DKBA

华为技术有限公司内部技术规范

DKBA 7475-2016.02

密码算法应用规范



2016年2月16日发布 2016年2月16日实施

华为技术有限公司

Huawei Technologies Co., Ltd.

版权所有 侵权必究

All rights reserved

**修订声明Revision declaration**

本规范的相关系列规范或文件：

《密钥管理安全规范》

《身份和访问管理安全设计规范》

相关国际规范或文件一致性：

无

替代或作废的其它规范或文件：

无

相关规范或文件的相互关系：

互为补充

历次修订内容列表：

|  |  |  |
| --- | --- | --- |
| **修订版本** | **修订原因** | 修订说明 |
| V1.1版本对V1.0的修订内容 | 为对齐IETF规范（RFC 7465），禁止RC4用于TLS协议 | 1、在规则1.2标题中明确禁止使用不安全密码算法；  2、规则说明部分增加：根据IETF发布的RFC 7465规范，禁止RC4用于TLS协议； |
| 规定使用推荐的随机数发生器时设置种子的注意事项 | 1、规则2.1.1说明部分增加：随机数接口如需设置种子，则必须仅在程序初始化时设置一次；  2、规范说明部分修改：对随机数发生器或接口在设置种子方面做出明确规定； |
| 直接采用FIPS 186-4中对DSA算法生成数字签名的相关要求 | 1、修改规则2.3.2.1标题，使其直接符合NIST规范中对DSA算法的描述；  2、修改规则说明部分，使其符合NIST规范中对DSA算法的描述； |
| 新增有关RSA数字签名中使用更安全的PSS填充方式的建议 | 为对齐德电有关RSA数字签名相关规定，增加建议2.3.2.4 |
| 新增有关DH密钥协商过程中防止简单替换攻击的规则 | 为防止DH密钥协商过程中被实施简单替换攻击，增加规则2.3.3.1 |
| 提供scrypt算法推荐使用参数、规定PBKDF2、scrypt算法输出长度 | 1、修改规则2.4.1标题为：对口令进行单向哈希时，必须使用基于口令的密钥导出算法  2、新增有关口令单项哈希算法scrypt的推荐使用参数；  3、规定PBKDF2、scrypt算法输出长度应该不小于256比特 |
| 明确计算MAC时所需密钥长度 | 在规则2.5.2中增加计算MAC时，对密钥长度做规定 |
| 增加CTR模式实际操作建议 | “附录——>常见工作模式介绍——>计数器模式”中描述，使用CTR模式加密时需保证计数器各不相同，这里提供为了达到这一使用要求可以采用的实际操作建议 |
| 参考专家评审意见修定 | 强烈建议在有条件的情况下使用SHA256或更安全的哈希算法 |
| 参考专家评审意见修定 | 对SHA-3系列哈希算法做牵引 |
| 参考专家评审意见修定 | 列出了典型算法的安全强度及各安全强度对应的生命周期 |
| 参考专家评审意见修定 | 原FIPS 186-2, [X9.31]和ANS [X9.62]等标准中规定的随机数生成算法将在2015年12月31日后不再被允许使用，本规范不推荐使用这些随机数生成算法。推荐使用修正后的SP 800-90A中的HASH\_DRBG、HMAC\_DRBG、CTR\_DRBG三种随机数生成器，禁用NIST SP 800-90A中的DualEC-DRBG生成器。 |
| 参考专家评审意见修定 | SSL3.0、TLS1.0、SSH2.0协议中使用CBC模式容易遭受攻击导致明文信息泄露。因此在这些协议中应避免使用CBC模式。说明可以使用的条件并提示风险。 |
| 参考专家评审意见修定 | 按照PKCS#1 V2.1规定，明确使用OAEP填充方式应该遵循的条件 |
| 参考专家评审意见修定 | 修改规则“使用非对称算法时，加密和签名要使用不同的密钥对”的说明部分 |
| 参考专家评审意见修定 | PBKDF2仅在用于口令哈希时对其输出长度规定应该不小于256bits |
| 参考专家评审意见修定 | 针对产品各种应用场景下的不同性能要求对PBKDF2迭代次数进行了规定 |
| V1.2版本对V1.1的修订内容 | 参考专家评审意见修订 | 内部规范库地址在下版中要体现 |
| 对发往它国的产品使用国标密码算法做出限定 | 新增规则：客户有明确要求时应优先选客户认可的密码算法，客户没有明确要求时优先择国际标准密码算法，国家标准算法（如中国国密算法）在发往它国的产品中应默认关闭 |
| 参考专家评审意见修订 | 对强密码算法的描述增加限定 |
| 参考专家评审意见修订 | 非对称加密实例指导中对于应用场景的描述增加限定 |
| 根据业界最新进展更新算法列表 | 2015年8月份SHA-3已经被正式标准化，所以对SHA-3的表述也做相应的调整  Chacha20流加密算法已经在RFC 7539被IETF引用为参考消息，从业界实践角度看，谷歌、苹果已使用该算法，OpenSSL 1.1.0开始支持该算法，其安全性也已经过众多评估，因此对于流加密，新增推荐chacha20 |
| 根据产品使用随机数时遇到的问题及解决方案修订规则说明部分 | 修订规则：密码算法中使用到的随机数必须是密码学意义上的安全随机数 |
| 新增优先使用模式GCM | 修订规则：使用分组密码算法时，应优先选择GCM或CBC模式 |
| 明确对于不同加密模式对于IV的要求是不可预测还是唯一，长度分别是多少 | 修订规则：分组密码算法使用到的IV值，必须满足具体加密模式对IV值的要求 |
| 新增可选流机密算法chacha20、chacha20-poly1305 | 修订规则：使用AES分组密码算法来代替RC4流密码算法 |
| 对齐德电需求中关于对称加密算法填充的相关要求 | 新增建议：使用分组加密算法时，填充方式建议选择CMS-Padding 或ISO-Padding |
| 从《密钥管理安全规范》移到《密码算法应用规范》中 | 新增规则：非信任环境内进行密钥协商，必须确保双方身份的合法性 |
| 跟进业界进展 | 建议：密钥协商时推荐支持完全前向保密特性 |
| 参考密码专家组评审的安全强度修订 | 更新PBKDF2迭代次数要求 |
| 跟进业界进展 | 新增规则：禁止使用SSL2.0、SSL3.0协议，仅遗留使用TLS1.0 |
| 跟进业界进展 | 新增规则： SSH协议中禁止使用CBC模式 |
| 参考专家评审意见修订 | 敏感数据非对称加解密示例指导中说明增加RSA在加密大量数据时如何处理 |

历次修订主要参与人员：

|  |  |  |  |
| --- | --- | --- | --- |
| Specification No | Prepared by | Reviewed by | Revision |
| DKBA 7475-2014.01 | 网络安全能力中心：  邹国雄/ 00214764，卢龙萍/90006842，罗东/ 00256178 | 谢尔德实验室：鲍丰/00236039，徐磊/ 00179725  网络安全能力中心：郎风华/ 00134482，何纲/ 00152106，冯凯锋/ 00210675，官日岗/ 00118884，张祺琪/ 90006907  中央软件院：田文罡/ 00232224，孙达/00120886，黄蕴芳/ 00246654  中央硬件院：施文超/00109740  企业网络：顾新/00136101，李昂/ 57909  IT产品线：吴鸿钟/00204126，胡红山/53611  固定网络产品线：吴涛/00117633  终端：洪太亮/65062  终端云：许汝波/62966  无线网络产品线：郭伟/00105071  电信软件与核心网：丁鹏亮/00192854  UC&C产品线：龚连阳/129383  公司架构部：张冠男/00256163  内网安全实验室：马宏伟/ 00152121  微软: Michael Howard  上海交通大学：来学嘉 | V1.0 |
| DKBA 7475-2015.05 | 网络安全能力中心：  张杰/0031659  韩庆/00265140  邹国雄/00214764  李赤阳/00306646 | 谢尔德实验室：  鲍丰/00236039、Wang Guilin/00253931、雷浩/00197658  网络安全能力中心：  何 纲/00152106、李秀峰/69636、郎风华/00134482、王功远/117010  中央软件院：  蒋孝霞/ 00254995、马岚/ 00256205、魏巍/ 00256168、涂异兰/ 00267497  中央硬件研究院：  施文超/00303178  无线产品线：  熊晓春/00257132、吴颂期/69062  固定网络产品线：  付天福/ 00168510、魏建雄/222905、张宁/ 00238449  电信软件：  刘高峰/00257474、付进/ 00257475、丁鹏亮/00192854  IT产品线：  叶思海/00 266267、吴鸿钟/00204126、  终端产品线：  徐亚新/ 00150328、洪太亮/ 00266693、邓太生/210606、方习文/00297002  核心网产品线：  王宏磊/00161798、  交通产品线：  龚连阳/129383、张轶炯/ 00147086、孙宏/ 00276216  网络能源产品线：  黄海鹏/ 00209481  海思：  潘时林/ 00256039、辛桂珍/ 00151940  网络操作系统部：  陈双龙/ 00152307 | V1.1 |
| DKBA 7475-2016.02 | 网络安全能力中心：  张杰/0031659  韩庆/00265140  邹国雄/00214764  李赤阳/00306646 | 谢尔德实验室：  鲍丰/00236039、雷浩/00197658  新研中央研究分部：  BAO FENG/00236039、YUEN TSZ HON/00268076、Wang Guilin/00253931、Wu Shuang/00253831、Chu Cheng Kang/00253107  网络安全能力中心：  丁国锋/90003999、韩庆/00265140  中央软件院：  孙达/00340853  中央硬件研究院：  施文超/00303178  无线产品线：  熊晓春/00257132、吴颂期/00330423  固定网络产品线：  付天福/ 00168510、魏建雄/222905、张宁/ 00238449  电信软件：  刘高峰/00257474、欧锻灏/00321872、付进/ 00257475  IT产品线：  徐秀兰/00301387、蒋世建/00333964  终端产品线：  何纲/ 00152106、洪太亮/ 00266693、方习文/00297002  核心网产品线：  王宏磊/00161798  交通产品线：  张轶炯/ 00147086、孙宏/ 00276216、张益波/00136876  网络能源产品线：  黄海鹏/ 00209481 | V1.2 |

目 录 Table of Contents

[简介 13](#_Toc443399108)

[使用对象 14](#_Toc443399109)

[适用范围 14](#_Toc443399110)

[规范解释 14](#_Toc443399111)

[用词约定 14](#_Toc443399112)

[术语解释 14](#_Toc443399113)

[1 密码算法的选择 16](#_Toc443399114)

[1.1 规则：禁止使用私有的、非标准的密码算法 16](#_Toc443399115)

[1.2 规则：禁止使用不安全的密码算法，推荐使用强密码算法 16](#_Toc443399116)

[1.3 规则：优先择国际标准密码算法，中国国密算法在发往国外的产品中应限制使用 20](#_Toc443399117)

[1.4 建议：在设计阶段就应考虑密码算法使用的敏捷性 21](#_Toc443399118)

[2 密码算法的应用 23](#_Toc443399119)

[2.1 随机数 23](#_Toc443399120)

[2.1.1 规则：密码算法中使用到的随机数必须是密码学意义上的安全随机数 23](#_Toc443399121)

[2.2 对称密码算法 25](#_Toc443399122)

[2.2.1 规则：使用分组密码算法时，应优先选择GCM或CBC模式 25](#_Toc443399123)

[2.2.2 规则：分组密码算法使用到的IV值，必须满足具体加密模式对IV值的要求 26](#_Toc443399124)

[2.2.3 规则：对称密码算法中在使用同一密钥时，不应当两次使用同一个IV加密数据 26](#_Toc443399125)

[2.2.4 规则：使用3DES算法，必须保证3个密钥K1、K2、K3各不相同 27](#_Toc443399126)

[2.2.5 规则：使用AES分组密码算法来代替RC4流密码算法 27](#_Toc443399127)

[2.2.6 建议：使用分组加密算法时，填充方式建议选择CMS-Padding 或ISO-Padding 27](#_Toc443399128)

[2.3 非对称密码算法 28](#_Toc443399129)

[2.3.1 非对称加密 28](#_Toc443399130)

[2.3.2 数字签名 29](#_Toc443399131)

[2.3.3 密钥协商算法 30](#_Toc443399132)

[2.4 哈希函数 32](#_Toc443399133)

[2.4.1 规则：对口令进行单向哈希时，必须使用基于口令的密钥导出算法 32](#_Toc443399134)

[2.5 消息认证码 34](#_Toc443399135)

[2.5.1 规则：禁止将hash(key||message)方式当作MAC来使用 34](#_Toc443399136)

[2.5.2 规则：在同时进行加密操作和MAC计算时，要使用不同的密钥 35](#_Toc443399137)

[2.6 密码安全协议 36](#_Toc443399138)

[2.6.1 规则：禁止使用SSL2.0、SSL3.0协议，TLS1.0协议仅可遗留使用 36](#_Toc443399139)

[2.6.2 规则： SSH协议中禁止使用CBC模式 36](#_Toc443399140)

[3 密码算法常见应用实例指导 38](#_Toc443399141)

[3.1 敏感数据对称加解密 38](#_Toc443399142)

[3.1.1 数据段类型的敏感数据加解密 38](#_Toc443399143)

[3.1.2 文件类型的敏感数据加解密 41](#_Toc443399144)

[3.2 敏感数据非对称加解密 44](#_Toc443399145)

[3.3 敏感数据的数字签名 45](#_Toc443399146)

[3.4 敏感数据的加密传输 46](#_Toc443399147)

[3.5 口令的安全存储 47](#_Toc443399148)

[3.6 非标准安全协议下密钥协商 49](#_Toc443399149)

[3.7 软件完整性保护 51](#_Toc443399150)

[4 附录 52](#_Toc443399151)

[4.1 常见密码算法标准 52](#_Toc443399152)

[4.2 分组密码算法工作模式 54](#_Toc443399153)

[4.2.1 常见工作模式介绍 54](#_Toc443399154)

[4.2.2 常见工作模式对比分析 62](#_Toc443399155)

[5 参考文献 64](#_Toc443399156)

**图目录 List of Figures**

[图 3‑1 数据段类型的敏感数据加解密方案流程图 38](#_Toc443399074)

