CPE-03:不得實作未經使用者同意，使行動應用 App 可擅自修改使用者資料的行為，包括在使用者無確認情況下刪除或修改使用者連絡人資料、通話記錄、簡訊資料和多媒體簡訊資料的行為。

安全程式碼範例(Android：Java)

關於個資的部分(連絡人資料、通話記錄、簡訊資料)，開發人員需要檢查是否具備此授權，如果沒有，需要主動向使用者要求允許權限。

```
import android.support.v4.app.ActivityCompat;
import android.Manifest;
import android.content.pm.PackageManager;
import static android.Manifest.permission.*;
public class MainActivity extends AppCompatActivity{
  private static final int MY_REQUEST_ACCESS_CONTACTS = 5;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      int permission = ActivityCompat.checkSelfPermission(this,
              Manifest.permission.READ_CONTACTS);
      if (permission != PackageManager.PERMISSION_GRANTED) {
          //未取得權限，向使用者要求允許權限
          ActivityCompat.requestPermissions( this,
                  new String[]{READ_CONTACTS, WRITE_CONTACTS},
                  MY_REQUEST_ACCESS_CONTACTS);
      }else{
         //已有權限，可進行檔案存取
          readContacts();
      }
  }
        @Override
public void onRequestPermissionsResult(int requestCode,
        String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MY_REQUEST_ACCESS_CONTACTS: {
            // 如果請求被取消的話，傳回的陣列的大小是
            if (grantResults.length > 0
&& grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                // 請求被允許，接下來就可以做一些和連絡人相關的動作
                readContacts();

            } else {
                          //使用者拒絕權限，顯示對話框告知
                    new AlertDialog.Builder(this)
                          .setMessage("必須允許聯絡人權限來使用此功能")
                          .setPositiveButton("OK", null)
                          .show();

            }
            return;
        }

        // 如果有其它的權限，那就需要處理個別的 CASE
    }
  }
}
```

CPF-07:敏感性資料或包含敏感性資料的日誌檔，除非已加密，應避免儲存於與其他行動應用 App 共用或全域可讀寫儲存區域或外部儲存媒體。

不安全程式碼範例(Android：Java)

- *在 API Level 17 Android 4.2 (JELLY_BEAN_MR1)之前的版本，可以允許開發人員的 APP 將資料輸出到 APP 公有區中，下列範例即為將資料輸出到公有區中，並設其權限為共同寫入。*

```
String FILENAME = "hello_file";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_WORLD_READABLE, );
fos.write(string.getBytes());
fos.close();
```

安全程式碼範例(Android：Java)

- *下列範例為安全的範列，此範例改將資料輸出到私有區中，並設其權限為私有，以進一步確保資訊安全。*

```
String FILENAME = "hello_file";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

CPF-08:需注意記憶體暫存的敏感性資料，如固定金鑰與密碼的安全性，當不需要時或於一定合理期間應強制清除或使用於一定期間就失效的可變量金鑰替代。

安全程式碼範例(Android：Java)

因為 Java 的 Garbage Collection(GC)的特性是，開發人員無法確認何時資料會真的被清空，而 String 是 immutable，基本上無法主動清空，就算指向 null，還是需要等待 GC 完成後，它才不會存在於記憶體中，最有效的方式為改以 byte array 來存放，在使用完之後，直接將每一個 byte 值覆寫為 0，如此一來立刻就能將資料從記憶體中清空，防止惡意人士從記憶體中盜取。

```
• SecretKey key = KeyGenerator.getInstance("DES").generateKey();
  byte[] data = key.getEncoded();//處理完相關作業之後立即清空，以避免存在於 RAM

for (byte oneByte:data){     oneByte=0;
}
```

CPF-17:行動應用 App 有提供標準的設定參數檔函數，但由於駭客取得安裝檔之後，非常容易就可以修改這些標準的參數檔，於是為了加強安全層級，應用程式的設定參數，建議放在安全的地方，例如編譯至程式碼中，或者進行加密。

安全程式碼範例(Android：Java)

此範例說明如何將加密儲存於 property file 的密碼來和使用者輸入的密碼做安全的比對

