GM/Y 5007-2024

基于 SM4 密码算法的保留格式加密技术研究

密码行业标准化技术委员会
CRYPTOGRAPHY STANDARDIZATION TECHNICAL COMMITTEE

2024 年 12 月

摘要

随着计算机与网络技术的发展，数据库存储和网络传输中存在大量敏感信息，如用户名、密码、身份证号和信用卡号等等这些具有隐私性不应公开的信息都属于敏感信息。这些敏感信息的保护一直都是困扰数据库使用和管理的难题，每年都有大量的关于敏感信息的泄露造成数千万甚至更多用户的严重损失的事件，增强敏感信息的安全性已成为需要迫切解决的问题。保护敏感信息最有效的方法就是加密，然而使用传统分组加密方式加密数据通常会使得数据长度和类型发生改变，导致数据库结构和应用程序的修改，理想的办法是采用保留格式加密（FPE: format preserving encryption）。FPE 是一种全新的密码学技术，它将一种特定格式的明文加密成相同格式的密文，即密文的类型和长度与明文相同。

本研究报告首先对 FPE 的背景、定义以及典型应用进行了介绍，其次对 FPE 的三种典型方案进行总结，并且对方案的安全性和方案的模型给出了分析。然后研究报告对于 FPE 的标准化情况进行了介绍，提出了基于商用密码（SM4）的保留格式加密算法。最后给出了基于 SM4 的保留格式加密的测试向量。

关键词：保留格式加密、Feistel 网络、SM4

目录

前言

本研究报告是由密码行业标准化技术委员会根据国家密码管理局批准的《2018 年密码行业标准制订计划（商用密码领域）》下达的密码行业标准研究编制工作任务。项目名称为《基于国密算法的保留格式加密（FPE）研究》，项目类型为研究类项目，项目所属工作组为基础工作组。在后续编制的过程中，经过专家的讨论，项目名称更改为《基于 SM4 密码算法的保留格式加密技术研究》。格尔软件股份有限公司所作为牵头单位，成立相应的编制工作组，组织完成该标准研究报告的编制工作。

本报告起草单位：格尔软件股份有限公司、中电科网络安全科技股份有限公司、三未信安科技股份有限公司、北京炼石网络技术有限公司、北京海泰方圆科技股份有限公司、飞天诚信科技股份有限公司、北京数字认证股份有限公司、深圳奥联信息安全技术有限公司、北京信安世纪科技股份有限公司、安徽问天量子科技股份有限公司、兴唐通信科技有限公司、北京握奇智能科技有限公司。

本报告主要起草人员：郑强、张立廷、潘文伦、刘会议、白小勇、蒋红宇、罗影、朱鹏飞、傅大鹏、李向锋、程朝辉、汪宗斌、刘婧婧、王妮娜、张渊。

基于 SM4 密码算法的保留格式加密技术研究

1. 概述

1.1 背景

2017 年 6 月 1 日《中华人民共和国网络安全法》（简称《网络安全法》）正式实施。《网络安全法》框架性地构建了许多法律制度和要求，重点包括个人信息和重要数据保护制度、关键信息基础设施安全保护制度等。为了保障法律的有效实施，国家互联网信息办公室为主的监管部门制定了多项配套法规，进一步细化和明确了各项制度的具体要求；另外，全国信息安全标准化技术委员会同时制定并公开了一系列以信息安全技术为主的重要标志作为配套标准规范。

在 GB/T 35273—2020《信息安全技术　个人信息安全规范》（以下简称《个人信息安全规范》）标准中，个人信息安全的隐私保护的重要程度再次提升。在《个人信息安全规范》中，个人信息所可能涉及的各个环节，包括收集、保存、传输等，均有严格的标准依据。其中，在个人信息保护部分，《个人信息安全规范》明确指出，个人信息在保存时应当经过去标识化处理，以保证个人信息的隐私属性，提高个人信息使用的安全程度。2019 年发布的 GB/T 37964—2019《信息安全技术 个人信息去标识化指南》中将保留格式加密技术作为常用的去标识化技术。2016 年美国 NIST 标准组织将保留格式加密作为标准发布，可见对保留格式加密技术及其标准化进展进行研究对于保障网络空间安全具有重大意义。

保留格式加密（Format-Preserving Encryption，FPE）是信息安全和隐私保护领域的一个新兴分支，是一类能够将某种特定格式的明文加密成具有相同格式的密文的新型加密技术，自 20 世纪 80 年代提出以来，得到了广泛的关注和研究。国内外研究学者发表了多篇关于 FPE 的研究成果，推动了 FPE 在各阶段的发展。对于 FPE 的研究大体可以从基本算法研究、加密模型研究以及应用场景三个方向进行开展。

基本算法研究方向致力于基本类型数据保留格式加密算法的探索，如 Prefix 算法、Cycle-walking 算法和 Generalized-Feistel 算法；加密模型研究方向主要在基本构建方法的基础上，试图解决更复杂的保留格式加密问题，提出更高效的 FPE 模型，如 FFSEM、RtE、FFX、BPS 和 CtE 等；应用场景研究方向主要是根据具体应用场景的要求和特定数据的格式，来设计特定的保留格式加密算法。

1.2 FPE 定义

根据已有的研究成果，从两个方面对 FPE 进行了定义，包括基本 FPE 和一般化 FPE。前者强调密文和明文具有相同的格式，即确保密文和明文属于相同的消息空间；后者则强调待加密消息空间的复杂性决定 FPE 问题的复杂性。

1）基本 FPE：可简单描述为一个密码：

$$E{:}K \times X \to X$$

其中，K 为密钥空间，X 为消息空间。

基本 FPE 描述了 FPE 要解决的问题，即密文与明文处于相同的消息空间。如，对 n 位的信用卡号进行保留格式加密，要求密文与明文的格式相同，都是 n 位十进制数字组成的字符串。即明文与密文都在消息空间$\{0,1,\ldots,9\}^n$内。根据基本 FPE 的定义，分组密

码也是某种形式的 FPE，分组密码是字符串集合$\{0,1\}^n$上的置换。但是，FPE 的消息空间的复杂性要远远高于分组密码，如日期型的消息空间“YYYY-MM-DD”，其格式不仅有长度为 10 的限制，且年、月、日必须在合法范围内等特定格式的要求。

为了更完整地描述 FPE 问题，定义集合 Ω 为格式空间，任意一个给定的格式ω∈Ω，可确定一个由ω确定的消息空间 X 的子空间X_ω，FPE 与集合$\{X_\omega\}_{\omega\epsilon\Omega}$有关。$X_\omega$称为由格式ω确定的待加密消息空间 X 上的一个分片，每个分片X_ω都是一个有限集。给定密钥k，格式ω和调整因子t，FPE 就是一个定义在X_ω上的置换$E_k^{\omega,t}$。

2）一般化 FPE：可描述为一个密码：

$$E: K \times \Omega \times T \times X \to X \cup \{\perp\}$$

其中，K为密钥空间，Ω 为格式空间，T为调整因子空间，X为消息空间。所有空间非空且$\perp \notin X$。

可通过算法三元组$E_{FPE} = (setup, encryption, decryption)$来描述一般化 FPE，其中：

算法$setup$：

初始化系统参数$params$。不同的 FPE 模型需要初始化不同的参数，通常包括以下 3 个部分：

- 初始化对称密码算法（要求对称密码足够安全）的参数，比如 Feistel 网络的分组长度、轮函数和轮次数等；
- 初始化待解决的问题域，包括需要保留的格式ω以及由ω确定的消息空间分片X_ω；
- 初始化加（解）密使用的对称密钥k，要求安全存储该密钥，不对外公开。

算法$encryption$：

输入明文x和调整因子$t \in T$，返回消息空间分片X_ω内的密文y或者⊥。该算法执行$E_K^{\Omega,T}(X) = E(K,\Omega,T,X)$，$E_K^{\Omega,T}(\cdot)$是$X_\Omega$上的一个置换。如果$x \in X_\omega$，则返回$y = E_k^{\omega,t}(x)$，否则返回⊥。

算法$decryption$：

输入密文y和调整因子$t \in T$，返回相同消息空间分片X_ω内的明文x或者⊥。该算法是$encryption$的逆运算：如果$y \in X_\omega$，则返回$x = D_k^{\omega,t}(y)$，否则返回⊥。

1.3 FPE 典型应用场景

FPE 最初是为了解决数据库或应用系统中的敏感信息加密问题，自 2008 年美国 Voltage 公司公布其安全产品所使用的 FPE 技术的白皮书，FPE 得到了更多研究学者的关注，发展成为极具实用价值的密码学体系。由于 FPE 保持密文与明文具有相同格式的特性，因此，适用于格式敏感的数据加密领域。

在加密模型方面，提出的典型模型（如 FFSEM，RtE 与 FFX 模型）为特定领域的 FPE 问题提供了较好的解决办法。在应用研究方面，目前已提出了适用于支付卡行业安全的信用卡号等 FPE 方案、适用于数据库加密的日期型 FPE 方法、满足格式兼容要求的 JPEG2000 加密等实用的解决方案，开拓了 FPE 在不同领域的应用研究。

1.3.1 支付标记化

1）应用背景

PCI DSS（Payment Card Industry Data Security Standard，支付卡行业数据安全标准）是一套被广为接受的政策和程序，目的是为了保证信用卡、借记卡和现金卡交易的安全，保护持卡人的个人信息，防止被他人利用。PCI DSS 所规定和阐述的六大目标之一就是，持卡人信息无论存放在哪里，都必须受到保护，确保存放的社会保险号和身份证号等重要数据不被破解。当持卡人的数据通过公共网络进行传送时，这些数据必

须经过有效的加密。

数据加密技术在各种形式的信用卡交易中的重要性无需赘言，金融交易过程很复杂、约束也很多，传统的分组密码加大了改变这些系统的复杂度，代价昂贵。而 FPE 因为实现简单，保留敏感信息的格式，避免了从根本上去设计和审查整个系统的繁杂。

支付标记化（Tokenization）作为 FPE 的一种典型应用，可以在产业中跨行使用，具有通用性，可应用于支付卡行业的数据保护，满足 PCI DSS 的要求。支付标记化就是使用支付标记（token）来替代传统的银行卡主账号进行交易验证，从而避免卡号信息泄露带来的风险。

其中，支付标记（token）是指主账号（PAN）的一个替代值，一般由 13 至 19 位的数字组成，该数值必须符合主账号的基本验证规则，其中包括 LUHN 算法校验。在银行卡支付交易中用支付标记替换卡号，用支付标记的有效期替换卡号有效期，不影响交易处理，增强了交易安全。

相比银行卡号，支付标记的外在形式没有变化，仍是 13-19 位数字，前 6 位是卡组织分派的 BIN；内在属性也没有减少，同样具备卡的性质，遵循同样的校验位规则，同样区分借贷记，同样拥有失效期。

2）技术架构

在支付标记化系统中，描述了支付标记化的主要角色及关系，标记请求方（TR）与标记服务提供方（TSP），两个新增角色与现有传统支付流程的关系和数据交互接口，明确了支付标记如何共同为用户提供标记服务，如下 0 示，

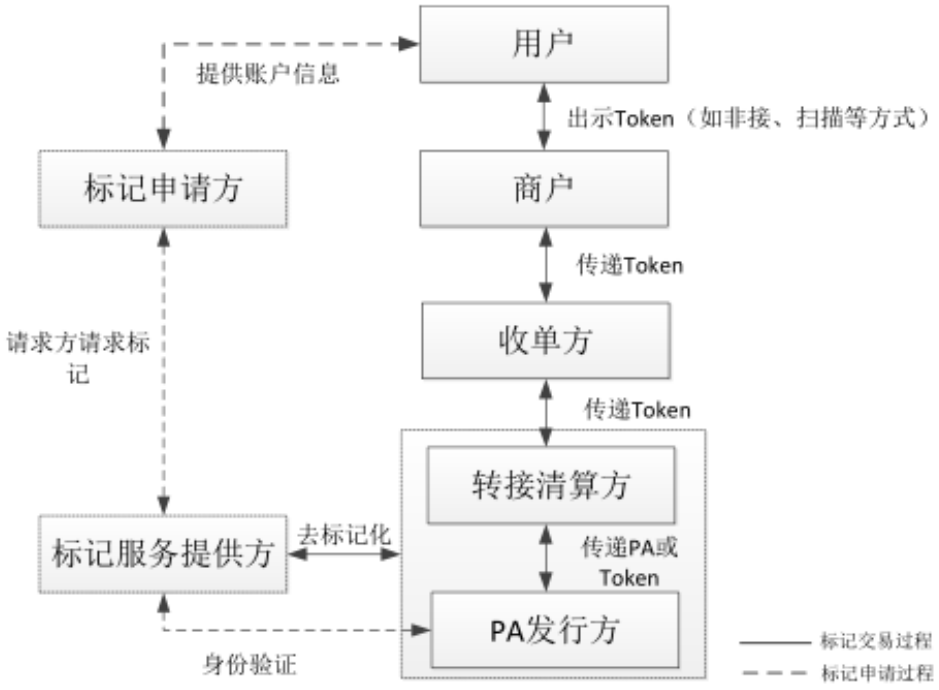


图 1-1 支付标记化的技术架构

其中，TSP 是该标记化框架的核心角色，它提供了 PA 标记的申请、生成、管理、去标记化等功能，包括 TR 的注册和管理职责。根据不同的业务场景、受理渠道以及标记的应用域控，TSP 应制定与之配套个性化参数和控制措施，最终达到标记交易控制和风险监控。