[图 3‑2 DH算法数据流图 49](#_Toc443399075)

[图 4‑1 ECB模式下明文的加密图 54](#_Toc443399076)

[图 4‑2 ECB模式下密文的解密图 54](#_Toc443399077)

[图 4‑3 CBC模式下的加密 55](#_Toc443399078)

[图 4‑4 CBC模式下的解密 55](#_Toc443399079)

[图 4‑5 CFB模式下的加密 56](#_Toc443399080)

[图 4‑6 CFB模式下的解密 56](#_Toc443399081)

[图 4‑7 OFB模式下的加密 57](#_Toc443399082)

[图 4‑8 OFB模式下的解密 57](#_Toc443399083)

[图 4‑9 CTR模式Counter构造方法示意图 59](#_Toc443399084)

[图 4‑10 CTR模式下的加密 59](#_Toc443399085)

[图 4‑11 CTR模式下的解密 60](#_Toc443399086)

**表目录 List of Tables**

[表格 1‑1 算法列表 16](#_Toc443399049)

[表格 1‑2 典型加密算法安全强度表 17](#_Toc443399050)

[表格 1‑3 典型哈希算法安全强度表 17](#_Toc443399051)

[表格 1‑4 安全强度与生命周期对照表 18](#_Toc443399052)

[表格 1‑5 特殊场景下算法使用要求 18](#_Toc443399053)

[表格 1‑6 国密算法列表 19](#_Toc443399054)

[表格 2‑1 明文最后一个分组长度和填充字节对应关系 27](#_Toc443399055)

[表格 2‑2 特殊临时公钥值及其对应的会话密钥 30](#_Toc443399056)

[表格 2‑3 1000次迭代时口令复杂度和安全强度对照表 32](#_Toc443399057)

[表格 3‑1 业界应用PBKDF2算法的迭代次数 47](#_Toc443399058)

[表格 4‑1 常见密码算法及方案的标准化信息 51](#_Toc443399059)

[表格 4‑2 常见工作模式对比分析表 61](#_Toc443399060)

密码算法应用规范

# 简介

通常来说，使用密码算法可以提供以下四种服务：

1. **机密性：**机密性也称为保密性，是指敏感信息不泄露给未经授权的实体。使用加密算法最原始的目的也是如此。例如当使用口令登录或往数据库中储存机密医疗档案时，采用加密的方式能够确保只有拥有相应密钥的用户能够访问被保护的数据。
2. **完整性：**采用密码学相应手段能保证信息在储存和传输过程中不被查看和修改。如使用哈希函数可以为数据提供校验来保证数据安全。
3. **真实性：**使用密码算法相应方式可以确定远程用户或系统的身份。比如Web服务器提供的SSL证书能够保证用户连上的是正确的服务器。
4. **不可抵赖性：**不可抵赖性的概念在财务和电子商务应用中非常重要，它通过密码学工具或手段来证明一个唯一的用户进行了交易请求。用户是不可能对他或她的行为进行否认的。

当分析产品需求以及安全风险时，需要考虑使用上述服务中的哪几种来保护数据。安全的实现这些服务，前提是正确的选择和使用密码算法。

本规范的制订目的是希望能指导读者选择安全的密码算法，并结合应用实例指导其正确使用密码算法。规范涵盖的内容包括：对称加密算法、非对称加密算法、哈希算法、消息认证码、随机数产生等。

本规范总共分为4章：

第一章：主要介绍如何选择安全的加密算法；

第二章：介绍了使用安全加密算法时的一些规则；

第三章：通过提供一些场景实例和源代码方式介绍了密码算法的使用；

第四章：附录内容列出了一些常用密码算法标准并对常用加密工作模式进行介绍。

密钥管理相关内容不在本规范中描述，如产品需参考密钥管理相关内容，请参考公司《密钥管理安全规范》。数字证书管理相关内容不在本规范描述，如产品需参考数字证书管理相关内容，请参考公司《身份和访问管理安全设计规范》中相关内容。

# 使用对象

本规范使用对象主要为密码算法应用相关的需求分析人员、设计人员、开发人员、测试人员等。

# 适用范围

本规范适用于华为公司涉及密码算法使用的所有产品。

# 规范解释

本规范由网络安全能力中心维护。实施中遇到问题，可以到网络安全能力中心团队空间论坛<http://3ms.huawei.com/hi/group/9741/threads.html>上讨论。

规范内容会不定期更新，本规范及所引用规范的最新版本请登录[华为公司内部技术标准规范库](http://w3.huawei.com/ipd/tsl/#!tsl/portal/index.html)下载。

# 用词约定

规则：必须遵守的约定

建议：需要加以考虑的约定

说明：对此规则或建议进行相应的解释

# 术语解释

|  |  |
| --- | --- |
| **名词** | **解释** |
| 对称算法（Symmetric key algorithm） | 对称算法的加密密钥能够从解密密钥中推算出来，反过来也成立。在大多数对称算法中，加/解密密钥是相同的。 |
| 非对称算法（Asymmetric key algorithm） | 非对称算法（也叫公开密钥算法）是这样设计的：用作加密的密钥不同于用作解密的密钥，而且解密密钥不能根据加密密钥计算出来。之所以叫做公开密钥算法，是因为加密密钥能够公开，即陌生者能用加密密钥加密信息，但只有用相应的解密密钥才能解密信息。 |
| 哈希函数（Hash function） | 是一种将不同长度的数据映射成固定长度数据的函数。 |
| 消息认证码（Message authentication code） | 密码学上的一种数据校验和，通过使用对称密钥来检测数据是否发生意外或有意的修改。消息认证码简称MAC。 |
| 分组密码（Block cipher） | 分组密码是对称算法的一种，它将明文分成多个等长的组，并用相同的算法对每组进行加密。 |
| 流密码（Stream cipher） | 流密码也称为序列密码，是对称算法的一种，它是用算法和密钥一起产生一个随机码流，将其与数据流XOR产生加密后的数据流。 |
| 工作模式（Block cipher mode of operation） | 工作模式是指使用分组密码算法对数据进行加密转换的方法。常见的工作模式有ECB、CBC、CFB、OFB和CTR五种基本加密模式以及CCM、GCM等认证加密模式。 |
| 初始向量（IV，Initialization Vector） | 初始向量是许多工作模式中用于随机化加密的一块数据，因此可以由相同的明文、相同的密钥产生不同的密文。 |
| 填充（Padding） | 分组密码算法只能对数据长度为分组长度整数倍的数据进行处理，填充就是保证数据长度符合这种要求的一种方式。 |
| 盐值（Salt） | 通过在口令任意位置插入字符串，让哈希后的结果和使用原始口令的哈希结果不相符，这种过程称之为“加盐”。这个字符串就是盐值。 |
| 安全强度（Security strength） | 安全强度用来衡量密码算法或者密码系统的安全性，它是对破解密码算法或者系统所需要的工作量的一个数值度量。 |
| 口令（Password） | 口令是指用于身份认证、鉴权或者导出加密密钥的字符串，可由字母、数字和符号组成。 |
| 密钥（Key） | 密钥是一个结合密码算法一起使用的参数，拥有它的实体可以加密或恢复数据，没有它的实体则不行。 |

# 密码算法的选择

密码学领域是动态变化的。随着计算机技术的发展，许多原来安全的密码算法因被发现存在缺陷已不再安全。为保证系统安全性，产品开发人员在选择密码算法时需要有一定的前瞻性，尽量选择成熟的、具有足够安全强度的算法。这章主要介绍了在选择密码算法时需要注意的问题。

## 规则：禁止使用私有的、非标准的密码算法

**说明：**私有的、非标准的密码算法不应该在产品中使用。首先，如果不是具有密码学专业素养的专家设计的密码算法，这些算法难以达到密码学领域的专业性要求；此外其技术上也未经业界分析验证，有可能存在未知的缺陷；另一方面其违背了加密算法要公开透明的原则[[1]](#footnote-1)。**这些私有、非标准的算法包括但不限于：**

* 未公开的、自行设计的密码算法；
* 自行对标准密码算法进行改造的；
* 自行定义的通过变形/字符移位/替换等方式执行的数据转换算法；
* 用编码的方式（如Base64编码）实现数据加密目的的伪加密实现；
* 用差错控制编码（如奇偶校验、CRC）实现完整性校验。

在非密码学安全场景（不用于如机密性、完整性等密码学用途）且产品未在界面、资料声称算法用于密码学安全目的，出于正常业务需要可以使用Base64等算法。

## 规则：禁止使用不安全的密码算法，推荐使用强密码算法

**说明**：随着密码技术的发展以及计算能力的提升，一些密码算法已不再适合现今的安全领域。例如MD5算法，该算法已经于2004年由山东大学王小云教授的团队予以破解（人为构造出两个具有相同MD5值的信息），并且在2008年由国际密码学专家Lenstra利用该破解成果成功伪造出CA数字证书，其安全性已非常低下，因此， MD5不应用于所有密码学安全用途，包括用于数字签名，HMAC，口令单向保护、密钥导出、RNG等。又比如DES算法，因为计算能力提升导致其暴力破解成为可能，现有的暴力破解设备能将破解DES的时间减少到一天以内。这些算法统称为不安全的密码算法，如果产品中继续使用这些不安全的密码算法，有可能为客户的数据带来风险。

可遗留使用的密码算法是指那些本身并没有证实存在安全问题，但使用不当容易引入安全威胁的算法，或因为安全强度不足导致生命周期很快就要结束的算法。此类算法只可遗留在老代码中，新版本中应升级到推荐的强密码算法。

强密码算法是指当前被业界普遍认可，在其适合的应用场景下安全强度相对该场景下的其它加密算法有相对优势，在合理的安全假设下具有可证明安全性或对其实施破解在计算上显著不可行的密码算法。

下面表格基于NIST SP800-57定义的安全强度以及对应生命周期，列出了常见不安全密码算法、可遗留使用的密码算法和推荐使用的强密码算法。

表格 1‑1 算法列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **用途** | | **常见不安全密码算法** | **可遗留使用的密码算法** | **推荐使用的**  **强密码算法** |
| 对称加密 | 分组加密 | Blowfish, DES, DESX, RC2, Skipjack, 2TDEA, TEA | 3DES(k1、k2、k3各不相同) | AES(≥128 bits) |
| 流加密 | SEAL, CYLINK\_MEK, RC4 (<128 bits) | RC4(≥128 bits) \* | AES-CTR(≥128 bits)  AES-OFB(≥128 bits)  chacha20 |
| 哈希算法 | | SHA0, MD2, MD4, MD5, RIPEMD, RIPEMD-128 | SHA-1\* | SHA256或以上  SHA3 |
| 非对称加密 | | RSA (< 1024 bits) | RSA(1024 bits) | RSA(≥2048 bits) |
| 数字签名 | | RSA (< 1024 bits)  DSA (< 1024 bits)  ECDSA(≤160bits) | DSA(1024 bits)  RSA(1024 bits)  ECDSA(224-255 bits) | RSA(≥2048 bits)  DSA(≥2048 bits)  ECDSA(≥256 bits) |
| 密钥协商 | DH | L≤1024 bits  N\*≤160 bits | 1025≤L≤2047 bits  161≤N≤223 bits | L≥2048 bits  N≥224 bits |
| ECDH | ECDH(≤223bits) | ECDH(224-255bits) | ECDH(≥256 bits) |

* RC4算法安全性不足，IETF已在其发布的RFC 7465中禁止RC4用于TLS协议。因此自本规范v1.1版本发布之日起，禁止RC4用于TLS协议。在新版本规范中流加密推荐使用AES分组加密算法的CTR、OFB模式代替或使用chacha20流加密算法，参考规则“[使用分组密码算法来代替RC4流密码算法](#_规则：使用AES分组密码算法来代替RC4流密码算法)”；另外，由于单独使用流加密算法存在比特翻转攻击的风险，推荐使用认证加密算法，如AES-GCM、chacha20-poly1305。
* SHA-1用于数字签名及以及hash-only时为不安全算法，用于消息认证码（MAC）、密钥导出函数（KDF）、随机数生成（RNG）场景时可以认为是安全密码算法。另外，对MD5有效的攻击方法对于SHA1同样有效，所以强烈建议使用SHA256或更安全的哈希算法。
* DH密钥交换算法在域参数构成上目前有PKCS#3、X9.42两种标准，PKCS#3标准的DH安全强度和群规模参数L有关，而X9.42标准的DH安全强度除和群规模参数L有关外，还和子群规模参数N有关。因此，产品如采用PKCS#3标准的DH域参数，则表格中的N的大小可以不关注。

以下两表格来自于NIST SP800-57，列出了常用密码算法的安全强度及各安全强度对应的生命周期。

表格 1‑2 典型加密算法安全强度表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **安全强度** | **对称算法** | **离散对数**  **(DSA, DH)** | **大整数分解**  **(RSA)** | **椭圆曲线**  **(ECDSA，ECDH)** |
| ≤80 | 2TDEA | L\*=1024, N\*=160 | K\*=1024 | F\*=160-223 |
| 112 | 3TDEA | L=2048, N=224 | K=2048 | F=224-255 |
| 128 | AES-128 | L=3072, N=256 | K=3072 | F=256-383 |
| 192 | AES-192 | L=7680, N=384 | K=7680 | F=384-511 |
| 256 | AES-256 | L=15360,N=512 | K=15360 | F=512+ |

* L为公钥的比特长度（也为群规模的比特长度），N为私钥的比特长度（也为子群规模的比特长度）
* K为模的比特长度
* F为椭圆曲线基点阶数的比特长度

表格 1‑3 典型哈希算法安全强度表

|  |  |  |
| --- | --- | --- |
| **安全强度** | **数字签名/hash-only** | **HMAC/密钥导出/随机数产生** |
| ≤80 | SHA-1 |  |
| 112 | SHA224, SHA512/224, SHA3-224 |  |
| 128 | SHA256, SHA512/256, SHA3-256 | SHA-1 |
| 192 | SHA384, SHA3-384 | SHA224, SHA512/224 |
| 256 | SHA512, SHA3-512 | SHA256, SHA512/256, SHA384, SHA512, SHA3-512 |