```
public class MainActivity extends ActionBarActivity {
    String flag = "com.ereach.helloworld";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


String encryptPassword = loadPassword();
//因此加密方式為不可逆，如需要比較密碼時，請將使用輸入的密碼加密後比對
// if(encrypt(userInputPassword).equals(encryptPassword))
    }
public static String encrypt(String s){
        MessageDigest sha = null;
        try{
            sha = MessageDigest.getInstance("SHA-256");
            sha.update(s.getBytes());
        }catch(Exception e){
            e.printStackTrace();
            return "";
        }
        return byte2hex(sha.digest());
    }
    private static String byte2hex(byte[] bytes){
        String hs="";
        String stmp="";
        for (byte myByte: bytes){
            stmp=(java.lang.Integer.toHexString(myByte & 0XFF));
            if (stmp.length()==1) hs=hs+"0"+stmp;
            else hs=hs+stmp;
        }
        return hs.toUpperCase();
    }
       //讀取資料的方法
private String loadPassword(){
//建構 properties 物件
Properties properties = new Properties();
    try{
        FileInputStream stream = this.openFileInput("music.cfg");
        //讀取兩個檔的內容
        properties.load(stream);
        properties.load(stream1);


      }catch(FileNotFoundException e)
      {
      return;
      }catch(IOException e){
      return;
      }
       return String.valueOf(properties.get("password"));
}
}
```

CPL-01:行動應用 App 應設計並實作適當身分認證機制，並依使用者身分授權，以防止敏感資料被非授權人員存取。

安全程式碼範例(Android：Java)

```
• //定義使用 ACCOUNT_MANAGER 與 INTERNET 權限
<manifest ... >
    <uses-permission android:name="android.permission.ACCOUNT_MANAGER" />
    <uses-permission android:name="android.permission.INTERNET" />
    ...
</manifest>
//實作 CallBack
AccountManager am = AccountManager.get(this);
Bundle options = new Bundle();
am.getAuthToken(
    myAccount_,                     // Account retrieved using getAccountsByType()
    "Manage your tasks",            // Auth scope
    options,                        // Authenticator-specific options
    this,                           // Your activity
    new OnTokenAcquired(),          // Callback called when a token is successfully acquired
    new Handler(new OnError()));    // Callback called if an error occurs

• private class OnTokenAcquired implements AccountManagerCallback<Bundle> {
    @Override
    public void run(AccountManagerFuture<Bundle> result) {
        ...
        Intent launch = (Intent) result.getResult().get(AccountManager.KEY_INTENT);
        if (launch != null) {
            startActivityForResult(launch, 0);
            return;
        }
    }
}
//獲取 token
private class OnTokenAcquired implements AccountManagerCallback<Bundle> {
    @Override
    public void run(AccountManagerFuture<Bundle> result) {
        // Get the result of the operation from the AccountManagerFuture.
        Bundle bundle = result.getResult();

        // The token is a named value in the bundle. The name of the value
        // is stored in the constant AccountManager.KEY_AUTHTOKEN.
        token = bundle.getString(AccountManager.KEY_AUTHTOKEN);
        ...
    }
}
//呼叫 OAuth2 服務
URL url = new URL("https://www.googleapis.com/tasks/v1/users/@me/lists?key=" +
your_api_key);
URLConnection conn = (HttpURLConnection) url.openConnection();
conn.addRequestProperty("client_id", your client id);
conn.addRequestProperty("client_secret", your client secret);
conn.setRequestProperty("Authorization", "OAuth " + token);
```

CPM-03:行動應用 App 連線使用交談識別碼，應實作具備逾時失效(Session time-out)機制。

安全程式碼範例(Android：Java)

可以在 Activity 內加上下面二個方法來達成 local-session-timeout