TR 作为标记请求的实体向 TSP 申请 PA 标记。TR 必须向 TSP 注册，由 TSP 向 TR 分配唯一的 TR ID。一般来说，承担 TR 角色的实体为转接清算机构、发卡机构、商户、非

银行支付机构、电信运营商、手机终端制造商或者互联网移动支付终端设备制造商等。

3）优势

支付标记化技术作为全球支付领域的最新前沿技术，其优势有：

首先，敏感信息无需留存，持卡人卡号与卡片有效期在交易中不出现；

其次，支付标记仅可在限定交易场景使用，使得支付更安全；

最后，支付标记灵活性更高，与传统银行卡验证功能相比较，支付标记综合了个人身份与设备信息验证、支付信息附加验证、风险等级评估等功能进行交易合法性识别和风险管控。

因此，支付标记化不仅可防范交易各环节的持卡人敏感信息泄露，同时，又降低了欺诈交易的发生概率。

1.3.2 涉及相关个人信息的存储

在实际应用中，对数据库中的信用卡号、身份证号、手机号等敏感数据进行加密非常必要，而数据脱敏也是常规的数据库安全技术之一。数据脱敏指对某些敏感数据通过脱敏规则对敏感信息字段进行转换和覆盖，变换为与原始信息具有相同业务规则、代表实际业务属性而不具有真实业务功能的虚构信息，实现敏感隐私数据的可靠保护。

数据脱敏技术本质上是对于数据的变形处理，因此数据脱敏技术的另一大特点是能够在一定程度上保持数据原本的一些特性，使脱敏后的数据依旧存在可用性。适当地使用数据脱敏技术，可以有效地减少敏感数据在采集、传输、使用等环节中的暴露，降低敏感数据泄露的风险，尽可能降低数据泄露造成的危害。[30]中提出了几种在数据脱敏中常用到的数据格式的保留格式加密的方法。

1）关于银行卡号

根据 JR/T 0008—2000《银行卡发卡行标识代码及卡号》规定，银行卡的卡号长度及结构符合 ISO 7812-1 有关规定，由 13-19 位数字表示，具体由以下几部分组成：

9	XXXXX	X...X	X
	发卡行标识代码	自定义位	校验位

其中，

发卡行标识代码标识发卡机构，由 6 位数字表示，第 1 位固定为‘9’，后 5 位由 BIN 注册管理机构分配。

自定义位，由发卡行自定义，由 6-12 位数字组成。

校验位，卡号最后一位数字，根据校验位前的数字计算（Luhn 算法）得到。

目前，可整理出来的应用需求，大多来自标准法规，如 JR/T 0098.8—2012《中国金融移动支付 检测规范 第 8 部分：个人信息保护》。

JR/T 0098.8 要求，“客户端上不能存储敏感个人信息及其密文，敏感个人信息及其密文在使用后应立即清除。客户端存储一般个人信息时，应进行加密处理”、“服务器存储个人信息，应根据个人信息自动和非自动处理的特点，制定相应保护策略，防止个人信息的不当使用、毁损、泄露、删除等。服务器存储敏感个人信息时，应采用加密的方式存储。”

JR/T 0098.8 还规定，“敏感个人信息中的个人身份信息在下发至客户端之前，应屏蔽个人身份信息中不可猜测的一部分，被屏蔽部分使用统一的符号替代。”

此外，《人民银行关于银行业金融机构做好个人金融信息保护工作的通知》（银发[2011]17 号）规定，“银行业金融机构在收集、保存、使用、对外提供个人金融信息（包括个人身份信息、个人账户信息、个人金融交易信息等）时，应当严格遵守法律规定，采取有效措施加强对个人金融信息保护，确保信息安全，防止信息泄露和滥用”、“银行业金融机构要完善信息安全技术防范措施，确保个人金融信息在收集、传输、加工、保

存、使用等环节中不被泄露。”

2）关于身份证号

根据 GB 11643—1999《公民身份号码》规定，公民身份号码是特征组合码，由十七位数字本体码和一位数字校验码组成。

排列顺序从左至右依次为：六位数字地址码，八位数字出生日期码，三位数字顺序码和一位数字校验码 。

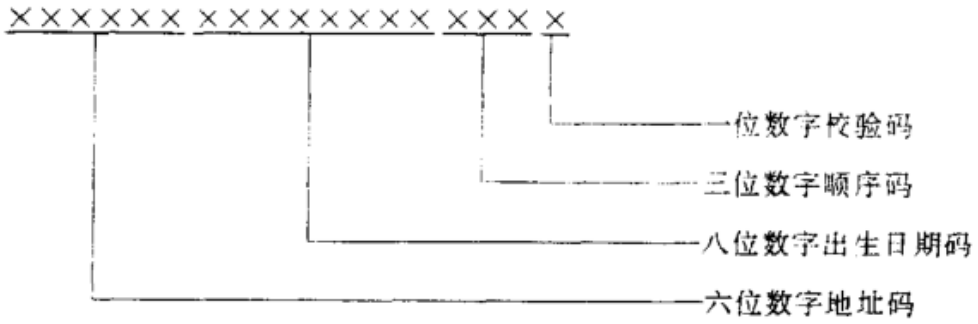


图 1-2 身份证号样式

其中，

地址码，表示编码对象常住户口所在县（市、旗、区）的行政区划代码，按 GB/T 2260 的规定执行；

出生日期码，表示编码对象出生的年、月、日，按 GB/T 7408 的规定执行；

顺序码，表示在同一地址码所标识的区域范围内，对同年、同月、同日出生的人编号的顺序号，顺序码的奇数分配给男性，偶数分配给女性；

校验码，采用 ISO 7064：1983，MOD 11-2 校验系统。

根据《中华人民共和国居民身份证法》（主席令第五十一号）要求，公民身份号码是每个公民唯一的、终身不变的身份代码，由公安机关按照公民身份号码国家标准编制。其中，第三章第十三条，“公民从事有关活动，……，有关单位及其工作人员对履行职责或者提供服务过程中获得的居民身份证记载的公民个人信息，应当予以保密。”

3）关于手机号码

我国使用的手机号码为 11 位，其中各段有不同的编码方向：前 3 位-网络识别号；第 4-7 位-地区编码；第 8-11 位-用户号码。号码也就是所谓的 MDN 号码，即本网移动用户作被叫时，主叫用户所需拨的号码，采取 E.164 编码方式，存储在 HLR 和 VLR 中，在 MAP 接口上传送。

MDN 号码的结构如下：

CC + MAC + $H_0H_1H_2H_3$ + ABCD，其中

CC：国家码，中国使用 86

MAC：移动接入码，

$H_0H_1H_2H_3$：HLR 识别码，由运营商统一分配（HLR 归属位置寄存器）

ABCD：移动用户号，由各 HLR 自行分配。

据不完全统计，截止 2018 年 7 月 1 日，手机号码段总量在 31 万-41 万之间，而且可能还会增加。手机号码段是区分手机运营商和手机区域的唯一标志，通过手机号码段可以进行手机号归属地查询。

手机号码的安全需求，可以参照《电信和互联网用户个人信息保护规定》。其中，第十条规定，“电信业务经营者、互联网信息服务提供者及其工作人员对在提供服务过

程中收集、使用的用户个人信息（包括用户姓名、出生日期、身份证件号码、住址、电话号码等）应当严格保密，不得泄露、篡改或者毁损，不得出售或者非法向他人提供。”

4）关于个人信息保护

除了上述个人敏感信息外，关于个人信息的保护，还有一些定性的标准法规。

如 GB/T 35273—2017《信息安全技术 个人信息安全规范》，明确要求，传输和存储个人敏感信息时，应采用加密等安全措施；存储个人生物识别信息时，应采用技术措施处理后再进行存储，例如仅存储个人生物识别信息的摘要。

如《网络安全法》第四十二条要求，网络运营者应当采取技术措施和其他必要措施，确保其收集的个人信息安全，防止信息泄露、毁损、丢失。

再比如《快递暂行条例》，它进一步细化了保护范畴，规定企业应当建立快件运单及电子数据管理制度，妥善保管用户信息等电子数据，定期销毁快件运单，企业及其从业人员不得出售或者非法泄露用户信息等。

其中，第三十四条规定，经营快递业务的企业应当建立快递运单及电子数据管理制度，妥善保管用户信息等电子数据，定期销毁快递运单，采取有效技术手段保证用户信息安全。

综上，这些编号类数据具有分段约束的格式特征，可以采用存在分段约束的编号类数据的保密格式加密算法。根据各分段的特征，在各分段内进行不同的保留格式加密计算，然后将加密结果合并，最后根据校验规则由本体码的加密结果计算得到校验码。

- 对于特征码分段，可采用 FFX，RtE 或 CtE 等算法进行保留格式加密，将特征码在特征码范围内映射成整数，对整数加密结果再映射回合法的特征码。
- 对于顺序码分段，可采用整数 FPE（Prefix 或 FFSEM）进行保留格式加密，将顺序码直接转换整数类型，对整数加密后，再转换回字符类型。

1.3.3 私有系统上云

私有系统加速上云，当网络数据流在虚拟机之间传输，用户对敏感信息和高级恶意软件的监视和控制能力会被削弱，而且，也有其自身的弱点，如云环境的设备和系统往往归属不同的厂商，在对接整合时有着不少困难。而且，虚拟化让资源边界变得模糊，动态扩展了计算、存储、网络资源，打破了传统的物理隔离，使得原有的管理环境复杂起来。无论是基础设施还是系统架构，都有可能随着业务需求的变化而加大不确定性。

私有系统上云后，存储方面，除了在数据防护时，加强机密性、完整性、可用性的能力，还要考虑到不同应用所采用的加密算法对防护强度的不一致。大多数应用系统都是基于数据库的应用系统，如金融、社保、电子政务、电子商务等，如果数据库中存储的大量用户敏感信息（如银行卡号、社保卡号、用户名和密码等）被窃取，将造成致命的破坏。引入 FPE 技术，将极大地提高数据库的安全性。无论新部署还是已有的数据库应用系统，都可以引入 FPE，增强系统安全性，实现在线解密预览，因其具有以下优势：

- 不改动现有软件系统的代码；
- 不改动现有数据库结构。

从数据库出发，在私有系统上云的应用场景中，FPE 的应用有以下几类：

1）数据遮蔽

数据遮蔽源于解决数据从生产环境向测试环境(或者开发环境)导入时可能产生的数据内容安全问题，它通过克隆原始数据进行掩码转换，输出一个与原数据格式、关联等一模一样的数据，用以进行功能测试、性能测试和模拟测试等。数据遮蔽内嵌数据修改规则，通过各种复杂算法，可以自动批量、快速地完成对敏感数据的修改，同时保证克

隆出来的数据库的数据量完全等同于生产库的数据量；敏感数据(如身份证号、电话号码、信用卡号码等)又作了伪装，看起来是真实数据，实际上是已经进行了修改的假数据，从而消除了敏感数据的泄露隐患。如果充分且合理利用的话，数据遮蔽必成为保障数据安全的重要技术。

2）格式兼容加密

FPE 也适合于遵循既有协议格式、格式兼容的数据加密的应用领域，这将是保留格式加密在未来的一个重要的应用领域。有关于 FPE 在多媒体加密领域的应用，将 FPE 方法应用到 JPEG2000 的加密中，对 JPEG2000 压缩后的码流进行保留格式加密。JPEG2000 压缩流中不包括超过 0xff8f 的序列，且不以 0xff 结尾。加密后的图像必须保证能被标准解码器解码，因此加密后的码流不能产生多余的标记字段，也不能包括超过 0xff8f 的序列，并且不能以 0xff 结尾。研究者以有限自动机描述了 JPEG2000 压缩流的包体部分，将 JPEG2000 的加密问题转化为正则语言上的 FPE 问题。现有的正则语言的 FPE 解决方案都可应用到 JPEG2000 压缩流包体数据的加密中。

3）数据加密

首先，对数据库中各种类型数据实现保留格式加密，并将数据转化为相同数据类型的密文存储在数据库中，即：

c＝PreEncrypt(k，m]

m＝PreDncrypt(k，c)

其中，PreEncrypt( )表示保留格式加密函数，PreDecrypt( )表示保留格式解密函数，k 为密钥，m 为明文，c 为密文，且 c 与 m 为相同数据类型。

利用 FPE 加密，对原始数据库中明文进行加密，包括所有表名、字段名称以及每一数据项，再利用产生的密文数据来生成一个新的密文数据库。

其次，实现对用户使用的透明性，即保证拥有加密密钥的用户以与明文数据相同的操作来访问数据库，自动实现对数据的加解密，避免用户参与。在实际应用中，用户对数据库的访问包括，增删改查等。