表格 1‑4 安全强度与生命周期对照表

|  |  |  |  |
| --- | --- | --- | --- |
| **安全强度** | **应用方式** | **2014-2030** | **2031及以后** |
| <112 | 正向应用\* | 禁止使用 | |
| 反向应用\* | 继承性使用 | |
| 112 | 正向应用 | 可以使用 | 禁止使用 |
| 反向应用 | 继承性使用 |
| 128 | 正/反向应用 | 可以使用 | 可以使用 |
| 192 | 可以使用 | 可以使用 |
| 256 | 可以使用 | 可以使用 |

* 正向应用指算法作用于明文数据产生受保护数据的应用方式，比如加密、签名等。
* 反向应用指算法作用于受保护数据的应用方式，比如解密、验证签名等。

关于[NIST SP800-57](http://csrc.nist.gov/publications/PubsSPs.html)、[RFC7465](https://tools.ietf.org/html/rfc7465)请参考：



**规则描述：**

* **禁止使用不安全的密码算法；**
* **产品新版本中必须缺省使用推荐的强密码算法；**
* **由于行业标准遵从、与第三方系统对接、向下兼容性等原因，产品可支持不安全密码算法和遗留使用的密码算法，具体要求如下：**

表格 1‑5 特殊场景下算法使用要求

|  |  |  |  |
| --- | --- | --- | --- |
|  | **不安全密码算法** | **遗留使用的密码算法** | **强密码算法** |
| **行业标准遵从** | 只在标准自身必须支持不安全的密码算法或没有更安全的替代算法时允许支持 | 可支持，限制使用 | 必须支持并缺省使用，除非标准自身不支持强密码算法 |
| **与第三方对接\*** | 可支持，客户选择时提示风险 | 可支持，客户选择时提示风险 | 必须支持 |
| **向下兼容** | 除非新老设备间是通过标准协议交互，否则只允许继承性使用（解密、验证数字签名、验证HMAC等） | 可支持 | 必须支持并缺省使用，对于新的数据都必须使用 |

* 这里的“第三方”是指由客户指定对接的第三方设备或软件（如友商设备），公司内部组件或产品的对接不属于这里的第三方对接场景。
* **因遵从行业标准、特定国家/地区客户要求等因素必须支持某些不在表格1-1中的特殊密码算法（如TripleChurning、ZUC、GSM的A5算法、国密算法等）的场景，如果所支持算法符合客户所属国要求，则不强制要求使用表格1-1中所列算法；**
* **除上述原因外如果要使用不在表格1-1中的特殊密码算法，需到公司安全工程技术委员会密码学专家组进行评审。**

## 规则：优先择国际标准密码算法，中国国密算法在发往国外的产品中应限制使用

如果使用非客户所属国的国家标准算法，可能出现密码算法不被客户认可的情况。对于已公开的中国国密算法（SM2/SM3/SM4），在发往国外的产品中可以支持，但须默认关闭，除非客户明确要求或者明确声明不做要求。发往国外的产品中应禁止使用未公开的中国国密算法SSF33/ SM1/SM7/SM9）。

表格 1‑6 国密算法列表

|  |  |  |
| --- | --- | --- |
| **国密算法名称** | **所属分类** | **是否公开** |
| SSF33 | 分组加密算法 | 暂未公开 |
| SM1 | 分组加密算法 | 暂未公开 |
| SM2 | 椭圆曲线公钥密码算法 | 已公开 |
| SM3 | 哈希算法 | 已公开 |
| SM4 | 分组加密算法 | 已公开 |
| SM7 | 分组加密算法 | 暂未公开 |
| SM9 | 基于身份标识的非对称密码算法 | 暂未公开 |

对于某领域已被采纳为国际标准的中国国密算法（如被3GPP LTE采纳为国际标准的祖冲之算法），在该领域使用时不受此规则约束。

同时发多个国家的产品中，默认密码算法应该采用国际标准算法或这些国家都认可的国标算法。

## 建议：在设计阶段就应考虑密码算法使用的敏捷性

**说明：**密码算法的安全使用期限是基于当前的计算能力以及密码分析水平估算出来的，如果计算能力得到突破性的提高，或者密码算法设计上的缺陷被新发现，都会使密码算法达不到预计的安全寿命。当某种算法或密钥长度被发现是不安全的或不满足客户的要求时，就要进行替换。此外出于向下兼容原因，也经常需要更换其它加密算法来对以前的旧数据进行解密。因此，在设计密码算法应用方案时，就要考虑如何使密码算法使用更敏捷的问题。避免由于兼容性问题导致无法切换或者产生高昂的切换成本。

在设计时应使得算法模块独立，代码接口具有较好的扩展性，易于替换升级，而且保证系统支持的算法是可替代的。**不应将加密算法硬编码在应用代码中，可以使用工厂模式去设计一个加密类，这样可以根据实际加密的内容和加密需求来灵活选择加密算法。**

**常用的算法库（OpenSSL、IPSI、JDK等）中都提供了类似的接口函数，在使用时要尽量使用此类接口。不要将算法标识等参数写死，应从配置文件中读取。**随着密码分析技术的发展和计算机处理能力的提高，当前安全的密码算法可能在以后某个时间点开始就不再安全，为了保证应用系统在出现此类情况时仍然保持较高的安全性，应用系统使用的算法应该可配置，当前使用的某些密码算法在未来不再安全时，可以及时的平滑的切换到更安全的密码算法。可参考如下代码：

CipherFactory cipher = CipherFactory::defaultFactory();

String cipherName = ConfigurationManager.AppSettings["MyPreferredCipher"];

Cipher MyCipher = cipher.Create(cipherName);

代码里的“MyPreferredCipher”是配置文件里的配置项，配置文件内容示例如下：

<encryption>

MyPreferredCipher =”aes-256-cbc”

MyPreferredHash=”SHA256”

SaltFlag=”TRUE”

</encryption>

**为防止攻击者对密码参数配置文件进行恶意篡改，还应对该配置文件进行访问控制，只允许可信用户进行访问和修改，或者使用HMAC来校验其完整性。**

**开发人员在设计密文结构时应让密文中包含足够的密码运算参数的信息（如算法标识，密钥ID,IV,盐值等），确保算法更新后，密码算法接口可以根据密文中的密码参数信息正确的解密历史数据，实现应用系统版本的向下兼容。**例如：



# 密码算法的应用

选择了安全的密码算法还必须保证算法的使用是正确的，这样才保证密码系统的安全。本章介绍了密码算法应用时的一些规则和建议。

## 随机数

安全随机数在密码学中的应用很多，比如算法实现时某些参数的生成，算法使用时密钥的产生、IV的生成等。密码学意义上的安全随机数发生器必须能够保证其产生的随机数很难被全部或部分预测到，否则会带来很大的安全隐患，使其成为整个安全链中最薄弱的一环。比如，如果密钥的产生使用了不安全的随机数，攻击者可以绕过算法本身直接去猜解密钥。

### 规则：密码算法中使用到的随机数必须是密码学意义上的安全随机数

**说明：**随机数用于生成IV、盐值、密钥等，都属于密码算法用途。不安全的随机数使得密钥、IV等可被全部或部分预测。密码学意义上的安全随机数，要求必须保证其不可预测性。

**产品可以直接使用真随机数产生器产生的随机数。或者使用真随机数产生器产生的随机数做种子，输入密码学安全的伪随机数产生器产生密码学安全随机数。**

按照BSI（德国信息安全办公室）分类标准，真随机数产生器分为物理真随机数产生器和非物理真随机数产生器。物理真随机数产生器其随机性来源于真正的物理随机过程，如热噪声，散粒噪声，二极管击穿熵、自由运行的振荡器等。各种商用硬件随机数产生器都属于物理真随机数产生器。非物理真随机数产生器其随机性来源于系统外部的随机事件，如磁盘I/O、中断、键盘鼠标操作时间等。

关于真随机数产生器，目前没有统一的实现标准，但是有一些评估和认证标准可以遵循。如BSI的AIS 31标准以及NIST SP 800-90B等，其中NIST SP 800-90B的评估工具已经开源，需要自行设计硬件随机数产生器的产品可以使用该工具做熵评估。

如果产品有过CC、NIST等认证的需求，推荐选取已过认证的硬件随机数产生器以节省开发时间，否则如果自行设计硬件随机数产生器，需要满足目标认证机构的认证标准。

已知的可供产品使用的密码学安全的非物理真随机数产生器有：

* Linux操作系统的/dev/random设备接口
* Windows操作系统的CryptGenRandom接口

对Linux操作系统下的/dev/random接口，在系统内核没有搜集到足够的外部随机事件时会阻塞，具体阻塞的时间不确定。因此，在某些产品中，直接使用/dev/random会有性能方面的问题，如果产品面临这样的问题，可以使用密码专家组提供的种子文件方法解决（见密码专家组第21次会议会议纪要），如果没有条件使用种子文件方法，可以使用安全能力中心提供的《熵源不足条件下安全随机数产生方案》来解决。

已知的可供产品使用的密码学安全的伪随机数产生器包括：

* OpenSSL库的RAND\_bytes
* OpenSSL FIPS模块中实现的各种DRBG（符合NIST SP 800-90A标准）
* 中软iPSI组件的CRYPT\_random( )；（符合X 9.31标准）
* 中软VPP的IPSI\_CRYPT\_rand\_bytes（符合NIST SP 800-90A标准）
* JDK的java.security.SecureRandom

如果产品有过认证的需求，推荐选择符合标准的伪随机数产生器。各种伪随机数产生器必须设置安全的种子，种子（包括默认种子）可以由真随机数产生器产生。没有条件使用真随机数产生器（包括物理、非物理真随机数产生器）产生种子的产品，自行设计的用于产生种子的随机数产生器必须经过密码学专家组评审。

**注意：**

1. C标准库函数random()、rand( )以及Java的java.util.Random类，所产生的随机数都不属于安全随机数，禁止用于安全用途。
2. 按照NIST SP 800-131A标准规定，原FIPS 186-2, [X9.31]和ANS [X9.62]等标准中规定的随机数生成算法2015年12月31日后不再被允许使用。因此，对于有过FIPS认证需求的产品，在采用或实现随机数生成算法时应避免使用已经过期的算法。本规范推荐使用修正后的NIST SP 800-90A标准中的HASH\_DRBG、HMAC\_DRBG、CTR\_DRBG三种随机数生成器，禁用NIST SP 800-90A中的DualEC-DRBG生成器。
3. 随机数序列的随机性可以通过随机数检测工具检测，NIST提供了随机数检测工具并且在NIST SP800-22中详细说明了该工具的使用方法和检测报告的解读方法。检测工具网址：

<http://csrc.nist.gov/groups/ST/toolkit/rng/documents/sts-2.1.2.zip>

《熵源不足条件下安全随机数产生方案》及其相关附件，请参考：

、、

## 对称密码算法

对称密码算法使用相同的密钥来加密和解密数据。对称算法通过加密来提供机密性保护，通过认证来提供真实性或完整性保护，它包括分组密码算法和流密码算法。使用分组密码算法时要注意工作模式的选取以及IV的使用，使用流密码算法时也要保证每次加密使用不同的IV。算法的使用不当很有可能给系统带来安全隐患。

### 规则：使用分组密码算法时，应优先选择GCM或CBC模式

**说明：**分组密码算法有多种工作模式，不同工作模式有特定的适用场景。**GCM模式是目前业界推荐使用的对称加密算法工作模式，CBC模式是适用性最好、使用最广泛的工作模式，产品应该根据具体应用场景优先使用这两种加密模式。**

ECB模式对于同样的明文块会生成相同的密文块，不能提供严格的数据保密性，不能抵抗替换攻击，攻击者可以调换加密块的顺序而不被发现。因此，如果没有通过公司安全工程技术委员会密码学专家组的评审，应禁止ECB模式直接用于加密目的。

对于有并行处理、加密认证等明确的特殊需求的，可以选择具备对应特性的工作模式。如下：

* 需要进行认证加密以保证明文或其它认证内容完整性时，应该使用GCM或CCM模式；
* 需要对数据进行并行处理、预计算或者需支持随机访问时，可使用CTR模式；
* 存储设备加密时，可使用XTS-AES模式；
* 图象、语音等信号流加密时，可使用OFB模式，即使信号数据流在传输过程中部分错误也不影响数据流其他部分。

更多工作模式选择以及特性介绍请参考附录[常用工作模式介绍](#_常见工作模式介绍)。

### 规则：分组密码算法使用到的IV值，必须满足具体加密模式对IV值的要求

**说明：IV不需要加密存储，但是对于不同的对称算法加密模式，IV有不同的要求。根据**NIST SP800-38A标准，对于CBC、CFB模式， IV在产生之前都不应该被预测出来，对于OFB，CTR模式，则只须保证IV的唯一性。此外，GCM、CCM模式也只需保证IV的唯一性即可。这里的唯一性是指，对于相同的密钥，每次加密数据的IV各不相同。

如果CBC、CFB模式的IV值可以被预测，则可能会遭受某些密码攻击。例如，TLS1.0协议中使用CBC模式时遭受的BEAST攻击，SSH2.0协议中使用CBC模式时遭受的明文恢复攻击，这些攻击均由IV值可以被预测所致。对于对称算法的OFB、CRT模式，如果每次加密使用相同的IV值，就可能遭受密钥流重用攻击。

**推荐使用安全的随机数产生不可预测的IV，**具体参考[2.1.1](#_规则：密码算法中使用到的随机数必须是密码学意义上的安全随机数)章节。IV值使用安全的随机数，即使在使用同一密钥的情况下，攻击者也无法获得到密文间的相关性。

产生不可预测IV的方法除上述由安全随机数产生外，也可以使用对称加密密钥加密计数器并以其结果作为不可预测的IV。

对于OFB、CTR、GCM、CCM模式，其IV（CTR的IV即COUNTER）必须保证唯一性。对于CTR，须进一步保证每个分组运算时COUNTER值都不同，推荐参考“常见工作模式介绍”中“[计数器模式](#_计数器模式(Counter_Mode，CTR))”部分提到的COUNTER构造方法保证其唯一性。