```
@Override public boolean dispatchTouchEvent(MotionEvent ev) {
        lastActivity = new Date().getTime();  //取得最後動作的時間
        return super.dispatchTouchEvent(ev);
}
@Override public void onResume() {
        long now = new Date().getTime();
        if (now - lastActivity > 300000) {     //超過五分鐘則自動登出
        // startActivity and force logon     } }
```

CPM-05:行動應用程式應使用憑證綁定(Certificate Pinning)方式驗證並確保連線之伺服器為行動應用程式開發人員所指定。

安全程式碼範例(Android：Java)

假設你有由知名 CA 頒發的證書的 Web 伺服器，你可以用簡易程式碼安全要求：

```
URL url = new URL ( "https://wikipedia.org" );
URLConnection urlConnection = url . openConnection ();
InputStream in = urlConnection . getInputStream ();
copyInputStreamToOutputStream ( in , System . out );
```

如果你想客製設定 HTTP 請求，可以改使用 HttpsURLConnection 或是強制轉型為 HttpURLConnection

```
public class KeyPinStore {

    private static KeyPinStore instance = null;
    private SSLContext sslContext = SSLContext.getInstance("TLS");

    public static synchronized KeyPinStore getInstance() throws CertificateException,
IOException, KeyStoreException, NoSuchAlgorithmException, KeyManagementException{
        if (instance == null){
            instance = new KeyPinStore();
        }
        return instance;
    }

    private KeyPinStore() throws CertificateException, IOException, KeyStoreException,
NoSuchAlgorithmException, KeyManagementException{
        // Load CAs from an InputStream
        // (could be from a resource or ByteArrayInputStream or ...)
        CertificateFactory cf = CertificateFactory.getInstance("X.509");
        // randomCA.crt should be in the Assets directory
        InputStream caInput = new
BufferedInputStream(MainActivity.context.getAssets().open("randomCA.crt"));
        Certificate ca;
        try {
            ca = cf.generateCertificate(caInput);
            System.out.println("ca=" + ((X509Certificate) ca).getSubjectDN());
        } finally {
            caInput.close();
        }

        // Create a KeyStore containing our trusted CAs
        String keyStoreType = KeyStore.getDefaultType();
        KeyStore keyStore = KeyStore.getInstance(keyStoreType);
        keyStore.load(null, null);
        keyStore.setCertificateEntry("ca", ca);

        // Create a TrustManager that trusts the CAs in our KeyStore
        String tmfAlgorithm = TrustManagerFactory.getDefaultAlgorithm();
```

```
        TrustManagerFactory tmf = TrustManagerFactory.getInstance(tmfAlgorithm);
        tmf.init(keyStore);

        // Create an SSLContext that uses our TrustManager
        // SSLContext context = SSLContext.getInstance("TLS");
        sslContext.init(null, tmf.getTrustManagers(), null);
    }

    public SSLContext getContext(){
        return sslContext;
    }
}
```

CPO-03:行動應用 App 程式碼應運用人工或工具使之增加複雜度，並輔以限制除錯器使用、反追蹤、二進位剝離等措施，使惡意人士使用逆向工程方法分析程式碼難度增加。

安全程式碼範例(Android：Java)

使用 Google 提供的 ProGuard，來將程式碼進行混淆。
在 bulid.gradle 檔案中，把 minifyEnabled 改成 true 即可啟動