此外，传统模式下，基础的信息安全保障措施不全，甚至连机房等基础环境都不符合安全标准，如政务云。现阶段政务云建设，主要是借助 IaaS/PaaS，以数据为核心，实现基础设施资源整合与共享，在 2.0 阶段，采用 FPE 算法，将对业务连续性和数据可靠性提供更好的保障。

2. 现有 FPE 方案

2002 年,Black 和 Rogaway 提出了 3 种 FPE 构建方法：Prefix、Cycle-Walking 和 Generalized-Feistel。这 3 种方法不仅在一定程度上解决了整数集上的 FPE 问题，而且成为构造 FPE 模型的基本方法，其中，Cycle-Walking 和 Generalized-Feistel 在加密模型中得到了广泛使用。

2.1 Prefix 方案

2.1.1 基本思想

Prefix 方案首先将消息空间映射到整数空间$X = \{0,1,\cdots,n-1\}$，其中n为消息空间的大小；然后使用基础分组密码算法E及密钥K对整数集X中的每个整数进行加密得

X	0	1	⋯	n − 1
$E_K(X)$	$E_K(0)$	$E_K(1)$	⋯	$E_K(n-1)$

然后对$E_K(X)$按数值大小排序，并将其依次映射到集合$X=\{0,1,\cdots,n-1\}$，记$O_K(i)$表示$E_K(i)$的索引值，即$E_K(i)$为集合$E_K(X)$中第i大的值。从而，使用基础分组密码算法E及密钥K建立了集合X上的置换表

X	0	1	⋯	n − 1
$O_K(X)$	$O_K(0)$	$O_K(1)$	⋯	$O_K(n-1)$

使用该置换表可完成大小为n的消息空间上的加解密运算，并使运算结果仍在该消息空间内。

2.1.2 研究进展

2002 年，Black 等提出该方案，因其安全强度高，后续并无对该方案的安全性分析。针对该方案每次更换密钥时，需重新生成完整的置换表的问题，研究者提出可通过引入可调因子以及模运算的方式，使得更换密钥时不必重新生成置换表。

2.1.3 优缺点分析

优点：不会降低所选用的基础分组密码的安全强度。当消息空间较小时，如消息空间大小小于10^6时，该方案可使用基础分组密码算法迅速完成置换表的计算，基于置换表可以极高的速度完成加解密，且该方案不会降低所使用的基础分组密码算法的安全强度。

缺点：当消息空间规模较大时，生成置换表的效率低下，存储置换表也需极大的空间，该方案不再具有实用性。

2.2 Cycle-Walking 方案

2.2.1 基本思想

Cycle-Walking 方案的原理是利用基础分组密码算法对加密所得结果反复加密，直至其落入指定的消息空间：将消息空间映射到整数空间$X=\{0,1,\cdots,n-1\}$，其中n表示消息空间的大小。当使用基础分组密码算法E及密钥K对消息$i\in X$加密时，若$y=E_K(i)$不在空间X中，则再次调用E_K对y加密，直至加密结果$y=E_K(E_K(\cdots E_K(i)))$落入空间X为止。因$E_K$可逆，因此解密时，只需对消息y反复解密，解密结果首次出现在空间X中的值即为正确的明文。

2.2.2 研究进展

2002 年，Black 等提出该方案，因其安全强度高，后续并无对该方案的安全性分析，但作为基础部件应用到一些新型的保持格式加密中了。

2.2.3 优缺点分析

优点：不会降低所选用的基础分组密码的安全强度，可适用于任意空间大小上的保持格式加密。

缺点：受所选基础分组密码分组长度的影响，当消息空间大小与分组长度所表示的空间大小相差很大时，对消息加密时，可能出现多次调用基础分组密码的情况，从而使该方案的效率因调用基础算法的次数的增多而降低。假定基础分组密码的分组长度为L，

即E_K每次对长度为L的消息加密，得到长度为L的密文信息。而对空间X中的消息均可使用长度为$\lceil \frac{n}{2} \rceil$的二进制串表示，因此调用 1 次$E_K$对X中消息加密所得密文落入空间X中的概率不大于$\frac{1}{2^{L-\lceil \frac{n}{2} \rceil}}$，即平均需调用$E_K$的次数为$2^{L-\lceil \frac{n}{2} \rceil}$。因此，Cycle-Walking 方案适用于消息空间大小接近于所采用的基础分组密码算法的分组长度所表示的空间大小，例，采用分组长度64比特的基础算法，适用于对大小接近于2^{64}的消息空间使用Cycle-Walking方案完成保持格式加密。

2.3 基于 Generalized-Feistel 结构的方案

2.3.1 基本思想

Generalized-Feistel 方案的核心思想有两点：一是使用 Feistel 结构调用基础分组密码算法构造适用于消息空间大小的分组密码算法，其次是使用 Cycle-Walking 方法或模运算将加密所得消息映射到指定的消息空间。

首先，我们可以调用基础分组密码算法E_K，使用 Feistel 结构来构造任意分组长度的分组密码算法。设分组长度为blocksize，加密一个消息分组P的过程如下：

1）输入：$P = L_0 \| R_0$

2）For $i = 1$ to t

If i is odd

$L_i = f(L_{i-1}, E_{K,0}(R_{i-1}))$

$R_i = R_{i-1}$

Else

$L_i = L_{i-1}$

$R_i = f'(E_{K,1}(L_{i-1}), R_{i-1})$

3）输出：$C = L_t \| R_t$

其中$X = L_0 \| R_0$表示将消息X分成左右两部分，左右两部分的长度可以不一致。对于$z = f(x, y)$，当给定(z,y)时可求得x，同样对于$z = f'(x, y)$，当给定(z,y)时可求得x。因为在每一轮中，前一轮的部分信息传递至下一轮，且可通过这部分信息及后一轮中变化后的部分信息恢复上一轮的信息，因此每一轮都可逆，从而经过该结构加密后的结果可以正常解密。基于基础分组密码E_K采用 Feistel 结构构造的任意长度的分组密码算法的加解密过程如 0。

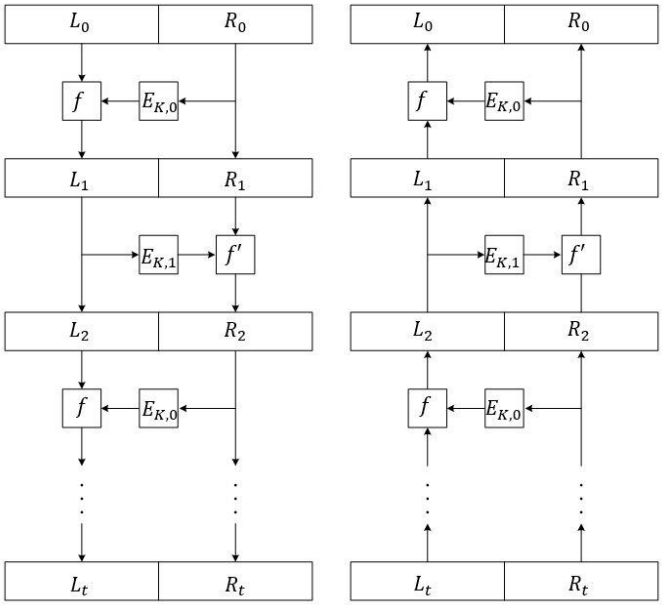


图 2-1 基于 Feistel 结构的任意分组长度的分组密码加密过程及解密过程

然后，采用上述构造的加解密算法对特定消息空间上的消息进行加解密，使加解密后的消息仍在该消息空间中。记消息空间大小为n，并使用整数空间$\mathrm{X}=\{0,1,\cdots,\mathrm{n}-1\}$表示该消息空间。对消息空间中长度为m的字符串加密时，消息空间大小为n^{m}，将所有长度为m的消息映射到整数集$\mathbf{X}=\{0,1,\cdots,\mathrm{n}^{\mathrm{m}}-1\}$。为使加密结果也为消息空间中长度为m密文，即加密后的消息在整数集$\mathbf{X}$中，可以采用 Cycle-Walking 的方式对消息反复加密，直至密文落入该消息空间[1]，过程如下：

1）计算分组长度$\mathrm{blocksize}=\lceil\log \mathrm{n}^{\mathrm{m}}\rceil$，即我们将消息空间中所有长度为m的字符串均转化为长度为blocksize的二进制串。其中⌈x⌉表示不小于x的最小的整数。

2）基于基础分组密码$\mathrm{E_K}$，采用 Feistel 结构构造分组长度为blocksize的分组密码$\mathrm{GFE_K}$。

3）使用$\mathrm{GFE_K}$对消息P加密，记加密结果为C，即$\mathrm{C}=\mathrm{GFE_K(P)}$，若C不在消息空间X中，则反复调用$\mathrm{GFE_K}$，即使$\mathrm{C}=\mathrm{GFE_K(GFE_K(\cdots GFE_K(P)))}$落入消息空间X中。

易知使用$\mathrm{GFE_K}$对$\mathbf{X}$中的消息加密时，每次加密所得结果落入消息空间$\mathbf{X}$中的概率为$\frac{\mathrm{n}^{\mathrm{m}}}{2^{blocksize}}=\frac{1}{2^{blocksize-\log n^{m}}}>\frac{1}{2}$，即平均调用$\mathrm{GFE_K}$两次就可以使加密结果落入消息空间$\mathbf{X}$，即按上述方式构造的保持格式加密方案具有较好的加密性能。安全性方面，Black 和 Rogaway 证明了，当攻击者拥有明密文对少于$2^{blocksize/4}$时，该方案是足够安全的。

除了使用上述 Cycle-Walking 方式对加密结果反复调用$\mathrm{GFE_K}$使最终结果落入消息空间中外，还可以使用模运算的方式，使采用$\mathrm{GFE_K}$加密时，所有中间运算结果均在特定的消息空间中。对消息空间中长度为m的字符串P加密时，将消息分成两块$\mathrm{P}=\mathrm{L_0}\parallel R_0$，其中$\mathrm{L_0}$、$\mathrm{R_0}$的长度分别为u、v，$\mathrm{u}+\mathrm{v}=\mathrm{m}$，即$\mathrm{L_0}$在消息空间$\mathbf{X}'=\{0,1,\cdots,\mathrm{n}^{u}-1\}$中，$\mathrm{R_0}$在消息空间$\mathbf{X}''=\{0,1,\cdots,\mathrm{n}^{v}-1\}$中。我们采用模运算使$\mathrm{GFE_K}$对消息P加密过程时，使所有中间运算值$\mathrm{L_i}$,$R_i$始终分别落入消息空间$\mathbf{X}'$,$\mathbf{X}''$中，过程如下：

1）输入：$\mathrm{P}=\mathrm{L_0}\parallel\mathrm{R_0}$

2）For $\mathrm{i}=1$ to t

If i is odd

$L_i=L_{i-1}+E_{K,0}(R_{i-1})\ mod\ n^u$

$R_i=R_{i-1}$

Else

$L_i = L_{i-1}$

$R_i = E_{K,1}(L_{i-1}) + R_{i-1}\ mod\ n^v$

3）输出：$C = L_t \| R_t$

其中$E_{K,0}(R_{i-1})$，$E_{K,1}(L_{i-1})$表示使用E_K对输入消息R_{i-1}、L_{i-1}进行变换，并将输出结果转化为n进制数。易知上述加密过程中，L_i, R_i始终在消息空间X', X''中，从而最终加密结果C也为消息空间中长度m的字符串。且对$L_i = L_{i-1} + E_{K,0}(R_{i-1})\ mod\ n^u$，当给定$L_i, E_{K,0}(R_{i-1})$时，可求得$L_{i-1} = L_i + E_{K,0}(R_{i-1})\ mod\ n^u$，从而整个加密过程可逆，即解密可得正确的明文。

在 NIST SP800-38G 中，研究者对上述采用模运算保持格式加密的方案设计了具体的调用基础算法E_K的方式，分别称为FF1,FF3方案，以提高安全性。FF1方案与FF3方案均将消息分成两块几乎等长的消息来处理，即分为长度为$\lfloor \frac{m}{2} \rfloor, \lceil \frac{m}{2} \rceil$的两部分来处理，不同之处主要在于对基础分组密码算法E_K（在两个方案中，E_K的分组长度均为 128 比特）的调用方式不同。

FF1 方案采用 10 轮上述 Feistel 结构，调用E_K的过程如下：

1）加解密双方预先共享可调因子T，并根据双方共有的参数信息生成以下 128 比特向量P；

2）记第i轮到第$i+1$中保持不变的部分消息为X，根据X构造向量Q；

3）根据向量P与Q，调用基础算法E_K，生成指定大小的$y = E_K(P, Q)$，然后使用y参与计算，完成加解密。

因加解密过程中，在对应的轮数，均可根据当前已知的信息计算出y，从而可以正确的加解密。

FF3 方案采用 8 轮上述采用模运算的 Feistel 结构，调用E_K的过程如下：

1）加解密双方预先共享消息空间上长度为64的字符串T，并记T_L、T_R分别为T的左半部分和右半部分；

2）记第i轮到第$i+1$中保持不变的部分消息为X，并结合共享消息W（取值为T_L或T_R，根据轮数的奇偶性逐轮循环选取），调用基础分组密码算法生成数值y；

3）使用y参与计算，完成加解密。

与 FF1 方案及此类方案类似，因加解密过程中，在对应的轮数，均可根据当前已知的信息计算出y，从而可以正确的加解密。相比于 FF1 方案，FF3 方案中y的计算过程更复杂，且中间对消息采用反序变换，使得模运算中，消息的高低位影响更均衡，使得对方案的每一轮的分析难度更大。