对于CBC、CFB、OFB、CTR模式，IV的长度等于对称加密算法的分组长度，具体到AES对称加密算法，其值为16字节。对于GCM，其IV长度可选，推荐采用96比特的IV。

### 规则：对称密码算法中在使用同一密钥时，不应当两次使用同一个IV加密数据

**说明：**初始化向量IV与密钥相比有不同的安全性需求，IV通常无须保密，然而在大多数情况下，不应当在使用同一密钥的情况下两次使用同一个IV来加密数据。因为同一密钥的情况下重用IV，则相同的密文块加密前的明文块是相同的，这样就可能会遭受选择明文攻击。另外对于某些特定算法或工作模式同一密钥的情况下重用IV会带来严重安全漏洞。如：

1. 使用OFB工作模式加密时，如果使用相同的IV和密钥来加密两条消息，攻击者知道了加密后的两个消息包之一的明文，就可以对另外一个消息包进行解密；
2. 对于CFB工作模式而言，如果使用相同的IV和密钥来加密两条消息，攻击者知道了加密后的两个消息包之一的明文，则另一个消息包的首个明文块信息泄露。

### 规则：使用3DES算法，必须保证3个密钥K1、K2、K3各不相同

**说明：**由3DES加密原理可知，如果K1=K2=K3，则3DES安全强度等同于DES。当K1=K3≠K2，如果攻击能够获取到足够多的明文密文对（约240），则3DES安全强度只有80位。只有K1≠K2，K2≠K3且K3≠K1，3DES才能达到112位的安全强度（由于中途相遇攻击，3DES的有效安全强度仅为112位，而不是3\*56的168位）。

### 规则：使用AES分组密码算法来代替RC4流密码算法

**说明：**RC4流加密算法安全性不足且使用时很容易出现错误，IETF已在SSL/TLS标准中禁用RC4，为此，本规范将其列为不安全密码算法，如产品确实有使用流密码算法加密数据的应用场景，可使用AES-CTR、AES-OFB替代RC4。

目前chacha20流加密算法已经由IETF采纳为参考消息类别（RFC 7539），因此在流加密场景下产品也可以选择使用该算法，如在加密的同时须保证被加密数据的完整性，产品可以使用chacha20-poly1305认证加密算法。

### 建议：使用分组加密算法时，填充方式建议选择CMS-Padding 或ISO-Padding

分组加密算法的某些操作模式（如CBC模式）要求明文被填充以保证其长度满足分组长度的整数倍。除需要遵从标准协议（如IPSec中的ESP填充）的场景外，本规范建议选择的填充方式为：CMS-Padding（RFC5652中定义）、ISO-Padding（ISO/IEC 7816-4中定义）。

CMS-Padding填充方法为：缺几补几，直到补齐分组，刚好为整分组时视为缺一个整分组。

ISO-Padding填充方式为：先补0x80，后补若干0x00直到补齐分组。

下表为分组长度为16字节（如AES算法）时，明文最后一部分长度和填充字节之间的对应关系。

表格 2‑1 明文最后一个分组长度和填充字节对应关系

|  |  |  |
| --- | --- | --- |
| **最后一部分明文长度** | **CMS-Padding** | **ISO-Padding** |
| 0字节（刚好为整分组） | 16字节0x10 | 1字节0x80后跟15字节0x00 |
| 1字节 | 15字节0x0F | 1字节0x80后跟14字节0x00 |
| 2字节 | 14字节0x0E | 1字节0x80后跟13字节0x00 |
| … | … | … |
| 14字节 | 2字节0x02 | 1字节0x80后跟1字节0x00 |
| 15字节 | 1字节0x01 | 1字节0x80 |

如果产品可以保证明文长度为分组长度的整数倍，加解密时可以选择不做填充。

## 非对称密码算法

非对称算法主要用于提供数据完整性、真实性和不可抵赖性服务（如数字签名）。RSA算法是使用最广泛的非对称加密算法，常用的数字签名算法有DSA、RSA以及基于椭圆曲线的ECDSA。本小节主要介绍非对称密码算法使用时的一些注意规则。

### 非对称加密

#### 规则：使用RSA算法时要选取合适的公共指数e

**说明：**公共指数e位长越小，RSA加密或签名校验时的效率越高。但位长过小时（如e=3）可能会遭受一系列安全攻击[[2]](#footnote-2)。如e=3时，若两不同消息m1、m2都使用RSA加密且使用相同的模N，当m1和m2满足某些特定的关系时，攻击者有可能恢复出m1和m2。

实际使用中e推荐取值为216 + 1，即0x10001，具体可参考实例[敏感数据非对称加解密](#_敏感数据非对称加解密)。

#### 规则：使用RSA算法进行加密操作时，应使用OAEP填充方式

**说明：**对数据进行填充要非常谨慎，因为一些有经验的黑客有可能从中找到一些线索。早期的PKCS#1填充标准就曾受到一种自适应选择明文攻击的威胁。RSA实验室在PKCS#1 V1.5以后的标准中增加了OAEP（Optimal Asymmetric Encryption Padding）填充模式，可以有效阻止这类威胁。一般算法库提供的接口函数中都会有填充方式选择参数，如OpenSSL中为RSA\_PKCS1\_OAEP\_PADDING。

按照PKCS#1 V2.1规定，使用OAEP填充方式时，应该保证：明文长度+2×填充时使用的HASH算法的输出长度+2≤密钥长度，其中长度均以字节计。如果明文长度太长而不满足这个限制条件，会导致OAEP填充失败。例如，密钥长度2048bits（256字节），使用SHA256哈希算法（输出长度32字节），则明文长度最长为：256-2×32-2=190字节。

#### 规则：使用非对称算法时，加密和签名要使用不同的密钥对

**说明：**使用非对称算法时，加密和签名要使用不同的密钥对。签名密钥对的私钥仅用于签名，其对应的公钥用于验证签名，而加密密钥对的公钥仅用于加密，其对应的私钥用于解密。用于加密和用于签名的密钥对的管理方式和使用时限不同。为了对已有签名结果进行验证，用于验证签名的公钥需要长期保存，而用于签名的私钥在有效期之后必须销毁。为了对已有密文进行解密，用于解密的私钥需要长期保存，用于加密的公钥在有效期之后需要销毁。

### 数字签名

#### 规则：使用DSA签名时要保证签名的r、s值均不为0

**说明：**FIPS PUB 186-4中4.6节对DSA算法的签名过程进行了描述，其中提到，进行DSA数字签名时，签名结果为(r,s)，签名方需要对r、s值进行检查，若r为0或者s为0则重新产生随机数k并生成签名。[FIPS PUB 186-4](http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf)原文相关描述请参考：



#### 规则：在同时需要对数据进行对称加密和数字签名时，使用先签名后加密的方式

**说明：**先签名后加密是指先对消息进行签名，然后对消息的签名值和消息一起进行加密。如果采用先加密后签名的方式，接收方只能知道该消息是由签名者发送过来的，但并不能确定签名者是否是该消息的创建者。比如在发送一个认证凭据时采用先加密后签名的方式，消息在发送过程中就有可能被第三方截获并将认证凭据密文的签名值修改为自己的签名，然后发送给接收方。第三方就有可能在不需知道认证凭据的情况下通过这种方式来通过认证获取权限。

采用先签名后加密方式可以避免这类问题的发生，因为只有在知道消息明文的情况下才能对其进行签名。

#### 规则：不要使用二进制域的ECDSA认证方式

**说明：**ECDSA使用了椭圆曲线密码学原理（ECC , Elliptic curve cryptography），ECC有素数域和二进制域两种实现方式，后者在专门的硬件上实现计算更为有效，前者通常在通用的处理器上更为有效。

2011年3月19日，两位研究者在他们发表的IACR论文中证明：如果二进制域椭圆曲线标量乘法实现不当（例如OpenSSL0.9.8o中利用蒙哥马利阶梯法计算标量乘法的实现），通过时间攻击从使用带二进制域ECDSA认证的OpenSSL的服务器中恢复出TLS私钥是可能的[[3]](#footnote-3)。这种威胁在OpenSSL 1.0.0e版本中已经修复。出于安全考虑，不要再使用二进制域的ECDSA签名认证方式。

#### 建议：使用RSA算法进行签名操作时，采用PSS填充方式

**说明：**对于RSA签名运算，德国电信（DT）已将PSS填充方式作为其唯一填充方式。该填充方式由RSA实验室在PKCS#1 V2.1中提出（请参考RSA实验室关于RSA-PSS的介绍：RAISING THE STANDARD FOR RSA SIGNATURES:RSA-PSS），PSS填充方式相比PKCS1-V1.5填充方式具有更高的“可证明”安全性，推荐新应用中的RSA签名操作采用PSS填充方式。

样例请参考常见应用实例指导中[敏感数据的数字签名](#敏感数据的数字签名)中的相关内容。

德电对于PSS填充方式的相关要求请参考：



### 密钥协商算法

密钥协商是指采用特定的协议在不同通信实体间共享密钥或密钥材料的过程。DH/ECDH是常见的密钥协商算法。

#### 规则：DH密钥协商过程中，双方需对对方的临时公钥值进行验证，以保证临时公钥值不能为某些特殊值，防止简单替换攻击

**说明：**DH协商过程中如果双方交换的临时公钥值（或）被攻击者替换为某些特殊值，那么最后通信双方协商出来的密钥将会是一些固定值。这样攻击者就可以获得通信双方协商出来的密钥，危及到通信的安全，所以需要协商密钥时双方均需对对方的临时公钥值做验证以保证其不能为某些特殊值。

表格 2‑2 特殊临时公钥值及其对应的会话密钥

|  |  |
| --- | --- |
| 特殊临时公钥值 | 双方协商得到的会话密钥 |
| 0 | 0 |
| 1 | 1 |
| p-1 | 1或p-1 |
| p | 0 |

关于DH算法流程请参考本规范[3.6](#非标准安全协议下密钥协商)节中相关描述。

关于简单替换攻击，请参考：



#### 规则：非信任环境内使用DH/ECDH算法进行密钥协商时，必须确保双方身份的合法性

**说明：**这里的非信任环境是指攻击者有条件实施主动攻击（如中间人攻击）的通信环境（例如公网）。在非信任环境中使用DH/ECDH算法进行密钥协商时，如果不能保证对方身份的合法性，则无法保证通信安全。确保对方身份合法性基础上进行的密钥交换称为认证密钥交换（AKE），认证密钥交换可以防止攻击者假冒通信中的某一方或双方实施主动攻击。

DH/ECDH密钥协商本身不提供身份认证功能，但是可以结合身份认证使用，如SSL/TLS协议就可以选择使用数字证书认证对方的身份并结合DH密钥交换算法保证会话密钥协商的安全性。

对于标准协议中支持单向认证的场景（如https单向认证）进行DH/ECDH密钥协商时，可以在仅使用单向认证的情况下进行密钥协商。

本规则相关例外场景请参考《不安全的密码算法和安全协议管控策略和要求》（待发布）。

#### 建议：在使用DH/ECDH进行密钥协商时推荐支持完全前向保密特性

在DH/ECDH密钥协商中，如果每次协商都使用有限域上固定的密钥对，而不是每次都重新产生密钥对，则当该密钥对被破解后，此前所有使用该密钥对协商出来的会话密钥都可以被计算出来，从而使得这些会话密钥加密的数据不再安全。如果每次都重新在有限域上选取新的密钥对用作密钥协商，则即便通过暴力破解获得某次协商时的密钥对，也仅能计算出该次协商到的会话密钥，仅可解密出该次会话的加密内容，这样就可以保护之前会话中的加密内容，做到完全前向保密。

在SSL/TLS中协议中，如使用DH/ECDH算法进行密钥协商，选取包含DHE或ECDHE密钥交换算法的加密套件即可支持完全前向保密特性[[4]](#footnote-4)，应避免选取仅包含DH/ECDH的加密套件。

## 哈希函数

哈希函数将任意长度的输入转换为固定长度的值，哈希算法也叫摘要算法、单向散列算法、数字指纹算法。许多提供安全服务的算法中都将哈希算法做为其算法的一部分。比如数字签名、基于哈希的消息验证码、密钥导出函数和随机数发生器。口令的单向哈希保存也需要用到哈希函数。

### 规则：对口令进行单向哈希时，必须使用基于口令的密钥导出算法

**说明：**口令属于个人敏感数据，即使是系统管理员也不应该知道用户的明文口令。所以应只存储口令的单向哈希值。但单纯的口令哈希值，无法防止彩虹表攻击（彩虹表就是一个庞大的、针对各种可能的字母组合预先计算好的哈希值的集合）。增加了盐值计算出来的哈希值，可以防止相同口令生成相同的哈希值，但在hashcat这种调用GPU计算资源的暴力破解工具面前，安全性仍不足。所以口令哈希的计算在增加盐值的基础上还要考虑迭代计算。在这种场景下可以使用基于口令的密钥导出算法，如PBKDF2、scrypt。

对于PBKDF2算法，IETF RFC 2898和NIST SP 800-132国际规范中已经进行了详细定义。在业界（思科、苹果、微软等）该算法已经得到广泛的应用并被主流的密码算法库（OpenSSL、JDK等）广泛的支持，因此本规范推荐使用PBKDF2。PBKDF2具体用法请参考实例[口令的安全存储](#_口令的安全存储)。

对于scrypt密钥导出算法，由于需要占用较多（可通过参数调整）的内存资源，使得实施并行计算进行破解的成本大大增加，因此通常认为其安全性高于PBKDF2。对于安全性要求非常高并且硬件资源允许的场景，可以使用scrypt。

其它密钥导出算法暂不推荐使用，如须使用，需到公司安全工程技术委员会密码学专家组进行评审。

**PBKDF2算法使用规则：**

* **迭代次数默认推荐10000次，对于性能有特殊要求的产品最低可以迭代1000次；**
* **哈希函数推荐选择SHA256或更安全的哈希算法；**
* **盐值至少8字节，应使用安全的随机数；**
* **用于口令单向哈希时，其输出长度应该不小于256比特；**