```
buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
```

CPQ-03:行動應用 App 提供使用者輸入值儘量可以參數化(Query parameterization)。

安全程式碼範例

Language - Library	Parameterized Query
Java - Standard	• String custname = request.getParameter("customerName"); • String query = "SELECT account_balance FROM user_data WHERE user_name = ? "; • **PreparedStatement pstmt = connection.prepareStatement(query);** • **pstmt.setString(1, custname);** • ResultSet results = pstmt.executeQuery( );

Java - Hibernate	• //HQL
	• Query safeHQLQuery = session.createQuery("from Inventory where productID=:productid");
	• safeHQLQuery.setParameter("productid", userSuppliedParameter);
	• //Criteria API
	• String userSuppliedParameter = request.getParameter("Product-Description"); // This should REALLY be validated too
	• // perform input validation to detect attacks
	• Inventory inv = (Inventory) session.createCriteria(Inventory.class).add
	• (Restrictions.eq("productDescription", userSuppliedParameter)).uniqueResult();

.NET/C#	• String query = "SELECT account_balance FROM user_data WHERE user_name = ?";
	• try {
	• OleDbCommand command = new OleDbCommand(query, connection);
	• **command.Parameters.Add(new OleDbParameter("customerName", CustomerName Name.Text));**
	• OleDbDataReader reader = command.ExecuteReader();
	• // …
	• } catch (OleDbException se) {
	• // error handling
	• }

ASP.NET	• string sql = "SELECT * FROM Customers WHERE CustomerId = @CustomerId"; • **SqlCommand command = new SqlCommand(sql);** • **command.Parameters.Add(new SqlParameter("@CustomerId", System.Data.SqlDbType.Int));** • command.Parameters["@CustomerId"].Value = 1;
Ruby - ActiveRecord	• **# Create** • Project.create!(:name => 'owasp') • **# Read** • Project.all(:conditions => "name = ?", name) • Project.all(:conditions => { :name => name }) • Project.where("name = :name", :name => name) • **# Update**

Language - Library	Parameterized Query
	• project.update_attributes(:name => 'owasp') • **# Delete** • Project.delete(:name => 'name')
Ruby	• insert_new_user = db.prepare "INSERT INTO users (name, age, gender) VALUES (?, ? ,?)" • insert_new_user.execute 'aizatto', '20', 'male'
PHP - PDO	• $stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value)"); • **$stmt->bindParam(':name', $name);** • **$stmt->bindParam(':value', $value);**
Cold Fusion	• <cfquery name = "getFirst" dataSource = "cfsnippets"> • **SELECT * FROM #strDatabasePrefix#_courses WHERE intCourseID =** • **<cfqueryparam value = #intCourseID# CFSQLType = "CF_SQL_INTEGER">** • </cfquery>
Perl - DBI	• my $sql = "INSERT INTO foo (bar, baz) VALUES (?, ?)"; • **my $sth = $dbh->prepare($sql);** • **$sth->execute($bar, $baz);**

CPQ-05:行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 SQL injection 之字串。

不安全程式碼範例(Android：Java)

SQL 語法直接使用+的法式將使用者的輸入串連起來

```
tring commandString = "INSERT INTO mytable (idno, name) VALUES
'"+userInputIdno+"','"+userInputName+"'";
try {
    mSampleDB.execSQL(commandString);
}catch (SQLException e) {
    ......
} finally {
    sqlStmt.close();
}
```

安全程式碼範例(Android：Java)

使用參數化方式避免使用者的惡意輸入

```
String commandString = "INSERT INTO mytable (idno, name) VALUES (?, ?)";
//使用參數化的SQLiteStatement以防止SQL Injection
SQLiteStatement sqlStmt = mSampleDB.compileStatement(commandString);
sqlStmt.bindString(1, userInputIdno);
sqlStmt.bindString(2, userInputName);
try {
    sqlStmt.executeInsert();
}catch (SQLException e) {
    ......
} finally {
    sqlStmt.close();
}
```

CPQ-08:行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 XML Injection 之字串。

不安全程式碼範例(網頁主機：Server .NET C#)

使用字串加減維護 XML

```
xmlstr = "<username>" + "foo<" + "</username>";
```

安全程式碼範例(Android；Java)

```
libxml_disable_entity_loader(true);
```

安全程式碼範例(網頁主機：Server .NET C#)

使用元件維護 XML

```
XmlNode node = doc.CreateNode("username");
node.Value = "foo<";
```

Android-01:Android 版本行動應用 App 應謹慎實作 Intents，避免資訊不慎外洩遭惡意運用。

不安全程式碼範例(Java)

使用不指定接收 Class 的 Intent 來傳遞機密資訊，是不安全的也可能會被攔截，而使得資料外洩。

```
Intent intent = new Intent();
intent.putExtra("RESULT", "Sensitive Info");
startActivity(intent);
```

安全程式碼範例(Java)

當使用 Intent 來傳遞資料給 Activity, 為維持資料安全，請使用明文指定接收 Class 的名稱。