2.3.2 研究进展

2002 年，Black 等提出 Prefix、Cycle-Walking 方案的同时提出了采用 General-Feistel 并结合 Cycle-Walking 的方案。自 2002 年 Generalized-Feistel 方法首次使用 Feistel 网络来构造 FPE 算法以来,其后的 FPE 方案大多都采用了这种结构,这与 Feistel 网络相对完备的研究成果和 FPE 算法本身对保留格式的要求相关.Feistel 网络是目前主流的分组密码设计模式之一,基于 Feistel 网络,可以通过定义分组大小、密钥长度、轮次数、子密钥生成、轮函数等来构造一个分组密码.在 FPE 模型的构造过程中,Feistel 网络被广泛应用于构造合适分组长度的分组密码,

2016 年，NIST Special Publication 800-38G 公布，从候选算法中选出 FFX、BPS 方案，并分别称之为 FF1、FF3 方案，其中，在 NIST SP 800-38G 原始文稿中记为 FF2 的方案 VAES3，因 NSA 指出该方案的安全性达不到所期望的 128 比特而被删除。FF1 和

FF3，都是基于 Festiel 的加密模式。

2.3.3 优缺点分析

优点：基于 General-Feistel 结构的保持格式加密方案可适用于对任意空间大小的消息域里面的消息保持格式加密，且加解密效率高，不需要预先生成及存储置换表等，可满足各种实际应用需求。

缺点：当消息空间域规模较小时，因 General-Feistel 结构的原因，基于该结构的方案，其针对密钥恢复攻击的实际安全性达不到预期。

2.4 方案的比较

Prefix,Cycle-Walking 和Generalized-Feistel 方法能够解决特定范围整数集上的FPE 问题。

1）Prefix 方法揭示了保留格式加密的本质,即消息空间内的置换.Prefix 方法采用建立预置换表的做法,为复杂问题域上的 FPE 问题提供了一种思路,即通过某种方式(比如排序)来建立消息空间内的置换,RtE 方法排序后加密的思想与此异曲同工;

2）Cycle-Walking 方法是一种解决任意有限问题域的 FPE 问题的通用办法,目前所提出的 FPE 模型在解决任意有限域上的 FPE 问题时,通常基于该方法来确保数据被加密到正确的范围内;

3）Generalized-Feistel 方法允许用户通过定义 Feistel 网络的相关参数来构建合适分组长度的对称密码,成为保留格式加密模型普遍适用的构造方法,目前所提出的 FFSEM,FFX,BPS 等模型都基于其完成模型的构造。

3. 现有安全性研究

作为一种规约的方法，可证明安全首先要确立密码体制的安全目标，然后定义一个合理的安全模型，同时指出密码方案的安全性和安全模型之间的规约关系。在对密码学方案设计的安全性进行研究的时候，主要考虑在现有的攻击模型下能达到的安全目标。因此，通常把安全目标和攻击模型相结合来定义密码学方案的安全性。

在公钥密码体制下，安全目标主要有：语义安全(semantic security)、不可区分性(indistinguishability，简称 IND)、非延展性(non-malleability，简称 NM)和明文可意识性(plaintext awareness，简称 PA)。目前，语义安全性和非延展安全性是公钥密码学中被广泛接受的两种安全性定义。1984 年 Goldwasser 和 Micali 首次提出了语义安全性的定义，指的是攻击者没有能力从给定的密文中得到任何关于明文的信息，并证明了语义安全和不可区分性等价。非延展安全性是由 Dolev、Dwork 和 Naor 在 1991 年提出的，指攻击者不能从给定的密文中，建立和明文意义相关的明文的密文。明文可意识性是指在不知道相应明文的条件下无法建立合法密文，由于明文可意识性是在随机预言机模型(random oracle model)下定义的，然而随机预言机模型是一种理想化的模型，随机预言机模型证明的有效性是有争议的，因此对实际系统的安全性讨论时很少提到明文可意识性。

公钥密码体制的安全目标在对称密码体制中同样适用。另外密码学方案一般建立在底层基础模块上，因此方案的安全性可规约到底层基础模块的安全性。保留格式加密(format preserving encryption，简称 FPE)是一种新型的对称加密技术，因此对称密

码体制的安全目标也适用于保留格式加密。2009年Bellare对FPE问题进行了深入研究，定义了针对FPE的多种安全目标。

3.1 安全目标

保留格式加密是一种特殊的对称密码，基础模块是分组密码和伪随机函数，因此它的一个重要的安全目标是伪随机性。2002年，Black和Rogaway首次描述FPE的安全性，认为标准FPE安全就是伪随机置换(PseudoRandom Permutation，简称PRP)安全。也就是说FPE的本质是在特定消息空间内的伪随机置换。较低的安全目标包括SPI(Single Point Indistinguishability)安全，MP(Message Privacy)安全和MR(Message Recovery)安全，这些安全目标虽然安全性相比而言有所削弱，但更易实现，并且也可以满足典型应用的安全性要求。

3.1.1 PRP 安全

根据基本FPE的定义，任意选定密钥空间的一个密钥 $\mathrm{k} \in \mathrm{K}$，$E_k(\cdot)$是消息空间X上由对称密钥k决定的一个置换。令$\mathrm{Perm(X)}$表示消息空间X上所有置换的集合，$\mathrm{P} \leftarrow \mathrm{Perm(X)}$表示从$\mathrm{Perm(X)}$中随机抽取一个置换P。设$A^f$是一个可以查询预言机f的攻击者，f要么是加密预言机$E_k(\cdot)$，要么是一个随机置换预言机$\mathrm{P}(\cdot)$。假定攻击者不能执行消息空间以外的查询，也不能重复相同的查询，这样的攻击者A可以认为是FPE方案ε_{FPE}的PRP攻击者，并且定义其在攻击中可获得的优势为

$$Adv_{\varepsilon_{FPE}}^{PRP}(\mathrm{A}) = |\Pr\left(k \leftarrow K: A^{E_k(\cdot)} = 1\right) - \Pr\left(P \leftarrow Perm(\cdot): A^{P(\cdot)} = 1\right)|.$$

该式度量了攻击者A区分FPE方案ε_{FPE}和随机置换P的概率优势。

接下来我们给出PRP安全的定义：令$Adv_{\varepsilon_{FPE}}^{PRP}(\mathrm{t},\mathrm{q}) \stackrel{\text{def}}{=} max_A Adv_{\varepsilon_{FPE}}^{PRP}(\mathrm{A})$，其中t为攻击者执行破解算法的时间，q为攻击者查询的次数。如果$Adv_{\varepsilon_{FPE}}^{PRP}(\cdot,\cdot)$是一个可忽略的量，则称该FPE方案$\varepsilon_{FPE}$为伪随机置换，也就是达到了PRP安全。

PRP安全的标准目标是攻击者无法区分该FPE方案和消息空间内的某个随机置换。Bellare等人基于游戏模型，进一步定义了适用于FPE一般定义的PRP安全目标。

该定义基于游戏$PRP_{\varepsilon_{FPE}}$，通过攻击者与挑战者之间的博弈，对攻击者区分保留格式加密和随机置换的能力进行量化。首先随机选取密钥k、待猜测的布尔值b和随机置换P。$PRP_{\varepsilon_{FPE}}$拥有加密预言机Oracle(Enc)，用于响应攻击者A的加密查询。如果$\mathrm{b} = 1$，加密预言机Oracle(Enc)以FPE方案ε_{FPE}响应；如果$\mathrm{b} = 0$，加密预言机Oracle(Enc)以随机置换P响应。攻击者A的目标是：在查询加密预言机Oracle(Enc)一定次数后，给出一个判定，Oracle(Enc)使用的是ε_{FPE}还是P，即输出一个判定值b'。如果$b' = b$，说明攻击者A判定成功，相反则失败。A在该游戏中具有的优势为

$$Adv_{\varepsilon_{FPE}}^{PRP}(\mathrm{A}) = 2\Pr\left(PRP_{\varepsilon_{FPE}}^{A} \Longrightarrow true\right) - 1.$$

当攻击者在该游戏中具有的优势不明显时，我们说该FPE方案ε_{FPE}达到了PRP安全。

3.1.2 SPI 安全

SPI安全即单点不可区分安全，要求攻击者在能够访问真正的加密预言机的前提下也不能区分是对给定的消息还是对某个随机点进行的加密。

SPI安全的定义基于游戏$SPI_{\varepsilon_{FPE}}$。类似于$PRP_{\varepsilon_{FPE}}$，$SPI_{\varepsilon_{FPE}}$首先随机选取密钥k和待猜测的布尔值b。攻击者A首先对某个待猜测的明文单点执行一次Oracle(Test)查询，该预言机根据b值返回该单点在FPE方案ε_{FPE}下的密文或消息空间中的某一个随机点。

攻击者A的目标是：在对真正的加密预言机Oracle(Enc)(该加密预言机只以ε_{FPE}响应)进行一定次数适应性的查询后(在此查询过程中，禁止重复查询，禁止对待猜测单点

进行查询)，给出一个判定：首次执行的Oracle(Test)查询得到的结果是选定单点所对应的密文还是消息空间中的某个随机点。即输出一个判定值b'。如果$b'=b$，说明攻击者A判定成功，相反则失败。A在该游戏中具有的优势为

$$Adv_{\varepsilon_{FPE}}^{SPI}(\mathrm{A})=2\Pr\left(SPI_{\varepsilon_{FPE}}^{A}\Rightarrow true\right)-1.$$

当攻击者在该游戏中具有的优势不明显时，我们说该FPE方案ε_{FPE}达到了SPI安全。

3.1.3 MR 安全

MR安全要求在对抗消息恢复攻击时，即使提供给攻击者加密预言机和明文的概率分布、格式、调整因子等信息，攻击者也不能从密文中恢复出明文。若加密是完全随机的，就意味着在已知目标密文y^*和能够访问加密预言机Oracle(Enc)的前提下也无法恢复明文。但是这对确定性加密的要求太高，因为攻击者可以通过Oracle(Enc)加密消息$x_1,\ldots,x_q$，得到其对应密$y_1,\ldots,y_q$。如果存在某个$y_i=y^*$，那么攻击者就可以确定待猜测的目标明文$x^*=x_i$。当此类攻击是针对该密码方案最好的攻击方式时，我们称该密码方案是MR安全的。即对于FPE方案ε_{FPE}，如果不存在成功概率明显大于上述攻击的攻击方式，则认为 ε_{FPE}达到了MR安全。

MR安全的定义基于游戏$MR_{\varepsilon_{FPE}}$。该游戏提供了3个预言机：Oracle(Enc)，Oracle(Eq)和Oracle(Test)，以及攻击者A和模拟器S。其中，Oracle(Enc)用于以FPE加密方案ε_{FPE}响应攻击者的加密查询，Oracle(Eq)允许攻击者查询一个消息空间内的元素是否是目标密文所对应的原始明文x^*，Oracle(Test)用于赋予攻击者获取目标密文y^*的能力。

在游戏开始前，先随机选取一个目标明文x^*，然后执行加密算法得到其对应密文y^*，最后将加密所用到的格式信息和调整因子发布给攻击者。攻击者A执行1次Oracle(Test)查询和q次Oracle(Enc)查询来获得有利的信息以便做出判定；模拟器S使用q次Oracle(Eq)查询。攻击者以及模拟器的目标都是通过多次查询各自预言机后，输出一个对目标明文的猜测值x，如果$x=x^*$，说明攻击者判定成功，否则失败。事实上，MR安全目标度量了攻击者在能够获取目标密文、并能q次访问加密预言机的情况下，与最一般的攻击方式—从消息空间中选取q个候选值依次与目标明文判定相比，成功的概率差。即攻击者A在该游戏中具有的优势为

$$Adv_{\varepsilon_{FPE}}^{MR}(\mathrm{A})=\Pr\left(MR_{\varepsilon_{FPE}}^{A}\Rightarrow true\right)-p_A,$$

其中$p_A=max_S\Pr\left(MR_{\varepsilon_{FPE}}^{S}\Rightarrow true\right)$，最大值遍历所有满足条件的模拟器S。