对于PBKDF2算法，从算法原理看，其计算时间和迭代次数成线性关系。对该算法在OpenSSL下的实现进行了实测以供产品评估其性能，结果如下：

操作系统：Linux suse-32 3.0.76-0.11-pae

CPU：Intel(R) Xeon(R) CPU E5310 @ 1.60GHz，8核

测试参数：口令长度10字节，盐值长度8字节，输出长度32字节，哈希算法SHA256

测试结果：平均每次迭代3微秒左右。10000次迭代约30毫秒。

对于性能要求较高，使用1000次迭代无法满足产品要求的场景，可以在系统口令复杂度（口令构成、口令长度）要求适当提升的场景下根据产品实际性能要求降低迭代次数，但是需要到产品线TMG报备或参考密码专家组2015年第26次会议会议纪要的要求保证口令长度达到安全长度。

对于性能要求非常高的场景，产品可根据性能要求在1000次迭代的基础上进一步降低迭代次数，极端情况下可使用HAMC做口令单向哈希，但是，低于1000次迭代时一般会存在安全风险（下表为密码专家组给出的1000次迭代时的比特安全强度与口令长度对照表，供参考），迭代次数越小风险越高，此外，HMAC算法本身不能有效抵抗暴力攻击（即蛮力猜测），其安全性仅达到PBKDF2迭代1次时的安全强度，存在较高的安全风险。使用低于1000次迭代的PBKDF2或HMAC进行口令单向哈希的场景必须到产品线TMG报备，经评估风险可接受时方可使用。HMAC具体使用要求请参考《身份和访问管理安全设计规范》中口令维护章节相关内容。

表格 2‑3 1000次迭代时口令复杂度和安全强度对照表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **安全强度** | **迭代次数** | **小写字符和数字** | **大小写字符和数字** | **所有可输入字符** |
| 60（低） | 1000 | 10 | 9 | 8 |
| 80（中) | 1000 | 14 | 12 | 11 |
| 128(高) | 1000 | 23 | 20 | 18 |

上表为迭代次数为1000时PBKDF2的比特安全强度和口令长度对照关系表，例如，口令为所有可输入字符的情况下，长度为8时可以达到60比特安全强度，长度为11时可以达到80比特的安全强度。

**scrypt算法使用规则：**

* **CPU/内存占用参数推荐至少为；**
* **盐值至少8字节，应使用安全的随机数；**
* **块规模参数推荐至少为8；**
* **并行化参数推荐至少为1；**
* **用于口令单向哈希时，其输出长度应该不小于256比特；**

scrypt是一个比较新的算法，IETF于2015年11月发布了该算法的Internet Draft的第五个版本。本规范中提供的使用参数来源于算法发明人Colin Percival对scrypt算法进行的性能测试结果。这组参数在计算时间上相当于PBKDF2算法进行86000次迭代所用的时间。关于[scrypt算法](https://datatracker.ietf.org/doc/draft-josefsson-scrypt-kdf/)及其计算性能请参考：



**例外情况：**本规则对于口令需要还原的场景（例如：需要记住口令或登录凭证等用于自动登录的场景，以及认证协议有约束的场景如digest-MD5认证等）例外。

## 消息认证码

消息认证码（MAC）用于保证数据的真实性和完整性。大多数MAC算法都是双方使用相同密钥对双方间交换的消息进行认证。本小节将介绍MAC应用时应注意的一些规则。

### 规则：禁止将hash(key||message)方式当作MAC来使用

**说明：**这种方式是指直接将密钥与消息进行拼接后再进行哈希运算以达到计算MAC的目的。这种方式存在风险，因为安全的MAC函数必须保证其不可伪造性，即在不知道密钥key的情况下，攻击者很难构造出新的消息和其对应的MAC值，但采用这种方式攻击者可以达到这种目的。

如果将hash(key||message)作为MAC来使用，很容易遭受消息扩展攻击（length extension attack）。攻击者如果知道hash(key||message)和message，可以在不知道key的情况下计算得到h=hash(key||message||||)（其中为计算hash(key||message)时的填充，可为任意消息），然后可将message||||和h（即新的消息和MAC值）一起发送给接收者。接收者察觉不到message已经被修改[[5]](#footnote-5)。

另外，如果将hash(message||key)作为MAC来使用，一旦哈希算法不够安全，就有遭受“存在性攻击”的风险。

因此，不要将hash(key||message)或hash(message||key)作为MAC来使用，应使用标准的MAC算法（如HMAC[[6]](#footnote-6)或CMAC）。

### 规则：在同时进行加密操作和MAC计算时，要使用不同的密钥

**说明：**加密操作和MAC计算使用相同的密钥在某些情况下会引入安全问题，如CBC分组加密结合CBC-MAC一起使用时，如果使用同一密钥，将会使得认证功能失效[[7]](#footnote-7)。加密操作和MAC计算使用同一密钥本身也违背了密钥用途单一性的原则，如果其中一种用途被破解后另外一种用途也同时失效。

推荐的做法：一种简单的方式是**使用带认证的加密模式，如GCM和CCM**。这种方式用户只需一个密钥和IV就可得到密文和MAC值，也能保证其安全性。但这两种模式只有部分算法库和协议支持（如OpenSSL的EVP接口，TLSv1.2）。对于新的应用，如果所选用的算法库和协议支持，应尽可的使用这两种加密模式。

还有一种更通用的做法是**使用密钥导出函数（KDF）通过一个主密钥导出两个子密钥，分别用于加密和MAC计算**。这种做法适用于算法库或协议不支持带认证的加密模式以及需要考虑兼容性的场景。此时的加密和MAC操作推荐使用Encrypt-then-MAC方式，即先对消息进行加密然后计算密文的MAC值。相比其它方式（Encrypt-and-MAC、MAC-then-encrypt），Encrypt-then-MAC还具备IND-CPA（Indistinguishability under chosen-plaintext attack）安全特性。强烈建议计算MAC时将加密用的IV和密文合在一起计算MAC值，这样可同时保证IV的完整性。

在计算MAC时，对于密钥长度要满足如下要求：

1、对于CMAC，其密钥长度取决于所基于的对称加密算法的密钥长度要求。

2、对于HMAC（参考RFC2104），如果其所使用的HASH算法的输出长度大于128bit，则密钥长度至少为HASH算法输出长度，如果其所使用的HASH算法的输出长度小于128bit，则密钥长度至少为128 bit。

[RFC2104](https://www.ietf.org/rfc/rfc2104.txt)中对HMAC密钥长度相关描述请参考：



## 密码安全协议

### 规则：禁止使用SSL2.0、SSL3.0协议，TLS1.0协议仅可遗留使用

**说明：**SSL2.0和SSL3.0协议因存在很多安全问题已分别于2011年3月和2015年6月被IETF禁用。SSL2.0存在的主要问题有：使用MD5做消息认证算法、握手消息未受保护，容易受到中间人攻击从而欺骗客户端选择弱加密算法、加密和消息认证使用相同的密钥等。SSL3.0存在的主要问题有：数据记录协议容易受到POODLE攻击、密钥交换时如启用会话恢复及重协商机制容易受到中间人攻击等。

TLS1.0协议支持的所有加密套件中，对称加密算法仅支持RC4算法和分组加密算法的CBC模式，RC4已经公认不安全并被IETF在TLS所有版本协议中禁用，而对称算法的CBC模式存在IV可预测的问题，从而容易受到BEAST攻击。因此，TLS1.0协议仅在向下兼容、第三方对接等场景下可遗留使用。

本规则相关例外场景请参考《不安全的密码算法和安全协议管控策略和要求》（待发布）。

### 规则： SSH协议中禁止使用CBC模式

**说明：**SSH协议当前的版本是2.0，该版本的SSH协议中，使用CBC模式的对称加密算法可能受到明文恢复攻击而泄露加密传输的内容。因此，在SSH2.0中应禁止使用CBC模式。

OpenSSH是对SSH协议的具体实现，在OpenSSH6.4以前版本中，产品可以选择AES-CTR对称加密算法，从OpenSSH6.4版本开始，可以选择AES-GCM对称加密算法，从OpenSSH6.5开始，可以选择chacha20-poly1305对称加密算法。

# 密码算法常见应用实例指导

本章内容主要介绍一些常见应用场景下密码算法的使用，结合源代码来说明这些应用场景下使用密码算法的一些注意事项。

## **敏感数据对称加解密**

### 数据段类型的敏感数据加解密

**场景描述：**

在实际业务中，为保证敏感数据的机密性，需要对敏感数据进行对称加密，常见的敏感数据如：

* 被认证端需要支持自动登录场景下的口令：如数据库Client端保存的连接口令、设备间通信Client端保存的口令、板间通信Client端保存的口令、远程服务调用口令等；
* 网管系统和被管理设备之间传输的敏感配置数据；
* 密钥、协议中的Key；
* 信用卡帐号、金额、CVV码；
* ……

**方案概述：**

对数据段类型的敏感数据进行对称加密时，通常采用的是对称分组密码算法对数据进行块加密。对于对称分组密码算法的使用，除了要选择安全的对称加密算法以及正确生成密钥外，还应注意密码算法的工作模式的选取以及IV值的生成。

简单的方案流程如下图，实际使用中可以调用算法库中的经过封装的接口，比如OpenSSL的EVP接口，则无需关心分组、填充这些细节。





图 3‑1 数据段类型的敏感数据加解密方案流程图

**设计要点：**

1. 对称加密算法使用AES128。
2. 对称加密算法的工作模式，使用CBC模式。
3. 分组的填充方式推荐使用PKCS#7中定义的填充方式，即填充的值为填充的字节数，当应用环境要求密文与明文长度相同时，可使用特殊的填充方式CBC-CS（具体参考标准SP800-38A-Addendum）。
4. 加密中所需的IV值，由安全随机数发生器产生。对于AES分组加密算法CBC模式，IV的长度为16字节。

**参考样例：**

以下以使用OpenSSL库为例，来说明在对数据段类型的敏感数据进行对称加解密时，程序实现过程中应该注意的点。本参考样例包括关键代码、详细例程和测试结果。

/\* 数据加密过程 \*/

EVP\_CIPHER\_CTX ctx;

/\* IV取值为随机数\*/

RAND\_bytes(IV, AES\_BLOCK\_SIZE);

/\* 初始化加密结构体，设置密钥、IV与加密算法类型AES128-CBC \*/

EVP\_CIPHER\_CTX\_init(&ctx);

EVP\_EncryptInit\_ex(&ctx, EVP\_aes\_128\_cbc(), NULL, pKey, IV);

/\* 保存IV到加密数据头部 \*/

memcpy\_s(dataOut, dataOutMaxLen, IV, AES\_BLOCK\_SIZE);

/\* 数据加密 \*/

EVP\_EncryptUpdate(&ctx, dataOut+ AES\_BLOCK\_SIZE, &encLen, dataIn, inLen);

/\*不足一个分组进行填充\*/

EVP\_EncryptFinal\_ex(&ctx, dataOut+ AES\_BLOCK\_SIZE+encLen, &encLen);

/\* 释放结构体内存 \*/

EVP\_CIPHER\_CTX\_cleanup(&ctx);

/\* 数据解密过程 \*/

EVP\_CIPHER\_CTX ctx;

/\* 读取IV \*/

memcpy\_s(IV, IVMaxLen, dataIn, AES\_BLOCK\_SIZE);

/\* 初始化加密结构体，设置密钥、IV与加密算法类型AES128-CBC \*/

EVP\_CIPHER\_CTX\_init(&ctx);

EVP\_DecryptInit\_ex(&ctx, EVP\_aes\_128\_cbc(), NULL, pKey, IV);

/\* 数据解密 \*/

EVP\_DecryptUpdate(&ctx, dataOut, &decLen, dataIn+AES\_BLOCK\_SIZE, inLen-AES\_BLOCK\_SIZE);

/\* 对最后一个包含填充在内的加密块解密\*/

EVP\_DecryptFinal\_ex(&ctx, dataOut+ decLen, &decLen);

/\* 释放结构体内存 \*/

EVP\_CIPHER\_CTX\_cleanup(&ctx);

**附件：**

****

以上附件是基于方案概述和关键代码实现的一个数据段类型的敏感数据对称加解密例程。其中DataEncrypt.cpp为源代码，test.txt为程序运行后的结果。

**方案约束：**

此样例基于OpenSSL的加密库，实现对数据段进行加解密。样例中没有考虑到对IV值和原始数据长度的管理。实际应用中，推荐将IV值和原始数据的长度附加到密文上一起保存，解密时先读取IV值和原始数据长度，再对密文进行解密。

此外，实际使用中的加密和解密场景是分开的，此时的数据加密密钥应作为参数或变量传入，并有相应的安全保护措施，不能硬编码在代码中。

### **文件类型的敏感数据加解密**

**场景描述：**

为保护某些机密文件或防止敏感信息泄漏，需要将文件进行加密保存，这样保证只有相应权限的用户才可以查看，即使文件被窃取也不会造成泄密。这些文件包括但不限于:

* 设备间传输的重要配置文件；
* 包含用户私钥的密钥文件、密钥原材料；
* 包含重要信息的用户操作日志；
* ……

**方案概述 ：**

对文件的加密要使用对称加密，推荐使用AES加密算法的CBC模式。为防止文件过大，无法一次性读入内存，可采用将文件分块加密处理的方式，加密后将密文合并到一个文件。IV可附加到文件密文一起保存。

如果是采用用户输入口令方式解密文件，则文件加密密钥可由口令派生出的密钥（如使用PBKDF2函数）进行对称加密后和IV等信息一起保存在文件头或尾部。

**设计要点：**

1. 对文件等大量数据进行加密时，如果没有数据完整性保护的要求，应使用CBC模式。
2. IV不能采用固定数组，需要随机产生。加解密使用同一IV，但对不同文件加密操作的IV不能相同。
3. 分块读取文件内容进行加解密处理后再保存合并到一个文件。

**参考样例：**

下面是一个以OpenSSL库的AES128加密为例的文件加解密样例，包括关键代码和详细例程。文件加密时的IV与原文数据长度保存在文件头中，解密时先从文件头中读出IV，再采用相同的算法和模式对文件内容进行解密。

/\* 文件加密：\*/

EVP\_CIPHER\_CTX ctx;

/\* IV取值为随机数 \*/

RAND\_bytes(IV, AES\_BLOCK\_SIZE);