```
//明文指定接收此 intent 的類別，以避免其它程式攔截取得資訊，或改變程式流程
Intent intent = new Intent(this, LoginActivity.class);
intent.putExtra("RESULT", "Not Sensitive Info");
startActivity(intent);
```

Android-03:Android 版行動應用 App 實作廣播(Broadcast intent)應設定權限，避免被惡意行動應用 App 偽造元件。

安全程式碼範例(Java)

Sender.java

```
// 發送廣播事件用的 Action 名稱
public static final String BROADCAST_ACTION =
        "org.appsec.broadcast.action.MYBROADCAST";
...
// 建立準備發送廣播事件的 Intent 物件
Intent intent = new Intent(BROADCAST_ACTION);
// 如果需要的話，也可以設定資料到 Intent 物件
intent.putExtra("UserName", userName);
intent.putExtra("UserId", userId);
// 發送廣播事件
sendBroadcast(intent);
```

Recevier.java

```
// 繼承自 BroadcastReceiver 的廣播接收元件
public class MyBroadcastReceiver extends BroadcastReceiver {
    // 接收廣播後執行這個方法

    @Override
    public void onReceive(Context context, Intent intent) {
        // 讀取包含在 Intent 物件中的資料
        String name = intent.getStringExtra("UserName");
        String id = intent. getStringExtra ("UserId");
        ...
        // 因為這不是 Activity 元件，需要使用 Context 物件的時候，
        // 不可以使用「this」，要使用參數提供的 Context 物件
        Toast.makeText(context, message, Toast.LENGTH_SHORT).show();
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application … >
        <!-- 使用 receiver 標籤，名稱設定廣播接收元件類別名稱 -->
        <receiver android:name="MyBroadcastReceiver">
            <intent-filter>
                <!-- 使用 Action 名稱設定接收的廣播事件 -->
                <action android:name=
                    " org.appsec.broadcast.action.MYBROADCAST" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

Android-07: Android 版行動應用 App 應謹慎實作 Content Providers，避免因權限過高或未驗證資料目的地與來源，遭惡意程式注入式攻擊。

不安全程式碼範例(Java)

SQL 語法直接使用+的法式將使用者的輸入串連起來

```
@Override
public Cursor query(Uri uri, String[] projection, String selection, String[]
selectionArgs, String sortOrder) {
        String groupId = getFromUserInput();//假設有此方法可讀入畫面上使用者的輸入值
        SQLiteDatabase mDB = mHelper.getWritableDatabase();
        Cursor out = mDB.rawQuery("SELECT * FROM product where group_id="+groupId, null);
        out.moveToFirst();
        return out;
}
```

安全程式碼範例(Java)

使用參數化方式，將 SQL 語法和使用者的輸入分離

```
@Override
public Cursor query(Uri uri, String[] projection, String selection, String[]
selectionArgs, String sortOrder) {
        String groupId = getFromUserInput();//假設有此方法可讀入畫面上使用者的輸入值
        SQLiteDatabase mDB = mHelper.getWritableDatabase();
        Cursor out = mDB.rawQuery("SELECT * FROM product where group_id=?", new
String[]{groupId});
        out.moveToFirst();
        return out;
}
```

Android-09:行動應用 App 應實作過濾使用者輸入及伺服器端傳入資料中易導致 Intent Injection 之字串。

不安全程式碼範例(Java)

```
TextView tv = (TextView) findViewById(R.id.textview);
InputStreamReader isr = null;
char[] text = new char[1024];
int read;
try {
    String urlstr = getIntent().getStringExtra("WEBPAGE_URL");
    URL url = new URL(urlstr);
    isr = new InputStreamReader(url.openConnection().getInputStream());
      //此處並無防止"file://..." 這種型式的。在此情形之下，本機端的檔案會被開啟，並且將內容秀在 TextView 中，而非原先預請的遠端網頁

    while ((read=isr.read(text)) != -1) {
        tv.append(new String(text, 0, read));
    }
} catch (MalformedURLException e) { ...
```