3.1.4 MP 安全

MP安全是为了量化拥有加密预言机的攻击者从目标密文中计算其对应明文被某函数作用后得到的结果的能力，是相比于MR安全更一般的情形。

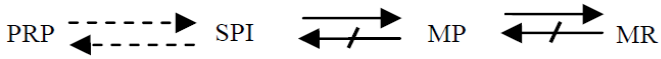
MP安全的定义基于游戏$MP_{\varepsilon_{FPE}}$。类似于$MR_{\varepsilon_{FPE}}$，$MP_{\varepsilon_{FPE}}$也提供了3个相同的预言机：Oracle(Enc)，Oracle(Eq)和Oracle(Test)，以及攻击者A和模拟器S。其中A可以执行1次Oracle(Test)查询和q次Oracle(Enc)查询，S可以执行q次Oracle(Eq)查询。不同之处在于，在$MP_{\varepsilon_{FPE}}$中，攻击者需要猜测的不是明文信息，而是该明文在某个函数f作用后的输出值，并将该猜测输出值与实际结果进行比较，判断攻击是否成功。

事实上，MR只是MP的一种特殊形式，也就是MP中f为恒等函数的情形。因此，攻击者A在游戏$MP_{\varepsilon_{FPE}}$中具有的优势的计算是一样的：

$$Adv_{\varepsilon_{FPE}}^{MP}(\mathrm{A})=\Pr\left(MP_{\varepsilon_{FPE}}^{A}\Rightarrow true\right)-p_A,$$

其中$p_A=max_S\Pr\left(MP_{\varepsilon_{FPE}}^{S}\Rightarrow true\right)$，最大值遍历所有满足条件的模拟器S。

3.1.5 几种安全目标之间的关系



我们所提到的4个安全目标之间的关系如上图所示，其中实箭头表示起始端能推导出指向端，虚箭头表示在推导过程中某些参数有损。PRP是保留格式加密的标准安全目标，SPI是PRP的一个变种，与PRP具有类似的安全性。PRP安全就意味着SPI安全，SPI安全同样也意味着PRP安全。SPI虽然安全性有损，但仍可达到比PRP更好的安全边界。为了适应确定性加密的概念，MP给保留格式加密带来了语义安全，它要求密文不能泄露真实信息，安全性要比PRP和SPI略低。最基本也是最常用的安全性要求就是MR安全，MR从整体上要求攻击者不能从消息的密文中恢复明文。Bellare等人证明了SPI可以推导出MP，但是MP不能推导出SPI。他们也证明了MP可以推导出MR，相反则不成立。事实上，MR安全只是MP安全的一种特殊形式，也就是MP中攻击者采用函数为恒等函数的情形。

3.2 攻击模型

攻击者根据可利用的条件可以采用不同的攻击模型。常见的攻击模型包括唯密文攻击(ciphertext only attack)、已知明文攻击(known plaintext attack)、选择明文攻击(chosen plaintext attack)和选择密文攻击(chosen ciphertext attack)等。进一步地，根据询问预言机方式的不同，可分为非适应性攻击模型和适应性攻击模型。适应性攻击模型允许攻击者随时根据每次询问的结果来调整之后询问的内容，而在非适应性攻击模型中，攻击者在开始询问之前就需要确定询问的全部内容。显然，适应性攻击强度比非适应性要高。

在设计加密方案时主要关注的攻击模型有：非适应性选择明文攻击(non-adaptive chosen plaintext attack，简称CPA1)、适应性选择明文攻击(adaptive chosen plaintext attack，简称CPA2)、非适应性选择密文攻击(non-adaptive chosen ciphertext attack，简称CCA1)、适应性选择密文攻击(adaptive chosen ciphertext attack，简称CCA2)。

一般使用安全目标与攻击模型相结合的方式来定义加密方案的安全性，把保留格式加密的4种安全目标和上述的攻击模型相结合，就得到了保留格式加密主要的安全性，比如PRP-CPA1，PRP-CPA2，MR-CCA1，MR-CCA2等。很显然，PRP-CCA2具有最高的安全性，而MR-CPA1是保留格式加密最基本的安全性。

如果攻击者只允许对加密预言机进行查询，而不允许查询解密预言机，就定义了选择明文安全的攻击模型。在这种情况下，如果要求攻击者在第1次查询加密预言机之前就必须准备好所有要查询的问题，则定义了非适应性选择明文攻击的游戏模型，即PRP-CPA1；相反地，如果允许攻击者通过分析前面的查询结果再给出下一次要查询的问题，那么就定义了适应性选择明文攻击的游戏模型，即PRP-CPA2。

上述描述是基于CPA攻击模型的，如果要达到CCA安全标准，上述的游戏模型中可以通过加入一个解密预言机来完成，比如在$PRP_{\varepsilon_{FPE}}$游戏中引入解密预言机Oracle(Dec)，用于响应攻击者的解密查询。如果$b=1$，解密预言机Oracle(Dec)以保留格式加密方案对输入的密文进行解密并响应；如果$b=0$，解密预言机Oracle(Dec)以随机选取的方式置换P对输入的密文进行解密并响应。在这种情况下，如果攻击者既允许对加密预言机进行查询，又允许对解密预言机进行查询，同时要求攻击者在第1次查询加密预言机之前就必须准备好所有要查询的问题，于是定义了非适应性选择密文攻击的游戏模型，即PRP-CCA1；相反地，如果允许攻击者通过分析前面的查询结果再给出下一次要查询的问题，那么就定义了适应性选择密文攻击的游戏模型，即PRP-CCA2。

4. 主要 FPE 加密模型

第 2 章节中介绍的 3 种 FPE 方案不仅在一定程度上解决了整数集上的 FPE 问题,而且成为构造 FPE 模型的基本方法，本章节介绍的 FFSEM、RtE 和 FFX 是三个最具代表性的 FPE 模型，对这些模型的构造特点进行研究，并分析它们的优缺点，对于研究 FPE 加密模型具有非常重要的意义。下面对这三个典型 FPE 模型进行详细介绍和分析[26]。

4.1 FFSEM 模型

FFSEM[11]是基于 Feistel 网络的适用于整数域上的 FPE 方案,FFSEM 主要由两部分组成：

1）平衡 Feistel 网络，构造合适分组长度的对称密码；

2）Cycle-walking，用来实现使用 2m 位的分组密码能够对大小为 n(n<2m)的整数集合进行加(解)密。

FFSEM 模型的算法可以描述为：

算法 setup：

FFSEM 初始化阶段需要确定：

- FFSEM-PRF，即平衡 Feistel 网络中使用的伪随机函数，FFSEM 利用截断分组密码 AES 输出的方式构造 FFSEM-PRF；
- 对称密钥 k、消息空间的大小 n 和轮次数 r 等。

算法 encryption：

输入为明文 x，输出为满足格式要求的密文 y。

加密过程可以描述为：首先，将明文 x 编码为 l=[$\log_2$n]+1 位的二进制数（[x]表示不大于 x 的最大整数)；然后，执行 Cycle-walking 过程，每次 Cycle-walking 都执行 r 轮 Feistel 轮运算，直到产生合适的二进制密文；最后，将二进制密文解码为消息空间内的整数 y，y 即为最终密文。

算法 decryption：

解密算法为算法 encryption 的逆过程。

加(解)密过程如图 4.1 所示。

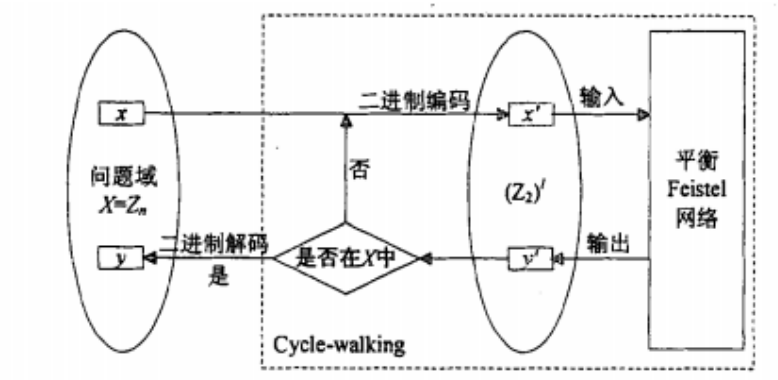


图 4.1 FFSEM 模型

目前，FFSEM 模型已被美国国家标准与技术研究院(NIST)采纳，并且被 Voltage 公司广泛应用。然而 FFSEM 仅实现了整数域上的 FPE 问题，并不能作为通用模型更广泛的使用。而且 Cycle-walking 会多次循环执行分组加密算法，因此具有不确定的性能。

4.2 RtE 模型

Bellare 等人提出的 RtE 是一种相对比较通用的模型，RtE 的基本思想是先排序后加密,将复杂问题域上的 FPE 问题转化为建立元素与索引的对应关系以及整数 FPE 问题。

为了实现对元素的排序，需要有效的描述消息空间，在编码理论研究领域，自 1977 年文献发表以来，已经开始了对特定消息空间上高效 rank 算法的研究。其中，Bellare 等人针对可用正则语言描述的消息空间，介绍了高效的 rank 和 unrank 算法，并将消息空间划分为两大类：可使用正则语言描述的消息空间和不能使用正则语言描述的消息空间。RtE 方法针对正则语言描述的消息空间，使用有限自动机 DFA 来描述待加密的有限域，并介绍了该情况下的 rank 和 unrank 算法，该算法是高效易用的。

RtE 方法的算法可以描述为：

算法 setup：

RtE 方法的初始化阶段确定：

- rank 和 unrank 函数。确定消息空间 X 上的排序函数 rank，建立 X 与整数域的映射 rank：$x \rightarrow Z_{|x|} \cup \{\perp\}$通过 rank 算法能够将元素 $x \in X$ 映射为其在 x 中的索引 $i \in Z_{|x|}$，如果 $x \notin X$，则 $rank(x)=\perp$。相应的，反排序函数 unrank：$N \rightarrow X \cup \{\perp\}$能够将一个整数（索引值 $i \in Z_{|x|}$）映射为 X 中的元素 $x \in X$，如果 $i \notin Z_{|x|}$则 $unrank(i)=\perp$
- 整数 FPE 方案 F_{Z_n}及其参数，Bellare 提出了基于非平衡 Feistel 网络的两个整数 FPE 方案。当然，RtE 方法并不限制使用其他安全的整数 FPE 方案，如 FFSEM 等。

算法 encryption：

输入为明文 x，整数 FPE 算法 F_{Z_n}，对称密钥 k 和调整因子 t 等，输出密文 y。

x 和 y 都属于消息空间 X。

加密过程可以描述为：首先执行 rank 算法，将明文 $x \in X$ 映射为消息空间 X 中的索引 $i \in Z_{|x|}$；然后执行整数 FPE 算法 F_{Z_n}。将 i 加密为 $j \in Z_{|x|}$，最后执 unrank 算法，将 j 映射回消息空间 X 中的元素 $y \in X$，从而得到密文 y。

算法 decryption：

解密算法为算法 encryption 的逆过程。

RtE 方法的加(解)密过程如 0 所示。

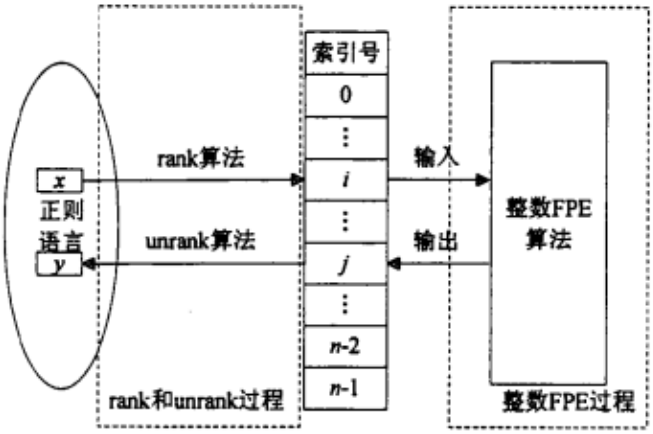


图 4.2　RtE 方法

RtE 方法的安全性归约于整数 FPE 算法 F_{Z_n} 的安全性，只要采用安全的 F_{Z_n}，RtE 就是安全的。

RtE 方法的效率主要决定于消息空间上的排序算法，其时间复杂度约为 rank 和 unrank 算法时间的总和。Bellare 提出的排序算法，是适用于正则语言的消息空间的高效算法。MäKinen 提出的排序算法可对上下文无关的语言进行排序。另外，可以使用 Lucas-Lehmer 编码提供的高效 rank 算法，在问题域 X_{ni}(X_{ni} 是由 n 个元素的全部置换集合)上加密。其他的例子还有伸展树、B 树、Dyck 语言等。然而，并非任意的消息空间都能构造高效的 rank 和 unrank 算法，Bellare 等人提出，在解决实际问题时，能够实现不需要 rank 算法的高效 FPE 方案作为学者们继续探索研究的开放性问题。

4.3 FFX 模型

Bellare 等人在 2010 年提出了 FFX 模型,该模型对 FFSEM 模型在消息空间、Feistel 网络等方面进行了扩展。FFX 模型使用了非平衡 Feistel 网络,能够处理消息空间 $Chars^n$ 上的 FPE 问题，$Chars^n$ 为长度(字符个数)为 n 的字符串构成的集合。