/\* 初始化加密结构体，设置密钥、IV与加密算法类型AES128-CBC \*/

EVP\_CIPHER\_CTX\_init(&ctx);

EVP\_EncryptInit\_ex(&ctx, EVP\_aes\_128\_cbc(), NULL, pKey, IV);

/\* 保存IV到文件头 \*/

fwrite(IV, sizeof(unsigned char), keySize, fp\_out);

/\* 分块读取加密并保存到文件，FILE\_BLOCK\_SIZE为每次读取的数据块大小 \*/

while (1) {

if ((inLen = fread(pFileBuff, 1, FILE\_BLOCK\_SIZE, fp\_in)) <= 0) break;

EVP\_EncryptUpdate(&ctx, pEncBuff, &outLen, pFileBuff, inLen);

fwrite(pEncBuff, sizeof(unsigned char), outLen, fp\_out); //保存加密数据

}

/\* 最后不足一个加密块时的填充加密并保存\*/

EVP\_EncryptFinal\_ex(&ctx, pEncBuff, &outLen);

fwrite(pEncBuff, sizeof(unsigned char), outLen, fp\_out);

/\* 释放结构体内存 \*/

EVP\_CIPHER\_CTX\_cleanup(&ctx);

/\* 文件解密：\*/

EVP\_CIPHER\_CTX ctx;

/\* 读取IV \*/

fread(IV, sizeof(unsigned char), keySize, fp\_in);

/\* 初始化加密结构体，设置密钥、IV与加密算法类型AES128-CBC \*/

EVP\_CIPHER\_CTX\_init(&ctx);

EVP\_DecryptInit\_ex(&ctx, EVP\_aes\_128\_cbc(), NULL, pKey, IV);

/\* 分块读取解密并保存到文件，FILE\_BLOCK\_SIZE为每次读取的数据块大小 \*/

while (1) {

if ((inLen = fread(pFileBuff, 1, FILE\_BLOCK\_SIZE, fp\_in)) <= 0) break;

EVP\_DecryptUpdate(&ctx, pDecBuff, &outLen, pFileBuff, inLen);

fwrite(pDecBuff, sizeof(unsigned char), outLen, fp\_out);

}

/\* 对最后一个包含填充在内的加密块解密并保存 \*/

EVP\_DecryptFinal\_ex(&ctx, pDecBuff, &outLen);

fwrite(pDecBuff, sizeof(unsigned char), outLen, fp\_out);

/\* 释放结构体内存 \*/

EVP\_CIPHER\_CTX\_cleanup(&ctx);

**附件：**



**方案约束：**

此样例需使用OpenSSL加密库，实际使用中的加密和解密场景是分开的，此时的文件加密密钥应作为参数或变量传入，并有相应的安全保护措施，不能硬编码在代码中。

## **敏感数据非对称加解密**

**场景描述：**

客户端与服务器之间无法使用安全协议或共享密钥建立安全通道时，此时使用非对称加密可以很方便的实现敏感数据的加密传输。但使用非对称加解密时一般只适合少量敏感数据的加密，如：

* 在服务器与客户端间传输一个会话密钥（密钥传输方式建立会话密钥）；
* 网管网元的非安全通道中传输用户口令和业务口令；
* 网管客户端与服务器间网管用户口令与Email口令的传输；
* ……

**方案概述：**

服务器端产生公私钥对，将公钥传给客户端。公私钥对可在程序中产生也可使用外部工具产生。公钥不需要加密，私钥在服务器端需要加密保存。客户端给服务器发送敏感数据时用服务器公钥进行加密，服务器端用私钥进行解密。

**设计要点：**

1. 为保证112位的安全强度，使用RSA算法时密钥长度要为2048及以上。
2. 使用RSA算法时采用OAEP填充方式。OpenSSL中使用OAEP填充方式时输入明文长度必须至少比RSA密钥长度少42字节（OpenSSL中OAEP填充方式默认哈希算法是SHA1），对于2048位的密钥，明文长度最大为2048/8-42=214字节。
3. 使用RSA算法时的公共指数e通常取值为216 + 1，即0x10001。
4. 非对称加密效率通常较低，不适合加密大量数据，大量数据的加密传输可以采用数字信封方式进行。对于长度超过一次加密允许的最大明文长度的情况，在数据量较小时可以对明文进行分割，再对分割后的各明文数据段分别进行非对称加密。

**参考样例：**

下面是一个基于OpenSSL库的使用RSA产生公私钥对并进行非对称加解密的程序样例。

/\* 获取公私钥对，公钥和私钥都在key里。如果从文件中读取密钥，可使用函数PEM\_read\_ PrivateKey()和PEM\_read\_PUBKEY()。\*/

RSA\* key = RSA\_generate\_key(2048, 0x10001, NULL, NULL);

/\* 公钥加密，选择OAEP填充方式 \*/

int encLen = RSA\_public\_encrypt(strlen(plainText), plainText, cipherText, key, RSA\_PKCS1\_OAEP\_PADDING);

/\* 私钥解密 \*/

int decLen = RSA\_private\_decrypt(encLen, cipherText, plainText1, key, RSA\_PKCS1\_OAEP\_PADDING);

**附件：**



**方案约束：**

此样例基于OpenSSL加密库，同时非对称加解密只适合少量敏感数据加密的场景，如需要批量传输敏感数据，建议使用安全协议。

## **敏感数据的数字签名**

**场景描述：**

在实际业务中，为了保证敏感数据的完整性和不可抵赖性，需要对数据进行数字签名。常见的敏感数据如：

* 金融业务中的交易数据；
* 有完整性要求的软件包；
* 安全性要求较高的电子邮件；
* ……

**方案概述：**

通信双方各自产生自己的公、私钥对并将公钥公布出去，公布的方式可以有多种，比如数字证书方式，但如论采用何种方式，都必须保证双方可以互相验证对方公钥的合法性。签名方用自己的私钥加密敏感数据哈希值，加密结果作为签名结果发送给验证方。验证方用相同的哈希算法计算敏感数据的哈希值，并将计算结果和公钥解密的签名结果比较，相同则验证通过，否则不通过。

**设计要点：**

1. 为保证112位安全强度，使用RSA算法时密钥长度应为2048及以上；
2. 使用RSA算法时推荐使用PSS填充方式；

**参考样例：**

以下代码基于OpenSSL，提供了RSA数字签名采用PSS填充方式的参考样例：



## **敏感数据的加密传输**

**场景描述：**

在非信任的网络中，需要进行敏感数据的批量传输，如果可能的话，尽量使用已经成熟的安全标准，而不是自己再构造一套。例如，尽量使用SSL/TLS、SSH、IPSec等安全协议来保护那些敏感数据，而不是自己去构造认证、密钥交换、加密及集成方案。此类场景常见于：

* 需要对语音文件进行加密，并传输；
* 用户名、口令、密钥的传输；
* 前后台通信报文的加密传输；
* HTTPS会话；
* ……

**方案概述 ：**

以下以部署SSL安全通道为例，来进行描述：

假设需要进行敏感数据的批量传输的双方为发送方A、接收方B。则发送方和接收方可以通过部署SSL/TLS协议，建立安全传输通道，来完成敏感数据的加密传输需求。

**设计要点：**

为确保SSL提供了必要的安全性，在部署SSL/TLS协议时，应正确的进行一些配置。下面内容部分参考了《SSL/TLS Deployment Best Practices》

1. 在所有的服务器上，都使用2048位的私钥；
2. 保护私钥，对私钥的可访问范围作最小化。建议的策略包括：

* 将私钥存储到备份系统时，需要用口令加密保护以防止被泄密；
* 一旦发生泄密，吊销所有旧的证书并且为新的证书生成新的密钥对；
* 每年更新证书，而且永远使用新的私钥。

1. 从可信的CA获得证书；
2. 默认情况下使用最新的TLS协议版本，使用一些老的版本保持与用户的互通性。
3. 只使用安全的加密套件。推荐使用128比特或更高安全强度的加密套件，其他的必须限制使用，如：

* 匿名DH(ADH)套件不提供认证功能；
* NULL套件不提供加密功能；
* 出口密钥交换套件使用了容易被破解的认证机制；
* 使用弱加密算法的套件使用的加密算法（如DES、3DES、RC4）很容易被破解。

**参考样例：**

关于SSL/TLS部署的最佳实践的[更详细内容](https://www.ssllabs.com/projects/best-practices/)，可以参考：



## **口令的安全存储**

**场景描述：**

口令是用户登录的重要凭证，即使是系统管理员也不应该知道用户的明文口令。在基于用户口令认证的系统中，在不需要还原口令的场景下，应将口令进行单向哈希保存。这种场景包括但不限于：

* C/S或B/S架构中储存在服务器端的用户登录口令；
* 操作鉴权类口令在服务器端的储存，如支付密码；
* 本地认证的用户口令在本地的储存；
* ……

**方案概述：**

口令单向哈希场景下推荐使用PBKDF2算法，通过输入口令(明文)生成哈希值。 PBKDF2算法（RFC 2898）是一个密钥导出算法，既可用于导出密钥，也可用于口令保存，计算公式：

DK = PBKDF2 (HashAlg, Password, Salt, Count, dkLen)

**输入**：

HashAlg 哈希算法（推荐使用SHA256）

Password 用户输入的口令，字符串

Salt 盐值，字符串

Count 迭代次数，正整数

dkLen 导出密钥的字节长度，正整数

（其中盐值和迭代次数的保存并不需要加密）

**输出：**

DK 导出的密钥，长度为dkLen个字节的字符串。在这里就是最终存储的口令哈希值。

采用PBKDF2算法可以有效增加彩虹表攻击难度。通过增加迭代次数，能显著增加暴力破解的耗时，但对完成一次认证的性能影响不大。

**设计要点：**

1. Hash函数的选择：选择SHA256或更安全的哈希算法。
2. 迭代次数的选择：对于通用操作系统产品至少10000次，对于有性能约束的产品（如认证服务器、嵌入式系统）至少1000次。业界取值参考：

表格 3‑1 业界应用PBKDF2算法的迭代次数

|  |  |  |
| --- | --- | --- |
| 厂家 | 应用领域/产品 | 迭代次数 |
| CISCO | Cisco IOS 15 | 1000 |
| Wifi WAP/WPA2 |  | 4096 |
| Kerberos | RFC 3962 | 4096 |
| Apple | iOS4.x & iOS5.x | 10000 |
| 微软 | office2007 | 50000 |

1. salt的选取：至少8字节；salt应为安全的随机数。
2. PBKDF2的输出长度应该不小于256比特。

**参考样例：**

主流密码算法库都实现了PBKDF2算法，（OpenSSL库中的函数接口为PKCS5\_PBKDF2\_HMAC( )，IPSI库中的函数接口为PKCS5\_deriveKey( )），以下实例基于OpenSSL ，描述了利用PBKDF2生成口令密文的主要过程。

char \*plainPwd = getPassword();

/\* 按照PKCS#5, 盐值不能小于8字节。可以与口令（密文）一起保存 \*/

RAND\_bytes(salt, 16)

/\*迭代次数\*/

int count = 50000;

/\*hash算法：推荐SHA256及更安全的算法。\*/

const EVP\_MD \*digest = EVP\_sha256();

int cipherPwdLen = 32;

/\*计算口令hash值\*/

PKCS5\_PBKDF2\_HMAC(plainPwd, strlen(plainPwd), salt, sizeof(salt), count, digest,

cipherPwdLen, cipherPwd);

**附件：**

****

**方案约束：**

该口令保存方案适用于口令不需要还原的场景。对于口令需要还原（例如：需要记住口令用于自动登录的场景，以及认证协议有约束的场景如digest-MD5认证等）的场景，请参考“敏感数据对称加解密”。

## **非标准安全协议下密钥协商**

**场景描述：**

在不能使用安全协议的情况下，有时需要在通信双方间协商一个会话密钥，建立安全通道。如设备空配置开局的时候，会和一个中心设备交换密钥来建立加密通道完成大包和配置文件升级。这种情况下可使用DH算法进行密钥协商。

**方案概述：**

DH算法是一种在非信任网络中创建共享密钥方法。可以让双方在完全没有对方任何预先信息的条件下通过不安全信道创建一个密钥。这个密钥可以在后续的通讯中作为对称密钥来加密通讯内容。

Alice和Bob想通过DH算法产生一个共享密钥源材料，先协定一组全局公开的参数。公开参数的选择可参考标准FIPS 186-1和X.942的附录A，也可以使用RFC 3526标准中的推荐值，推荐选取Group 14 - Group 18参数组。DH算法具体过程如下图：



图 3‑2 DH算法数据流图

经过图示过程后Alice和Bob就得到了同样的值K，这个值可以被作为共享密钥源材料。协商过程中需对双方交换的临时公钥值进行某些特殊值过滤，如0，1，P-1和P（P为公开参数）。一旦Alice和Bob得出了共享密钥源材料，他们就可以通过密钥导出算法（如PBKDF2）生成对称密钥，给双方通讯内容加密。

**设计要点：**

1. 公开参数的产生需要较长时间，可以直接选用推荐参数并保存到文件，需要时从文件中读取。
2. 为保证安全强度，DH算法的密钥长度要为2048比特及以上。
3. a和b必须使用安全的随机数且长度应大于224比特。
4. 随机数a、,b和最后计算的K在整个过程中是需要保密的，生成密钥后要立即删除。

**参考样例：**

下面是一个基于OpenSSL库的DH密钥协商样例

/\* alice从pem 文件中读取公共参数 \*/

DH \*alice = PEM\_read\_DHparams(fp, NULL, NULL, NULL);

/\* bob从pem文件中读取公共参数 \*/

DH \*bob = PEM\_read\_DHparams(fp, NULL, NULL, NULL);

/\* alice产生密钥对并将其公钥作为临时公钥值发送给bob \*/

DH\_generate\_key(alice);

/\* bob产生密钥对并将其公钥作为临时公钥值发送给alice \*/

DH\_generate\_key(bob);

/\* alice计算共享密钥 \*/

unsigned char\* key1 = (unsigned char \*)OPENSSL\_malloc(DH\_size(alice));

DH\_compute\_key(key1, bob ->pub\_key, alice);