安全程式碼範例(Java)

```
TextView tv = (TextView) findViewById(R.id.textview);
InputStreamReader isr = null;
char[] text = new char[1024];
int read;
try {
    String urlstr = getIntent().getStringExtra("WEBPAGE_URL");
    URL url = new URL(urlstr);
    //加入明確檢查是否是使用 http 或 https protocol, 如不是則產生例外
    String prot = url.getProtocol();
    if (!"http".equals(prot) && !"https".equals(prot)) {
        throw new MalformedURLException("invalid protocol");
    }
    isr = new InputStreamReader(url.openConnection().getInputStream());
    while ((read=isr.read(text)) != -1) {
        tv.append(new String(text, 0, read));
    }
} catch (MalformedURLException e) { ..
```

Server-01 與行動應用 App 連接之所有後端服務伺服器(包含網頁、資料庫及中介等)作業系統應有效強化及進行安全設定配置，並持續進行安全性程式修補。

不安全程式碼範例(網站主機：Server)

例如過時 OpenSSL 版本 OpenSSL/1.0.1e 與太多資訊洩漏。

```
telnet ---.---.---.tw 80
Trying 0.0.0.0……......
Connected to ---.---.---.tw.
Escape character is '^]'.
HEAD / HTTP/1.0
```

```
HTTP/1.1 200 OK
Date: Sat, 06 Aug 2016 01:57:04 GMT
Server: Apache/2.2.29 (Unix) mod_ssl/2.2.29 OpenSSL/1.0.1e-fips DAV/2 PHP/5.3.29
Last-Modified: Thu, 30 Jun 2016 06:22:49 GMT
ETag: "bc16f1-7f8-53678e6296040"
Accept-Ranges: bytes
Content-Length: 2040
Connection: close
Content-Type: text/html


遺失與主機的連線。
```

安全程式碼範例(網站主機：Server)

較少資訊洩漏

```
telnet www.------.com.tw 80
HTTP/1.0 200 OK
Date: Sat, 30 Jul 2016 05:52:56 GMT
P3P: policyref="http://info.------.com/w3c/p3p.xml", ... Cache-Control: max-age=3600, public
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8
Server: ATS
```

安全程式碼範例(網站主機：Server .NET)

啟用 HSTS 機制

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <rewrite>
            <rules>
                <rule name="HTTP to HTTPS redirect" stopProcessing="true">
                    <match url="(.*)" />
                    <conditions>
                        <add input="{HTTPS}" pattern="off" ignoreCase="true" />
                    </conditions>
                    <action type="Redirect" url="https://{HTTP_HOST}/{R:1}"
                        redirectType="Permanent" />
                </rule>
            </rules>
            <outboundRules>
                <rule name="Add Strict-Transport-Security when HTTPS" enabled="true">
                    <match serverVariable="RESPONSE_Strict_Transport_Security"
                        pattern=".*" />
                    <conditions>
                        <add input="{HTTPS}" pattern="on" ignoreCase="true" />
                    </conditions>
                    <action type="Rewrite" value="max-age=31536000" />
                </rule>
            </outboundRules>
        </rewrite>
    </system.webServer>
</configuration>
```

Server-06:網頁伺服器需預防網頁掛馬及點擊劫持攻擊。

安全程式碼範例(網頁主機：Server)

增加 X-Frame-Options

```
X-Frame-Options: DENY

X-Frame-Options: SAMEORIGIN

X-Frame-Options: ALLOW-FROM https://example.com/
```

DENY

The page cannot be displayed in a frame, regardless of the site attempting to do so.

SAMEORIGIN

The page can only be displayed in a frame on the same origin as the page itself.

ALLOW-FROM *uri*

The page can only be displayed in a frame on the specified origin.

Server-10:參考 CPG-01~CPG-03 實作 TLS 伺服器端設定。

不安全程式碼範例(主機：Server)

OpenSSL 1.01f(含以前) 版本
2048 位元以下、逾期、失效、根憑證失效、掛失等憑證。

安全程式碼範例(主機：Server)

OpenSSL 1.01g(含之後) 版本
2048 位元(含)以上，合法有效憑證。