FFX 模型通过将固定字符表与其索引表(数字集合)建立双射关系，将字符串中的每个字符编码为数字串，对数字串进行 Feistel 加密运算，从而实现对消息空间 $Chars^n$ 上的保留格式加密。

FFX 模型的算法可以描述为：

算法 setup：

FFX 模型的初始化阶段确定

- 字母表 Chars={$char_0$，$char_1$，…，$char_{radix-1}$}及其基数 radix；
- 确定非平衡 Feistel 网络类型：
- 消息空间 $Chars^n$ 中的字符串长度(字符个数)n：
- Feistel 网络使用的轮次数 r，伪随机函数 f_k 及其运算类型，和调整因子 t 等。

算法 encryption：

输入为明文字符串 x、对称密钥 k 和调整因子 r，输出为密文字符串 y，字符串 x 和 y 同属于消息空间 $Chars^n$，都是由字母表 Chars 中的字符组成的长度为 n 的字符串。

加密过程描述为：

建立字母表 Chars 与 Chars’={0，1，2，…，radix-1}的映射，从而将字符串中的每个字符 $char_i$ 编码为相应的第 i 个数字，注意 Chars’中每个数字前面的 0 和其他字符一样计入长度。举例：Chars={a，b，c，…，z}，x=acz，将字符串 x 编码为 010326。

然后，执行 r 轮的非平衡 Feistel 网络的运算：①将字符串 x 的编码作为输入，并分割为两部分 L 和 R，$|L| \neq |R|$；②执行伪随机函数 f_k，对 L 和 f_k（R）执行下面某种类型的运算得到 L`：(i)$C_i=(a_i+b_i) \bmod radix$，当 radix=2 时，该运算为异或运算；(ii) $\sum c_i radix^{n-1}=(\sum a_i radix^{n-1}+\sum b_i radix^{n-1}) \bmod radix$；③连接 L`与 R 得到输出 L`‖R，并作为下一轮运算的输入。最后，将非平衡 Feistel 网络得到的密文中的数字编码映射回字母表中的字母，从而得到密文 y。

算法 decryption：

解密算法为算法 encryption 的逆过程。

FFX 模型的工作原理如 0 所示。

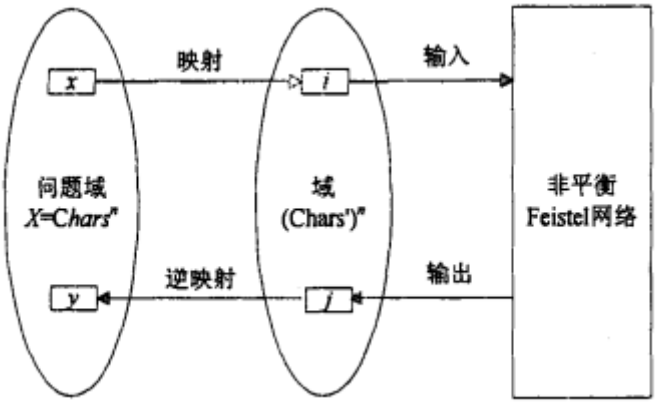


图 4.3 FFX 模型

与 FFSEM 相比，FFX 模型具有更广的适用范围，而且避免 Cycle-walking，具有较高的效率。

4.4 构造特点

已提出的 FPE 模型主要基于 2002 年提出的 FPE 三种基本方法进行设计和构建,已提出的主要模型或方案及其采用的构建方法见 0：

表 4-1 保留格式加密模型或方案

时间	FPE 模型或方案	所解决的问题域	FPE 基本方法			
			Prefix	Cycle-Walking	Generalized-Feistel	其他
2008	社会保险号的 FPE 方案[7]	社会保险号	√		√	
2008	FFSEM[8]	整数集		√	√	
2009	RtE[9]	任意正则语言				√
	基于 Feistel 网络的整数 FPE[9]	整数集			√	
2009	FFX[10]	$chars^n$			√	
2009	基于 Thorp Shuffle 的方案[17]	小型整数集			√	
2010	BPS[13]	$chars^n$			√	
	基于 Type-2 Feistel 网络的整数 FPE[18]	整数集			√	

由 0 可以看出来：

1）通用的 Feistel 方法得到了更多的关注。大多数 FPE 模型，如 FFSEM、FFX 和 BPS 等都使用了 Feistel 网络构造对称密码(RtE 方法的加密模块采用的整数 FPE 方案内部同样基于 Feistel 网络实现)，这是由于 Feistel 网络具有完备的研究成果，并且能够达到 FPE 算法保留格式的要求。

2）在解决复杂消息空间上的 FPE 问题时，通常要与 Cycle-walking 方法结合。由于待加密消息空间的复杂性，很难找到对不同消息空间的 FPE 通用解决办法。Cycle-walking 方法为确保明文和密文在同一消息空间内提供了通用的解决思路，通常在使用 Feistel 网络构造加密模型时和 Cycle-walking 结合用以解决任意有限域上的 FPE 问题。

3）降低问题域的复杂性已经成为 FPE 模型设计思想的发展方向。无论是 FFX 模型还是 RtE 方法，都没有在复杂消息空间上直接解决 FPE 问题，而是将 FPE 问题转化为较低复杂度的整数域上。通过降低复杂性的思路将成为解决复杂问题域上的 FPE 问题的有效研究手段。

5. FPE 标准化进展

保留格式加密的主流设计采用分组密码的工作模式，关于工作模式的标准有很多，例如[27][28]中介绍了五种常见的加密模式 ECB、CBC、CFB、OFB 和 CTR，同时给出了参数选择的一些建议。目前为止，美国 NIST 也发布了一批基于分组密码的工作模式，将其分类如下：其五种加密模式 ECB、CBC、CFB、OFB 和 CTR【SP800-38A】，一种消息鉴别码模式 CMAC【SP800-38B】，一种鉴别加密模式 CCM【SP800-38C】,一种高吞吐的鉴别加密模式 GCM【SP800-38D】，一种专用于存储设备的加密模式 XTS【SP800-38E】,一种密钥封装模式 KW(key wrapping)【SP800-38F】，两种保留格式加密模式 FF1 和 FF3【SP800-38G】。

5.1 NIST SP800-38G 简介

在 NIST SP800-38G 中，介绍了 2 种 FPE 模式，FF1 和 FF3，都是基于 Feistel 的加密模式，由于 FF2 被设计出来的时候不满足期望的 128bit 的安全强度，因此在标准中被弃用。每一个 FPE 模式控制在一个更大的叫做 FFX 的框架中。FF1 和 FF3 之间核心的不同是采用了不同的轮数，FF1 采用了 10 轮，FF3 采用了 8 轮。除了需要提供机密性的加密数据，FF1 和 FF3 都有一个额外的输入参数，“tweak”（公开的参数）；tweak 可以被视为密钥的可变部分，因为他们同时决定了加密解密函数。

对于 FPE，如果输入的数量非常少的情况下，很容易受到暴力破解，所以参数的选择对于影响安全性非常重要。为了防止中间相遇(meet-in-the-middle)攻击，FF1 和 FF3 要求的最低 $radix^{minlen}$ 不小于 100，但是通常建议其值至少为一百万。

5.2 对 NIST SP800-38G 的已知攻击

Bellare 等[20]在 CCS 2016 上，给出对基于 Feistel 结构的 FPE 方案消息恢复攻击。他们指出，当消息空间较小时，可以直接恢复出消息，如对于长度为 8 比特的消息，采用 8 轮 General-Feistel 结构的 FF3 方案，恢复消息的复杂度仅为2^{32}，对于采用 10 轮 General-Feistel 结构的 FF1 方案，复杂度为2^{40}。

在 CRYPTO 2017 上，F.Betiil 与 Serge Vaudenay[21]改进了对 FF1、FF3 方案的攻击，利用了较少的已知明文进行攻击，指出采用 FF1、FF3 方案加密的消息域规模较小时，FF1、FF3 方案均无法提供 128 比特的安全性，攻击复杂度从 8 轮降低为 4 轮，作者同时提出可通过更改 FF3 方案中所采用的 tweak 的长度来提高 FF3 方案的安全性。作为对这一个攻击结果的回应，NIST 在 2017 年 4 月 12 号发布的《Recent Cryptanalysis of FF3》中给出了 FF3 不再适合作为通用的 FPE 算法：“NIST has concluded that FF3 is no longer suitable as a general-purpose FPE method.”。

在 CRYPTO2018 上，[22]中给出了高效的恢复出消息的方案，并且首次提出了针对于非 Feistel 结构的 FPE 攻击。

对于通用的 Feistel 结构的攻击复杂度如下：其中，消息域为$\{0,1\}^{2n}(n \geq 3)$，加密轮数为r，$N = 2^n$，p表示消息数目，获得优势为$1 - 1/N$。

	攻击复杂度
Running time	$O(n^2N^{r-2} + N^{r-2}np)$
Total ciphertexts	$O(n^2N^{r-2} + N^{r-3}np)$
Time per target	$O(n \cdot N^{r-2})$
Ciphertexts per target	$O(n \cdot N^{r-3})$
Ciphertexts per tweak	$O(n \cdot N)$

在 EUROCRYPT 2019 上，Viet Tung Hoang、David Miller 和 Ni Trieu 等[25]人进

一步改进了[21]对 FF3 算法的分析结果，进一步降低了攻击的复杂度。具体说来，[21]提出的攻击需要大约 2^50 次计算来恢复 6 个数字的 PIN 码，而[25]提出的攻击仅需要 2^30 次计算，并给出了具体的测试结果来验证分析结果。[21]和[25]的结果都是利用了 FF3 结构中的调整因子（Tweak）字段来控制哪部分信息异或进轮计数（Round Index）中，也可以通过改变 FF3 算法中的调整因子的使用方法来抵御这两种攻击方式。

在 EUROCRYPT 2021 上,Amon 等构造了对该方案的三种滑动攻击方法[31]，其中利用循环结构的滑动攻击使数据存储、时间复杂度达到最优，分别为$O(N^{11/6})$和$O\left(N^{\frac{17}{6}}\right)$. 该成果对 FF3 方案的实际应用尚不构成直接威胁。原因在于，一方面，区分攻击不同于恢复密钥攻击，论文成果对实用的 FF3 方案密钥不构成威胁；另一方面，该成果的数据复杂度相对较高，我们可以通过调柄（tweak）的使用次数，直接避免这种攻击的发生。在 CRYPTO 2021 上 Beyne 则利用攻击者控制调参的能力,利用线性分析的方法将对 FF3-1 的消息还原攻击的数据复杂性降低为$O(N^{2.5})$[32]。

5.3 NIST SP800-38G 第一版修订

NIST SP800-38G 被不断的爆出攻击漏洞后，NIST 在 2019 年 2 月 28 号发布了 FPE 标准的修正版《Draft NIST Special Publication 800-38G Revision 1 Recommendation for Block Cipher Modes of Operation Methods for Format-Preserving Encryption》，这是通过综合考虑前述各类攻击后，给出的修正后的 FF1 和 FF3 算法结构。

NIST SP800-38G Revision1 中没有更改 FF1 算法的结构或者计算过程，只是对 FF1 算法的适用范围有了更新。之前的标准中 FF1 算法的适用条件是 $radix^{minlen}$ >=100，Revision1 中将该适用条件修正为 $radix^{minlen}$ >=1000000，原先标准中 FF3 的适用范围也从 $radix^{minlen}$ >=100 修正为 Revision1 中 FF3-1 的 $radix^{minlen}$ >=1000000。除了适用范围调整之外，R1 中也对 FF3 算法的计算过程做了调整，具体说来原先标准中 FF3 算法的 64 比特的调整因子（Tweak）在 FF3-1 中调整为 56 比特，这一改动也影响了加解密过程中临时变量 T_L 和 T_R 的构建方式：从 FF3 中的 T_L= T[0..31], T R = T[32..63]调整为 T_L = T [0 .. 27] || 0000, T R = T [32 .. 55] || T [28 .. 31] || 0000，由于具体计算过程的改变，FF3 算法也重命名为 FF3-1。

6. 基于 SM4 算法的 FPE 设计

NIST 800-38G 标准中 FF1 和 FF3 中的轮函数中的 PRF 组件是基于 AES-128 算法构建的，由于 SM4 算法与 AES-128 算法具有相同的分组长度和密钥长度，两个算法都提供了 128 比特的安全强度，并且根据分组密码分析的最新进展可知，目前并不存在任何分析方法对 AES-128 或者 SM4 算法造成影响，因此在 FF1 和 FF3 的轮函数中，可以用 SM4 对 AES-128 算法进行等效替换，而不会影响两个 FPE 算法的安全性。[29]也进一步论证了等效替换的安全性,提出了一种我国商用分组密码算法 SM4 的数字型数据的保留格式加密算法，通过对数字型特征数据进行分段处理后，利用平衡 Feistel 结构或非平衡 Feistel 结构进行轮运算和模运算，在每轮轮运算中用 SM4 加密截断实现 F 函数功能，并对分段加密结果组合后的密文进行效验得到加密后的保留格式密文，所提出的算法能正确实现保留格式加密，扩展了 SM4 的应用，安全性分析表明，提出的算法与 SM4 安全性相当。