/\* bob计算共享密钥 \*/

unsigned char\* key2 = (unsigned char \*)OPENSSL\_malloc(DH\_size(bob));

DH\_compute\_key(key2, alice ->pub\_key, bob);

**附件：**



**方案约束：**

DH算法是一种匿名交换协议，不对交换双方进行身份认证。因此容易受到中间人攻击威胁。攻击者与通信双方分别使用DH算法协商密钥，并交换其所收到的数据，使通讯的两端认为他们正在对方直接通信，但事实上所有的加密的通信内容都可以被攻击者解密。这里样例没有考虑双方间的身份认证，对于安全要求较高的场景建议使用标准安全协议。

## **软件完整性保护**

**场景描述：**

软件在分发过程中存在被篡改的风险，因此需要有软件完整性保护机制，能够在安装、运行过程中校验软件的完整性。典型要求如：

* 需确保传播的软件和加载的软件就是需要发送的软件；
* 需提供基于数字签名的完整性校验机制，以保证软件完整性和机密性；
* 设备在启动时需要校验软件的完整性；
* ……

**方案概述：**

**软件完整性保护方案**的获取请联系2012实验室 李赤阳（306646）。

# 附录

## **常见密码算法标**准

表格 4‑1 常见密码算法及方案的标准化信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 算法名称 | 标准编号 | 标准名称 | 发布机构 | 发布时间 |
| DES | [FIPS 46-3](http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf) | Data Encryption Standard (DES) | NIST | 1999 |
| AES | [FIPS 197](http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf) | Advanced Encryption Standard | NIST | 2001 |
| 3DES | [SP 800-67 Rev. 1](http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf) | Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher | NIST | 2012 |
| RC4 |  |  | Ron Rivest | 1994 |
| EEA3 |  | Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3.Document 2: ZUC Specification | 3GPP |  |
| SM4 | GM/T 0002-2012 | SM4分组密码算法 | 国家密码管理局 | 2012 |
| RSA | [PKCS#1V2.2](http://www.rsa.com/rsalabs/pkcs/files/h11300-wp-pkcs-1v2-2-rsa-cryptography-standard.pdf) | PKCS#1 V2.2 RSA Cryptography Standard | RSA Lab | 2012 |
|  | [RFC3447](http://www.ietf.org/rfc/rfc3447.txt) | Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 | IETF | 2003 |
| DSA | [FIPS 186-3](http://csrc.nist.gov/publications/PubsDrafts.html#FIPS-186--3%20Proposed%20Change) | Digital Signature Standard (DSS) | NIST | 2009 |
|  | [FIPS 186-3 Proposed Change](http://csrc.nist.gov/publications/PubsDrafts.html#FIPS-186--3%20Proposed%20Change) | DRAFT Proposed Change Notice for Digital Signature Standard (DSS) | NIST | 2012 |
| ECDSA | ANS X9.62-2005 | Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). | ANSI | 2005 |
| SM2 | GM/T 0003-2012 | SM2椭圆曲线公钥密码算法 | 国家密码管理局 | 2012 |
| DH | [RFC2631](http://www.ietf.org/rfc/rfc2631.txt) | Diffie-Hellman Key Agreement Method | IETF | 1999 |
|  | [PKCS#3 V1.4](ftp://ftp.rsasecurity.com/pub/pkcs/doc/pkcs-3.doc) | Diffie-Hellman Key Agreement Standard | RSA Lab | 1993 |
|  | [IEEE Std. 1363-2000](http://grouper.ieee.org/groups/1363/P1363/) | Standard Specifications for Public Key Cryptography | IEEE | 2000 |
| SHA1 | [FIPS 180-4](http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf) | Secure Hash Standard (SHS) | NIST | 2012 |
|  | [RFC3174](http://www.ietf.org/rfc/rfc3174.txt) | US Secure Hash Algorithm 1 (SHA1) | IETF | 2001 |
| SHA2 | [FIPS 180-4](http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf) | Secure Hash Standard (SHS) | NIST | 2012 |
| MD5 | [RFC1321](http://www.ietf.org/rfc/rfc1321.txt) | The MD5 Message-Digest Algorithm | IETF | 1992 |
| SM3 | GM/T 0004-2012 | SM3密码杂凑算法 | 国家密码管理局 | 2012 |
| HMAC | [FIPS 198-1](http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf) | The Keyed-Hash Message Authentication Code (HMAC) | NIST | 2008 |
|  | [RFC2104](http://www.ietf.org/rfc/rfc2104.txt) | HMAC: Keyed-Hashing for Message Authentication | IETF | 1997 |
| 工作模式 | [SP 800-38 A](http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf) | Recommendation for Block Cipher Modes of Operation - Methods and Techniques | NIST | 2001 |
|  | [SP 800-38 A - Addendum](http://csrc.nist.gov/publications/nistpubs/800-38a/addendum-to-nist_sp800-38A.pdf) | Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode | NIST | 2010 |
|  | [SP 800-38 B](http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf) | Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication | NIST | 2005 |
|  | [SP 800-38 C](http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf) | Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality | NIST | 2004 |
|  | [SP 800-38 D](http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf) | Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC | NIST | 2007 |
|  | [SP 800-38 E](http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf) | Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices | NIST | 2010 |
|  | [SP 800-38 F](http://csrc.nist.gov/publications/drafts/800-38F/Draft-SP800-38F_Aug2011.pdf) | Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping | NIST | 2012 |
| PBE基于口令的加密 | [PKCS#5 V2.1](http://www.rsa.com/rsalabs/pkcs/files/h11302-wp-pkcs5v2-1-password-based-cryptography-standard.pdf) | PKCS #5: Password-Based Cryptography Standard | RSA Lab | 2012 |
|  | [RFC2898](http://www.ietf.org/rfc/rfc2898.txt) | PKCS #5: Password-Based Cryptography Specification Version 2.0 | IETF | 2000 |
| 消息格式 | [PKCS#7 V1.5](ftp://ftp.rsasecurity.com/pub/pkcs/doc/pkcs-7.doc) | PKCS #7: Cryptographic Message Syntax Standard | RSA Lab | 1993 |
|  | [RFC3852](http://www.ietf.org/rfc/rfc3852.txt) | Cryptographic Message Syntax (CMS) | IETF | 2004 |
| KDF密钥导出函数 | [PKCS#5 V2.1](http://www.rsa.com/rsalabs/pkcs/files/h11302-wp-pkcs5v2-1-password-based-cryptography-standard.pdf) | PKCS #5: Password-Based Cryptography Standard | RSA Lab | 2012 |
|  | [RFC5869](http://tools.ietf.org/html/rfc5869) | HMAC-based Extract-and-Expand Key Derivation Function (HKDF) | IETF | 2010 |
|  | [SP 800-108](http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf) | Recommendation for Key Derivation Using Pseudorandom Functions | NIST | 2009 |

## **分组密码算法工作模式**

### 常见工作模式介绍

本节中用到的符号和函数列表。

|  |  |
| --- | --- |
| **符号** | **解释** |
| **Cj** | 第j次计算的密文分组 |
| **C#j** | CFB模式中第j次计算的密文数据段 |
| **C\*n** | 最后一个密文分组，可能仅含分组的一部分 |
| **Pj** | 第j次计算的明文分组 |
| **P#j** | CFB模式中第j次计算的明文数据段 |
| **P\*n** | 最后一个明文分组，可能仅含分组的一部分 |
| **Ij** | 第j次计算的输入分组 |
| IV | 初始向量 |
| **Oj** | 第j次计算的输出分组 |
| **Tj** | CTR模式第j个计数器 |

|  |  |
| --- | --- |
| **函数** | **解释** |
| **X | Y** | 比特串X后连接比特串Y |
| **X** ⊕ **Y** | 比特串X异或比特串Y |
| **CIPHK**(X) | 密钥K对分组X的加密 |
| **CIPH-1K(X)** | 密钥K对分组X的解密 |
| **LSBm(X)** | 比特串X的最低m比特 |
| **MSBm(X)** | 比特串X的最高m比特 |

#### 电子密码本模式(Electronic Code Book，ECB)

ECB模式是最简单和最快的分组密码工作模式，也是最弱的。一个明文分组加密成对应的一个密文分组；在密钥相同的情况下，相同的明文分组永远被加密成相同的密文分组。

* **基本原理**

将需要加密的信息分为大小合适的分组，然后分别对每一分组独立进行加密或解密处理。原理如下图所示：

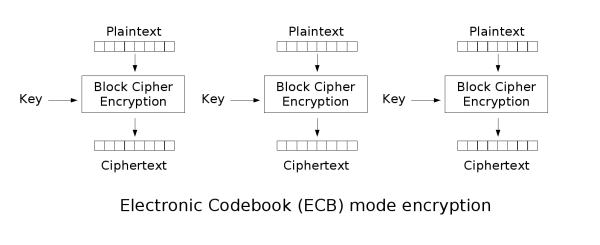


图 4‑1 ECB模式下明文的加密图

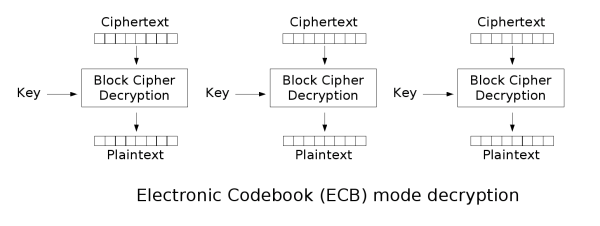


图 4‑2 ECB模式下密文的解密图

若第一个块的下标为1，则ECB模式的加解密过程用数学语言可表示为：

加密：**Cj = CIPHK(Pj)**

解密：**Pj = CIPH-1K(Cj)**

#### 密码分组链接模式(Cipher Block Chaining，CBC)

密码分组链接将一种反馈(Feedback)机制加入到分组密码中，前一个分组的加密结果被反馈到当前分组的加密中；即每一分组被用来修改下一个分组的加密。CBC模式中的初始化向量(IV)不允许重复使用，必须每次使用不同的不可预测的IV。

* **基本原理**

在CBC模式中，每个明文块与前一个密文块进行异或后再进行加密。在这种方式中，每个密文块都依赖于它前面的所有明文块。同时，为了保证每条消息的唯一性，在第一个明文块中需要使用初始化向量IV来与它进行异或。原理如下图所示：

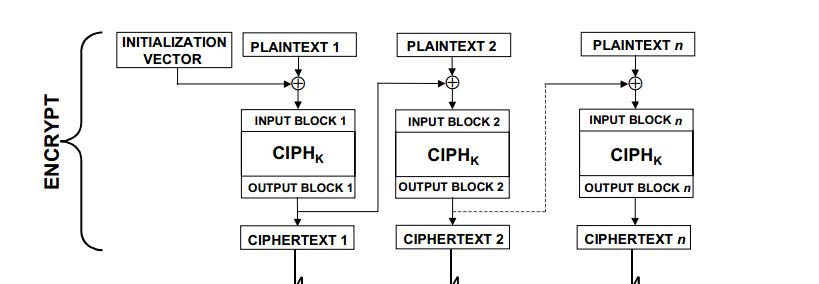


图 4‑3 CBC模式下的加密

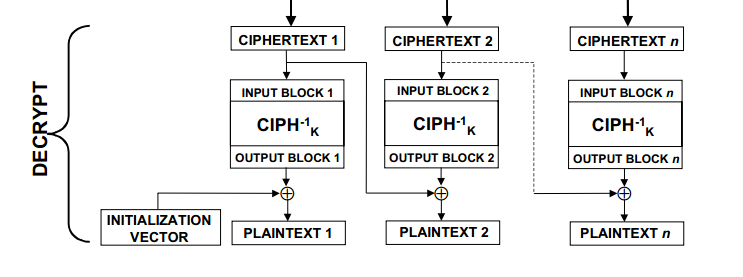


图 4‑4 CBC模式下的解密

假设第一个块的下标为1，则CBC模式的加解密过程用数学语言可表示为：

加密： 解密：

**C1 = CIPHK(P1⊕IV) P1=CIPH-1K(C1) ⊕ IV**

**Cj = CIPHK(Pj⊕Cj-1 ) Pj = CIPH-1K(Cj ) ⊕ Cj-1**

#### 密码反馈模式(Cipher FeedBack，CFB)

与CBC模式类似，CFB模式也需要将前一个分组的加密结果反馈到当前分组的加密中，每一个分组的加密结果会被用来修改下一个分组的加密。CFB模式每个分组运算时参与运算的明文（加密时）或密文（解密时）的长度为CFB数据段的长度。数据段的长度不大于分组长度，对于AES，数据段大小可以是1比特、8比特、64比特或128比特等，分别对应AES-CFB1（1比特实现）、AES-CFB8（8比特实现）、AES-CFB64（64比特实现）或AES-CFB128（128比特实现）。CFB模式中的初始化向量IV必须不可预测。

* **基本原理**

假设明文数据段(Plaintext)的大小为s Bits，每次加密后得到的密文(Ciphertext)的大小也为s Bits。每次输入加密函数的数据块的大小（分组大小）为b Bits，初始化向量(IV)的大小也为b Bits。

