);
        for (int i = 0; i < buf.length; i++) {
            result.append(HEX.charAt((buf[i] >> 4) & 0x0f)).append(
                    HEX.charAt(buf[i] & 0x0f));
        }
        return result.toString();
    }

    private static byte[] toByte(String hexString) {
        int len = hexString.length() / 2;
        byte[] result = new byte[len];
        for (int i = 0; i < len; i++)
            result[i] = Integer.valueOf(hexString.substring(2 * i, 2 * i + 2),
                    16).byteValue();
        return result;
    }
}
```

3、使用工具进行加密和解密

> 需要事先约定好密钥

```
String msg = "msg";
// 生成String
String encryptStr = AesUtils.encrypt(msg, "1234567890ABCDEF");
// 生成byte数组，用于base64编码
byte[] encryptBytes = AesUtils.encryptByte(msg, "1234567890ABCDEF");

// 解密
AesUtils.decryptString(encryptBytes, "1234567890ABCDEF".getBytes());
```

时间戳

1、Android如何获取指定格式的时间戳(YYYYMMDDHH24MISS)?

```
// 时间戳
String timeStamp = new SimpleDateFormat("yyyyMMddHHmmss")
        .format(new Date(System.currentTimeMillis()));
```

随机字符串

1、java中如何生成随机字符串

```java
import java.util.Random;
/**=========================
 * @function: 字符工具
 *  1. 生成随机的字符串
 * @author 6005001819
 * @date 20180918
 *=========================*/
public class CharacterUtils {

    /**
     * @function 生成随机字符串
     *     从62个字符中随机选取一个字符
     * @param length 字符串的长度
     * @return 随机字符串
     */
    public static String getRandomString(int length){
        //定义一个字符串（A-Z，a-z，0-9）即62位；
        String str="zxcvbnmlkjhgfdsaqwertyuiopQWERTYUIOPASDFGHJKLZXCVBNM1234567890";
        //由Random生成随机数
        Random random=new Random();
        StringBuffer sb=new StringBuffer();
        //长度为几就循环几次
        for(int i=0; i<length; ++i){
            //产生0-61的数字
            int number=random.nextInt(62);
            //将产生的数字通过length次承载到sb中
            sb.append(str.charAt(number));
        }
        //将承载的字符转换成字符串
        return sb.toString();
    }

    /**
     * @function 生成随机字符串(两次random，性能低)
     *     随机产生1~3的int数，对应于a-z，A-Z，0-9三种可能
     *     然后再产生随机数，从对应范围中取出字符
     * @param length 字符串的长度
     * @return 随机字符串
     */
    public static String getRandomString2(int length){
        //产生随机数
        Random random=new Random();
        StringBuffer sb=new StringBuffer();
        //循环length次
        for(int i=0; i<length; i++){
            //产生0-2个随机数，既与a-z，A-Z，0-9三种可能
            int number=random.nextInt(3);
            long result=0;
            switch(number){
                //如果number产生的是数字0；
                case 0:
                    //产生A-Z的ASCII码
                    result=Math.round(Math.random()*25+65);
                    //将ASCII码转换成字符
```

```
                    sb.append(String.valueOf((char)result));
                    break;
                case 1:
                    //产生a-z的ASCII码
                    result=Math.round(Math.random()*25+97);
                    sb.append(String.valueOf((char)result));
                    break;
                case 2:
                    //产生0-9的数字
                    sb.append(String.valueOf
                            (new Random().nextInt(10)));
                    break;
            }
        }
        return sb.toString();
    }
}
```

问题汇总

1. Base64是什么?
2. Base64末尾采用=结束
3. Base64每76个字符增加一个换行符
4. 标准的Base64包含64个字符
5. Url Safe的Base64编码
6. Java中的Base64类所在包?
7. Java标准版本的base64和Url安全版本的base64如何使用?
8. Android中的Base64类所在包?
9. Android中String如何进行编码和解码?
10. Android中Base64和java的差异
11. Android的Base64类具有哪些flag?
12. 文件如何进行Base64编码和解码?
13. AES是什么?
14. Android如何获取指定格式的时间戳(YYYYMMDDHH24MISS)?
15. java中如何生成随机字符串?

参考资料

1. Base64编码算法
2. Android Base64编码
3. Android数据加密之Aes加密
4. AesUtils