为了应对攻击者通过利用特定调参对 FF3 的攻击,[33]提出调参加密的 FPE:TE-FPE。其基本思路是通过对输入调参进行编码后使用分组加密获得调参密文，然后使用调参密文部分数据作为轮函数的调参，使得攻击者无法控制轮函数的调参数据，来抵抗线性分析、关联域攻击、轮函数控制等攻击。TE-FPE 对 FF3 的主要改进包括：1）将输入调参填充到 128 比特 D，对 D 进行分组加密获得密文 Z，采用密文 Z 的部分数据作为轮函数的调参，消除攻击者对轮函数调参的直接控制，抵抗[32]等攻击；2）对 $N \leqslant 2^{64}$ 的数据，将轮函数调参数据增加到 56 比特，并将轮数字段扩展到 8 比特，以支持超过 16 轮的 Feistel 结构；3）通过在调参填充中引入明文长度和基数，抵抗关联域攻击[31]；4）针对在小域消息时，采用可变轮数方案实现小域数据加密时使用更多轮数完成加密，轮数选择使得消息还原攻击数据复杂性大于输入调参和明文的组合空间大小，从而抵抗该类攻击；5)如同 FF3-1，采用[21]中的方法，轮数编码使用独立数据区和其他调参数据分开，使得各轮函数的调参输入一定不同，保证 Feistel 轮函数域的完整分离；6) 对奇数长度的消息，通过消息分割使得右部消息长度大于左部消息，提高整体还原消息的难度。

SM4 算法是国家分组密码算法标准，其算法设计简洁，安全高效，可以抵抗目前已知的攻击，拥有足够的安全冗余度，以国家分组密码标准算法 SM4 为基础，采用 NIST SP800-38G Revision 1 中的 FF1 和 FF3-1 算法。

6.1 SM4-FF1 算法

先决条件：密码算法 SM4；
密钥 K；
基 radix，$\text{radix} \in [2,\ 2^{16}]$;
支持的消息长度$2 \leq \text{minlen} < \text{maxlen} < 2^{32}$；
$radix^{minlen} \geq 1000000$;
Tweak 最大的字节长度记为 maxTlen。

6.1.1 SM4-FF1-Encrypt(K, T, X)

输入：　数字串 X 的长度$\text{n} \in [\text{minlen}..\text{maxlen}]$；
字节串 Tweak，表示为 T。字节串的长度为 t，　$\text{t} \in [0..\text{maxTlen}]$。

输出：　数字串 Y，且 LEN（Y）=n。

步骤：
1. 令$\text{u} = \lfloor n/2 \rfloor$; $\text{v} = \text{n} - \text{u}$。
2. 令$\text{A} = \text{X}[1..\text{u}]; \text{B} = \text{X}[\text{u}+1..\text{n}]$
3. 令$b = \lceil \lceil v \cdot \text{LOG}(radix) \rceil / 8 \rceil$.
4. 令$d = 4\lceil b/4 \rceil + 4$.
5. 令 $P = [1]^1 \| [2]^1 \| [1]^1 \| [radix]^3 \| [10]^1 \| [u \bmod 256]^1 \| [n]^4 \| [t]^4$
6. 对于 i 从 0 到 9

　1）令　$Q = T \| [0]^{(-t-b-1) \bmod 16} \| [i]^1 \| [\text{NUM}_{radix}(B)]^b$.

　2）令　$R = \text{PRF}(P \| Q)$.
　3）令 S 是下面$\lceil d/16 \rceil$个分组的前 d 个字节，即
　$R \| \text{SM4}_K(R \oplus [1]^{16}) \| \text{SM4}_K(R \oplus [2]^{16}) \ldots \text{SM4}_K(R \oplus [\lceil d/16 \rceil - 1]^{16})$.
　4）令　$\text{y} = \text{NUM}(\text{S})$.
　5）如果 i 是偶数，令$\text{m} = \text{u}$，否则令$\text{m} = \text{v}$。

6）令$c = (\mathrm{NUM}_{\mathrm{radix}}(A) + y) \bmod \mathrm{radix}^{m}$.

7）令$C = \mathrm{STR}^{m}_{\mathrm{radix}}(c)$

8）令$A = B$

9）令$B = C$

7. 返回A||B.

6.1.2 SM4-FF1-Decrypt(K, T, X)

输入： 数字串 X 的长度$n \in [minlen..maxlen]$；
字节串 Tweak,表示为 T。其字节串的长度为 $t \in [0..maxTlen]$。

输出： 数字串 Y，且 LEN（Y）=n。

步骤：

1. 令$u = \lfloor n/2 \rfloor$; $v = n - u$。
2. 令$A = X[1..u]; B = X[u+1..n]$
3. 令$b = \lceil \lceil v \cdot \mathrm{LOG}(radix) \rceil / 8 \rceil$.
4. 令$d = 4\lceil b/4 \rceil + 4$.
5. 令 $P = [1]^1 \| [2]^1 \| [1]^1 \| [radix]^3 \| [10]^1 \| [u \bmod 256]^1 \| [n]^4 \| [t]^4$
6. 对于 i 从 9 到 0

1） 令 $Q = T \| [0]^{(-t-b-1) \bmod 16} \| [i]^1 \| [\mathrm{NUM}_{radix}(A)]^b$.

2）令 $R = \mathrm{PRF}(P \| Q)$.

3）令 S 是下面$\lceil d/16 \rceil$个分组的前 d 个字节，即
$R \| \mathrm{SM4}_K(R \oplus [1]^{16}) \| \mathrm{SM4}_K(R \oplus [2]^{16}) \ldots \mathrm{SM4}_K(R \oplus [\lceil d/16 \rceil - 1]^{16})$.

4）令 $y = \mathrm{NUM}(S)$.

5）如果 I 是偶数，令$m = u$，否则令$m = v$。

6）令$c = (\mathrm{NUM}_{\mathrm{radix}}(B) - y) \bmod \mathrm{radix}^{m}$.

7）令$C = \mathrm{STR}^{m}_{\mathrm{radix}}(c)$

8）令$B = A$

9）令$A = C$

7. 返回A||B.

6.2 SM4-FF3-1 算法

先决条件：密码算法 SM4；
密钥 K；
基 radix，$radix \in [2, 2^{16}]$;
支持的消息长度$2 \le minlen \le maxlen < 2\lfloor log_{radix}(2^{96}) \rfloor$;
$radix^{minlen} \ge 1000000$;
Tweak 最大的字节长度记为 maxTlen。

6.2.1 SM4-FF3-1-Encrypt(K, T, X)

输入： 数字串 X 的长度$n \in [minlen..maxlen]$；
比特串 Tweak，表示为 T,其比特串的长度为 t,且 t=56。

输出： 数字串 Y，且 LEN（Y）=n。

步骤：

1. 令$\mathrm{u} = \lfloor \mathrm{n}/2 \rfloor$； $\mathrm{v} = \mathrm{n} - \mathrm{u}$。
2. 令$\mathrm{A} = \mathrm{X}[1..\mathrm{u}]; \mathrm{B} = \mathrm{X}[\mathrm{u}+1..\mathrm{n}]$
3. 令$\mathrm{T_L} = \mathrm{T}[0..27] \,||0^4 \quad \mathrm{T_R} = \mathrm{T}[32..55] \,||\mathrm{T}[28..31]||0^4$
4. 对于 i 从 0 到 7

   1）如果i是偶数，令 $m = u$， $W = T_R$，否则，令 $m = v$ ， $W = T_L$.

   2）令$P = W \oplus [i]^4 \| \left[\mathrm{NUM}_{radix}(\mathrm{REV}(B))\right]^{12}$.

   3）令$S = \mathrm{REVB}\left(\mathrm{SM4}_{\mathrm{REVB}(K)}(\mathrm{REVB}(P))\right)$.

   4）令$y = \mathrm{NUM}(S)$.

   5）令 $c = \left(\mathrm{NUM}_{radix}(\mathrm{REV}(A)) + y\right) \bmod radix^m$.

   6）令$C = \mathrm{REV}\left(\mathrm{STR}^m_{radix}(c)\right)$.

   7）令$A = B$.

   8）令$B = C$.
5. 返回A||B.

6.2.2 SM4-FF3-1-Decrypt(K, T, X)

输入： 数字串 X 的长度$\mathrm{n} \in [\mathrm{minlen}..\mathrm{maxlen}]$；
比特串 Tweak，表示为 T,其比特串的长度为 t,且 t=56。

输出： 数字串 Y，且 LEN（Y）=n。

步骤：

1. 令$\mathrm{u} = \lfloor \mathrm{n}/2 \rfloor$； $\mathrm{v} = \mathrm{n} - \mathrm{u}$。
2. 令$\mathrm{A} = \mathrm{X}[1..\mathrm{u}]; \mathrm{B} = \mathrm{X}[\mathrm{u}+1..\mathrm{n}]$
3. 令$\mathrm{T_L} = \mathrm{T}[0..27] \,||0^4 \quad \mathrm{T_R} = \mathrm{T}[32..55] \,||\mathrm{T}[28..31]||0^4$
4. 对于 i 从 7 到 0

   1）如果i是偶数，令 $m = u$， $W = T_R$，否则，令 $m = v$ ， $W = T_L$.

   2）令$P = W \oplus [i]^4 \| \left[\mathrm{NUM}_{radix}(\mathrm{REV}(A))\right]^{12}$.

   3）令$S = \mathrm{REVB}\left(\mathrm{SM4}_{\mathrm{REVB}(K)}(\mathrm{REVB}(P))\right)$.

   4）令$y = \mathrm{NUM}(S)$.

   5）令 $c = \left(\mathrm{NUM}_{radix}(\mathrm{REV}(B)) - y\right) \bmod radix^m$.

   6）令 $C = \mathrm{REV}\left(\mathrm{STR}^m_{radix}(c)\right)$.

   7）令$B = A$.

   8）令$A = C$.
5. 返回A||B.

6.3 SM4-TE-FPE 算法

先决条件：密码算法 SM4；
密钥 K；
基 radix，$\text{radix} \in [2, 2^{16}]$；
支持的消息长度$2 \le \text{minlen} \le \text{maxlen} < 2\lfloor log_{radix}(2^{96}) \rfloor$；
$radix^{minlen} \ge 100$；
Tweak 最大的字节长度记为 maxTlen。

6.3.1 SM4-TE-FPE-Encrypt(K, T, X)

输入：　数字串X的长度$n \in [\text{minlen}..\text{maxlen}]$；
比特串 Tweak，表示为 T,其比特串的长度为t,且$t = 56$。

输出：　数字串Y，且$\text{LEN(Y)} = n$。
步骤：

1. 令$u = \lfloor n/2 \rfloor$； $v = n - u$.
2. 令$A = \text{X}[1..u]; B = \text{X}[u+1..\text{n}]$.
3. 令$h = \lfloor n \cdot \text{LOG}(radix) + 0.1 \rfloor$.
4. 如果$h \ge 30$，令$r = 8$；否则，如果$h \ge 20$，令$r = 10$；
   否则，如果$h \ge 15$，令$r = 12$；否则，如果$h \ge 10$，令$r = 16$；
   否则，令$r = 22$.
5. 令$\text{D} = [255]^1[radix - 1]^2[\text{aid}]^1[t]^1[\text{T}]^7[0]^1[n]^1[u]^1[r]^1$. 对 SM4，$\text{aid} = 1$.
6. 令$\text{Z} = \text{SM4}_K(\text{D})$.
7. 令$e = \lceil v * \text{LOG}(radix) \rceil$.
8. 如果$e > 64$，令$\text{ZT} = \text{Z}[0..63] \oplus \text{Z}[64..127]$,
   $T_L = \text{ZT}[0..27]||0^4$，$T_R = \text{ZT}[32..59]||0^4$，$d = 4$；
   否则，令$T_L = \text{Z}[0..55]||0^8$，$T_R = \text{Z}[64..119]||0^8$，$d = 8$.
9. 对于i从 0 到$r - 1$

   1）如果i 是偶数，令 $m = u$， $W = T_R$，否则，令 $m = v$ ， $W = T_L$.

   2）令$P = W \oplus [i]^d \| \left[\text{NUM}_{radix}\left(\text{REV}(B)\right)\right]^{16-d}$.

   3）令$S = \text{REVB}\left(\text{SM4}_{\text{REVB}(K)}(\text{REVB}(P))\right)$.

   4）令$y = \text{NUM}(S)$.

   5）令$c = \left(\text{NUM}_{radix}\left(\text{REV}(A)\right) + y\right) \bmod radix^m$.

   6）令$C = \text{REV}\left(\text{STR}^m_{radix}(c)\right)$.

   7）令$A = B$.
   8）令$B = C$.
10. 返回A||B.