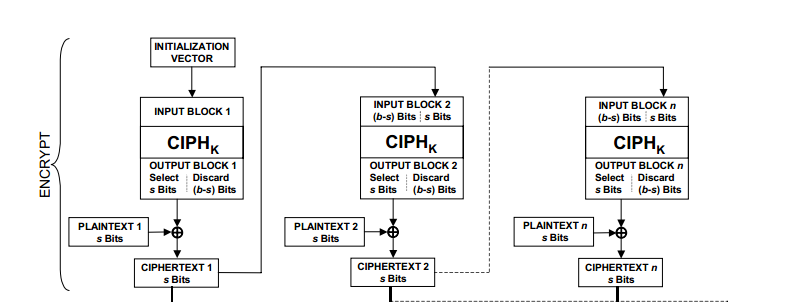


图 4‑5 CFB模式下的加密

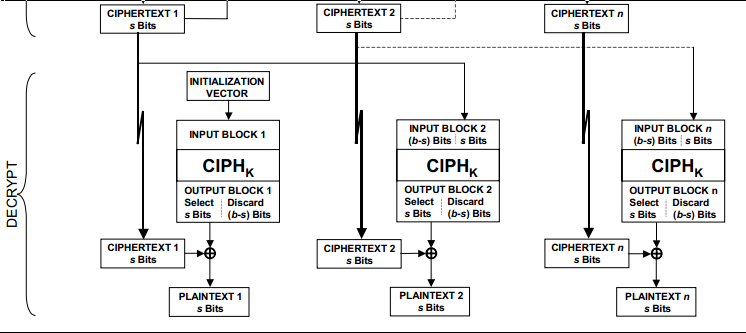


图 4‑6 CFB模式下的解密

若第一个块的下标为1，则CFB模式的加解密过程用数学语言可表示为：

加密： 解密：

**I1 = IV; I1 = IV**

**Ij = LSBb-s(Ij-1) | C#j-1 Ij = LSBb-s(Ij-1) | C#j-1**

**Oj = CIPHk(Ij) Oj = CIPHK(Ij)**

**C#j = P#j ⊕ MSBs(Oj) P#j = C#j ⊕ MSBs(Oj)**

解密和加密过程很相似，只是在解密的时候是密文与输出块的最高s Bits进行异或得到明文。

#### 输出反馈模式(Output FeedBack，OFB)

与CFB模式相似，OFB模式也可以将分组密码转化成流密码。不同的是CFB模式将密文单元反馈到下一个分组中，而OFB模式则是将加密算法的输出反馈到下一个分组中。OFB模式中的初始化向量(IV)不允许重复使用，必须每次使用不同的IV。

* **基本原理：**

1. 初始化向量IV作为第一个输入块(Input Block 1)，通过加密算法运算后，得到第一个输出块(Output Block 1)。第一个输出块(Output Block 1)与第一个明文分组(Plaintext 1)进行异或后，得到第一个密文分组(Ciphertext 1)。
2. 第一个输出块(Output Block 1)作为第二个输入块(Input Block 2)，通过加密运算后，得到第二个输出块(Output Block 2)。第二个输出块(Output Block 2)与第二个明文分组(Plaintext 2)进行异或后，得到第二个密文分组(Ciphertext 2)。
3. 其他明文块进行相同的处理。

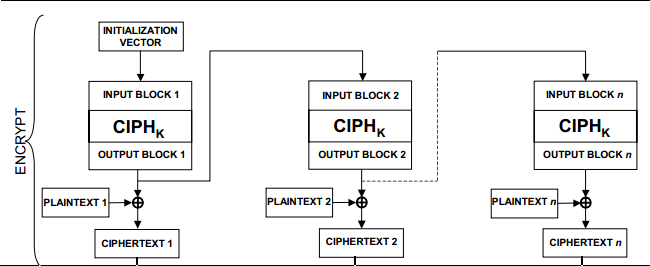


图 4‑7 OFB模式下的加密

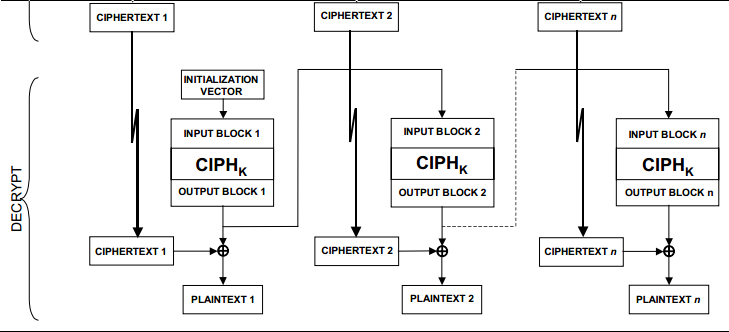


图 4‑8 OFB模式下的解密

若第一个块的下标为1，则OFB模式的加解密过程用数学语言可表示为：

加密： 解密：

**I1 = IV; I1 = IV**

**Ij = Oj-1 Ij = Oj-1**

**Oj = CIPHk(Ij) Oj = CIPHk(Ij)**

**Cj = Pj ⊕ Oj Pj = Cj⊕Oj**

**C\*n = P\*n ⊕ MSBu(On) P\*n = C\*n ⊕MSBu(On)**

#### 计数器模式(Counter Mode，CTR)

CTR模式，也称为ICM模式（Integer Counter Mode，整数计算模式），与OFB模式相似，它通过递增一个加密计数器来产生连续的密钥流。由于加密和解密过程均可以进行并行处理，CTR模式适用于多处理器的硬件。

使用同一个密钥加密数据时，把待加密的明文数据称为消息，每个消息又被分为若干等长的数据块（Block），除最后一块数据块外其余数据块的长度均等于对称密码算法的分组长度（最后一块数据块长度有可能小于密码算法分组长度）。对于每一个数据块，加密时都需要一个唯一的（各不相同）计数块（Counter，或称计数器）。如果使用相同的密钥加密，CTR模式要求无论加密多少个消息（每个消息包括多个数据块），所有用到的计数块（Counters，或称计数器）都必须各不相同。

通常来说，为了保证计数块的唯一性，计数块的生成按照下面的方法：

1. 设计一个递增函数，该递增函数可以根据任何初始计数块（加密消息时用到的第一个计数块）来产生剩余的计数块，并且这些产生的计数块不会重复，递增函数用于保证对同一消息加密时用到的各个计数块的唯一性；
2. 消息间计数块的唯一性由各个消息的初始计数块来保证；

实际操作建议：以AES为例，假设分组大小为16字节，则可按照如图所示方式生成计数块。每加密一条消息时都生成一个8字节的随机数，消息内部各个计数块的前8字节都使用这个随机数，而消息内部的后8个字节则按照计数块序号顺序递增，如第一块8字节存放整数0，第二块8字节存放整数1等。这样，前8字节随机数保证了各个消息间初始计数块的唯一性，而后8字节按计数块序号递增则保证了消息内部各计数块的唯一性。



图 4‑9 CTR模式Counter构造方法示意图

* **基本原理**

计数器1(Counter 1)作为第一个输入块(Input Block 1)，经过加密算法进行运算后，得到对应的输出块(Output Block 1)。第一个输出块(Output Block 1)与第一个明文分组(Plaintext 1)进行异或后，得到第一个密文分组(Ciphertext 1)。



图 4‑10 CTR模式下的加密

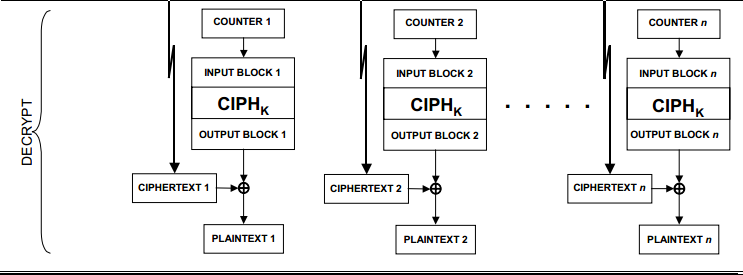


图 4‑11 CTR模式下的解密

若第一个计数器的下标为1，则CTR模式的加解密过程用数学语言可表示为：

加密： 解密：

**Oj = CIPHK(Tj) Oj = CIPHK(Tj)**

**Cj = Pj⊕Oj Pj = Cj⊕Oj**

**C\*n = P\*n ⊕ MSBu(On) P\*n = C\*n ⊕MSBu(On)**

CTR模式的加密和解密过程都可以进行并行运算。

在计数器模式中，每个块序列必须与其他块序列不同；如果两个消息流是用同一密钥加密，则两个消息的块序列也必须不同。

#### 基于分组密码的消息认证码(Cipher-based Message Authentication Code, CMAC)

CMAC是一种基于对称密钥分组密码(如AES算法)的消息认证码算法，因此，CMAC同样也可以看做是一种分组密码的工作模式。

#### 计数器与CBC\_MAC混合模式(Counter With CBC\_MAC, CCM)

CCM模式其实是一种用CTR模式来提供保密服务，并用CBC\_MAC模式来提供认证服务的混合模式。它结合了这两个模式的各自的优点，提出了一个既能加密又能认证的新模式。

#### 伽罗华计数器模式(Galois/Counter Mode, GCM)

GCM是一种使用计数器模式(CTR)和二元有限域上的泛哈希函数来提供认证加密的分组密码工作模式。GCM包括CTR模式的加密和采用泛哈希的MAC(消息认证码)两部分。GCM工作模式由于具有高速、低时延、没有专利限制等优点，应用于高速接入网络中具有明显优势。

#### 基于XEX和密文窃取技术的TCB模式(XEX Tweakable Block Cipher with Ciphertext Stealing for AES, XTS-AES)

XTS-AES是一种专用于存储设备加密的工作模式。IEEE于2007提出相关标准IEEE P1619， NIST指定其作为存储设备数据加密的推荐工作模式。XTS-AES模式中一个扇区可分成多个加密块，各加密块之间独立，且加密块的加密输入包含了索引信息（扇区号和块号）。因此使用XTS-AES模式时加密块可以随机访问，其加密解密也是可以并行化的。

### 常见工作模式对比分析

表格 4‑2 常见工作模式对比分析表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **常用工作模式** | **模式简述** | **优势** | **劣势** | **应用场景/典型应用** |
| 电子密码本 ECB | 用相同的密钥分别对明文分组加密 | * 简单、高效 * 可并行处理 * 加密数据块的随机访问 * 不存在错误传播 * 不需要额外的初始向量 | * 安全性差，易受重放攻击和替换攻击 * 只适用于短数据加密 | N/A |
| 密码分组链接  CBC | 加密算法的输出是上一个密文分组和下一个明文分组的异或 | * 明文块的统计特性得到了隐蔽， 安全性较高 * 具有自同步功能 | * 有限的（2块）错误传播 * 需要额外的初始向量 * 串行处理 | 适用于明文不易丢信号，对明文的格式没有特殊要求等场景   * 数据加密 |
| 密码反馈 CFB | 一次处理j位，上一分组密文作为加密算法的输入，用产生一个伪随机数输出与密文异或作为下一个分组的输入 | * 明文块的统计特性得到了隐蔽， 安全性较高 * 具有自同步功能 * 具有认证功能 * 安全性较高 * 可处理任意长度的消息 | * 效率较低 * 存在错误传播 * 需要额外的初始向量 | 适应于数据库加密、无线通信加密等对数据格式有特殊要求或密文信号容易丢失或出错的应用环境 |
| 输出反馈 OFB | 与CFB基本相同，只是加密算法的输入是上一次加密的输出 | * 无错误传播 * 可以预处理 * 可处理任意长度消息 | * 不能实现完整性认证 * 易受篡改攻击 * 需要额外的初始向量 | 适用于噪声信道上的数据流的传输(明文的冗余度特别大，信道不好，明文错些信号也不影响效果)的情形，如卫星通信、图象加密、语音加密等 |
| 计数器  CTR | 每个明文分组都与一个加密计数器异或，对每个后续计数器递增 | * 简单、高效 * 可并行处理和预计算 * 加密数据块的随机访问 * 具有可证安全性 * 不需要额外的初始向量 * 不存在错误传播 | * 没有错误传播，不易确保数据完整性 | * 普通目的的面向分组的传输 * 用于高速需求 |
| 基于分组的消息认证码  CMAC | CMAC是一种基于对称密钥分组密码(如AES算法)的消息认证码算法 | * 设计简单 * 提供消息认证功能 | * 不能进行预处理操作 | * 提供认证服务的场景 |
| 计数器与CBC-MAC混合模式  CCM | 用CTR模式来提供保密服务，并用CBC\_MAC模式来提供认证服务的混合模式 | * 高效 * 采用CBC-MAC模式提高了产生MAC值的安全性 * 提供加密、认证功能 | * 加密前必须知道消息长度 * 不能进行预处理操作 | * IEEE 802.11的无线局域网的加密工作模式，已在RFC 3601标准化 |
| 伽罗华计数器  GCM | 一种使用计数器模式(CTR)和二元有限域上的泛哈希函数来提供认证加密的分组密码工作模式 | * 高速率 * 硬件实现低成本、低延时 * 提供加密、认证 |  | * 高速光纤接入网 |
| XTS-AES模式 | 一种基于XEX和密文窃取技术的调整密码本模式（TCB），用于存储设备加密 | * 可并行处理 * 加密数据块的随机访问 | * 不提供认证功能 * 不适用于传输加密 | * 存储设备加密 |

# 参考文献

1. NIST Special Public 800-57, Recommendation for Key Management – Part1: General(Revision3), 2012
2. NIST Special Public 800-21-1, Guideline for Implementing Cryptography in the Federal Government, 2005
3. NIST Special Public 800-38a, Recommendation for Block Cipher Modes of Operation - Methods and Techniques, 2001
4. ECRYPT II Yearly Report on Algorithms and Key Lengths (2012), revision 1.0, 2012
5. RFC4086, Randomness Requirements for Security, 2005
6. RFC2898, PKCS #5: Password-Based Cryptography Specification Version 2.0, 2000
7. RFC7465, Prohibiting RC4 Cipher Suites, 2015
8. Deutsche Telekom Group 3.50, Security Requirement Cryptographic Algorithms and IP Security(IPSec) Version 2.0, 2014
9. Qualys SSL Labs, SSL/TLS Deployment Best Practices Version 1.3, 2013

1. 又称为柯尔霍夫原则：“一个密码系统应该在除密钥以外系统的任何部分都被公知的情况下，系统仍要安全” [↑](#footnote-ref-1)
2. 可参考连接：<http://crypto.stanford.edu/~dabo/pubs/papers/RSA-survey.pdf> [↑](#footnote-ref-2)
3. 可参考链接：<https://www.kb.cert.org/vuls/id/536044> [↑](#footnote-ref-3)
4. IETF TLS-1.3 Draft02开始不再支持静态RSA和静态DH密钥交换 [↑](#footnote-ref-4)
5. 可参考链接：<http://en.wikipedia.org/wiki/Length_extension_attack>

   <http://crypto.stackexchange.com/questions/1070/why-is-hkx-not-a-secure-mac-construction> [↑](#footnote-ref-5)
6. 基于SHA-3的KMAC/XMAC目前正在被标准化 [↑](#footnote-ref-6)
7. 可参考链接：<http://en.wikipedia.org/wiki/CBC-MAC> [↑](#footnote-ref-7)