6.3.2 SM4-TE-FPE-Decrypt(K, T, X)

输入：　数字串X的长度$n \in [\text{minlen}..\text{maxlen}]$；

比特串 Tweak，表示为 T,其比特串的长度为t,且$t = 56$。

输出：　　数字串Y，且$\mathrm{LEN(Y)} = n$。

步骤：

1. 令$u = \lfloor n/2 \rfloor$；$v = n - u$.
2. 令$A = \mathrm{X}[1..u]; B = \mathrm{X}[u + 1..\mathrm{n}]$.
3. 令$h = \lfloor n \cdot \mathrm{LOG}(radix) + 0.1 \rfloor$.
4. 如果$h \geq 30$，令$r = 8$；否则，如果$h \geq 20$，令$r = 10$；
   否则，如果$h \geq 15$，令$r = 12$；否则，如果$h \geq 10$，令$r = 16$；
   否则，令$r = 22$.
5. 令$\mathrm{D} = [255]^1[radix - 1]^2[\mathrm{aid}]^1[t]^1[\mathrm{T}]^7[0]^1[n]^1[u]^1[r]^1$.对 SM4，$\mathrm{aid} = 1$.
6. 令$\mathrm{Z} = \mathrm{SM4}_K(\mathrm{D})$.
7. 令$e = \lceil v * \mathrm{LOG}(radix) \rceil$.
8. 如果$e > 64$，令$\mathrm{ZT} = \mathrm{Z}[0..63] \oplus \mathrm{Z}[64..127]$,
   $T_L = \mathrm{ZT}[0..27]||0^4$，$T_R = \mathrm{ZT}[32..59]||0^4$，$d = 4$；
   否则，令$T_L = \mathrm{Z}[0..55]||0^8$，$T_R = \mathrm{Z}[64..119]||0^8$，$d = 8$.
9. 对于i从$r - 1$到 0

   1）如果i是偶数，令$m = u$，$W = T_R$，否则，令$m = v$，$W = T_L$.

   2）令$P = W \oplus [i]^d \| \left[\mathrm{NUM}_{radix}\left(\mathrm{REV}(A)\right)\right]^{16-d}$.

   3）令$S = \mathrm{REVB}\left(\mathrm{SM4}_{\mathrm{REVB}(K)}(\mathrm{REVB}(P))\right)$.

   4）令$y = \mathrm{NUM}(S)$.

   5）令$c = \left(\mathrm{NUM}_{radix}\left(\mathrm{REV}(B)\right) - y\right) \bmod radix^m$.

   6）令$C = \mathrm{REV}\left(\mathrm{STR}_{radix}^m(c)\right)$.

   7）令$B = A$.

   8）令$A = C$.
10. 返回A||B.

6.4 基于 SM4 算法的 FPE 设计的测试向量

T 是 Tweak 调整因子，key 是密钥，Round 是轮标号。

1） 其中对于 FF1 算法：

加密过程中：

PT/ X 是明文输入，Q、R、S、Y 是中间的过程值，A、B 是每一轮结束的值，CT 是最后的密文输出。

解密过程中：

X 是密文输入，Q、R、S、Y 是中间的过程值，A、B 是每一轮结束的值，PT 是解密的明文输出。

2） 对于 FF3-1 算法：

加密过程中：

PT/ X 是明文输入，W、P、S、Y 是中间的过程值，A、B 是每一轮结束的值，CT 是最后的密文输出。

解密过程中：

X 是密文输入，W、P、S、Y 是中间的过程值，A、B 是每一轮结束的值，PT 是解密的明文输出。

6.4.1 FF1 示例

1) 基于银行卡号的 FF1 测试向量

```
Sample #1-1

FF1-SM4

Key is  <2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c > (hex)
Radix = 10
----------------------------------

PT is   <6226090102675688>
FF1.Encrypt()

X is    [ 6, 2, 2, 6, 0, 9, 0, 1, 0, 2, 6, 7, 5, 6, 8, 8, ]
T is    <39 38 37 36 35 34 33 32 31 30 > (hex)
u is    8
v is    8
A is    [ 6, 2, 2, 6, 0, 9, 0, 1, ]
B is    [ 0, 2, 6, 7, 5, 6, 8, 8, ]
b is    4
d is    8
P is    [ 1, 2, 1, 0, 0, 10, 10, 8, 0, 0, 0, 16, 0, 0, 0, 10, ]
Round #0
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 0, 0, 40, 211, 232, ]
R is    [ 16, 49, 101, 255, 13, 221, 180, 133, 126, 35, 224, 21, 215, 226, 214,
184, ]
S is    103165ff0dddb485
Y is    1166825924589368453
m is    8
C is    [ 5, 1, 6, 2, 9, 3, 5, 4, ]
A is    [ 0, 2, 6, 7, 5, 6, 8, 8, ]
B is    [ 5, 1, 6, 2, 9, 3, 5, 4, ]
Round #1
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 1, 3, 19, 205, 42, ]
R is    [ 26, 182, 209, 105, 255, 1, 93, 143, 128, 55, 221, 212, 141, 254, 204,
192, ]
S is    1ab6d169ff015d8f
```

```
Y is    1924956143927516559
m is    8
C is    [ 3, 0, 1, 9, 2, 2, 4, 7, ]
A is    [ 5, 1, 6, 2, 9, 3, 5, 4, ]
B is    [ 3, 0, 1, 9, 2, 2, 4, 7, ]
Round #2
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 2, 1, 204, 178, 119, ]
R is    [ 249, 164, 221, 143, 233, 137, 25, 134, 12, 40, 250, 0, 252, 100, 188,
74, ]
S is    f9a4dd8fe9891986
Y is    17988746421792741766
m is    8
C is    [ 4, 4, 3, 7, 1, 1, 2, 0, ]
A is    [ 3, 0, 1, 9, 2, 2, 4, 7, ]
B is    [ 4, 4, 3, 7, 1, 1, 2, 0, ]
Round #3
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 3, 2, 165, 12, 176, ]
R is    [ 228, 246, 47, 173, 30, 110, 7, 159, 226, 73, 57, 238, 53, 3, 68, 178, ]
S is    e4f62fad1e6e079f
Y is    16498426705504765855
m is    8
C is    [ 3, 4, 9, 5, 8, 1, 0, 2, ]
A is    [ 4, 4, 3, 7, 1, 1, 2, 0, ]
B is    [ 3, 4, 9, 5, 8, 1, 0, 2, ]
Round #4
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 4, 2, 21, 107, 22, ]
R is    [ 240, 3, 188, 30, 245, 72, 216, 131, 5, 166, 234, 233, 159, 47, 37,
211, ]
S is    f003bc1ef548d883
Y is    17294873835183069315
m is    8
C is    [ 2, 7, 4, 4, 0, 4, 3, 5, ]
A is    [ 3, 4, 9, 5, 8, 1, 0, 2, ]
B is    [ 2, 7, 4, 4, 0, 4, 3, 5, ]
Round #5
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 5, 1, 162, 181, 51, ]
R is    [ 78, 141, 94, 44, 3, 169, 80, 7, 147, 29, 71, 246, 6, 34, 236, 211, ]
S is    4e8d5e2c03a95007
Y is    5660283849807581191
m is    8
C is    [ 4, 2, 5, 3, 9, 2, 9, 3, ]
A is    [ 2, 7, 4, 4, 0, 4, 3, 5, ]
B is    [ 4, 2, 5, 3, 9, 2, 9, 3, ]
Round #6
```

```
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 6, 2, 137, 25, 29, ]
R is    [ 79, 29, 183, 225, 27, 111, 94, 164, 172, 153, 35, 29, 233, 158, 43,
236, ]
S is    4f1db7e11b6f5ea4
Y is    5700914880776724132
m is    8
C is    [ 0, 4, 1, 6, 4, 5, 6, 7, ]
A is    [ 4, 2, 5, 3, 9, 2, 9, 3, ]
B is    [ 0, 4, 1, 6, 4, 5, 6, 7, ]
Round #7
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 7, 0, 63, 139, 215, ]
R is    [ 166, 167, 205, 82, 6, 8, 74, 208, 213, 75, 8, 174, 60, 155, 226, 182, ]
S is    a6a7cd5206084ad0
Y is    12008792683578936016
m is    8
C is    [ 2, 1, 4, 7, 5, 3, 0, 9, ]
A is    [ 0, 4, 1, 6, 4, 5, 6, 7, ]
B is    [ 2, 1, 4, 7, 5, 3, 0, 9, ]
Round #8
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 8, 1, 71, 175, 237, ]
R is    [ 67, 198, 50, 123, 55, 62, 184, 237, 148, 7, 243, 172, 99, 18, 135,
69, ]
S is    43c6327b373eb8ed
Y is    4883646350719105261
m is    8
C is    [ 2, 3, 2, 6, 9, 8, 2, 8, ]
A is    [ 2, 1, 4, 7, 5, 3, 0, 9, ]
B is    [ 2, 3, 2, 6, 9, 8, 2, 8, ]
Round #9
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 9, 1, 99, 17, 196, ]
R is    [ 168, 149, 249, 207, 215, 176, 237, 136, 47, 4, 141, 241, 30, 109, 219,
250, ]
S is    a895f9cfd7b0ed88
Y is    12147890240974024072
m is    8
C is    [ 9, 5, 4, 9, 9, 3, 8, 1, ]
A is    [ 2, 3, 2, 6, 9, 8, 2, 8, ]
B is    [ 9, 5, 4, 9, 9, 3, 8, 1, ]

A || B is       [ 2, 3, 2, 6, 9, 8, 2, 8, 9, 5, 4, 9, 9, 3, 8, 1, ]
CT is   <2326982895499381>

------------------------------------------------------------------------
```

FF1.Decrypt()

```
X is    [ 2, 3, 2, 6, 9, 8, 2, 8, 9, 5, 4, 9, 9, 3, 8, 1, ]
T is    <39 38 37 36 35 34 33 32 31 30 > (hex)
u is    8
v is    8
A is    [ 2, 3, 2, 6, 9, 8, 2, 8, ]
B is    [ 9, 5, 4, 9, 9, 3, 8, 1, ]
b is    4
d is    8
P is    [ 1, 2, 1, 0, 0, 10, 10, 8, 0, 0, 0, 16, 0, 0, 0, 10, ]
Round #9
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 9, 1, 99, 17, 196, ]
R is    [ 168, 149, 249, 207, 215, 176, 237, 136, 47, 4, 141, 241, 30, 109, 219,
250, ]
S is    a895f9cfd7b0ed88
Y is    12147890240974024072
m is    8
C is    [ 2, 1, 4, 7, 5, 3, 0, 9, ]
B is    [ 2, 3, 2, 6, 9, 8, 2, 8, ]
A is    [ 2, 1, 4, 7, 5, 3, 0, 9, ]
Round #8
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 8, 1, 71, 175, 237, ]
R is    [ 67, 198, 50, 123, 55, 62, 184, 237, 148, 7, 243, 172, 99, 18, 135,
69, ]
S is    43c6327b373eb8ed
Y is    4883646350719105261
m is    8
C is    [ 0, 4, 1, 6, 4, 5, 6, 7, ]
B is    [ 2, 1, 4, 7, 5, 3, 0, 9, ]
A is    [ 0, 4, 1, 6, 4, 5, 6, 7, ]
Round #7
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 7, 0, 63, 139, 215, ]
R is    [ 166, 167, 205, 82, 6, 8, 74, 208, 213, 75, 8, 174, 60, 155, 226, 182, ]
S is    a6a7cd5206084ad0
Y is    12008792683578936016
m is    8
C is    [ 4, 2, 5, 3, 9, 2, 9, 3, ]
B is    [ 0, 4, 1, 6, 4, 5, 6, 7, ]
A is    [ 4, 2, 5, 3, 9, 2, 9, 3, ]
Round #6
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 6, 2, 137, 25, 29, ]
R is    [ 79, 29, 183, 225, 27, 111, 94, 164, 172, 153, 35, 29, 233, 158, 43,
236, ]
```

S is 4f1db7e11b6f5ea4
Y is 5700914880776724132
m is 8
C is [2, 7, 4, 4, 0, 4, 3, 5,]
B is [4, 2, 5, 3, 9, 2, 9, 3,]
A is [2, 7, 4, 4, 0, 4, 3, 5,]
Round #5
Q is [57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 5, 1, 162, 181, 51,]
R is [78, 141, 94, 44, 3, 169, 80, 7, 147, 29, 71, 246, 6, 34, 236, 211,]
S is 4e8d5e2c03a95007
Y is 5660283849807581191
m is 8
C is [3, 4, 9, 5, 8, 1, 0, 2,]
B is [2, 7, 4, 4, 0, 4, 3, 5,]
A is [3, 4, 9, 5, 8, 1, 0, 2,]
Round #4
Q is [57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 4, 2, 21, 107, 22,]
R is [240, 3, 188, 30, 245, 72, 216, 131, 5, 166, 234, 233, 159, 47, 37, 211,]
S is f003bc1ef548d883
Y is 17294873835183069315
m is 8
C is [4, 4, 3, 7, 1, 1, 2, 0,]
B is [3, 4, 9, 5, 8, 1, 0, 2,]
A is [4, 4, 3, 7, 1, 1, 2, 0,]
Round #3
Q is [57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 3, 2, 165, 12, 176,]
R is [228, 246, 47, 173, 30, 110, 7, 159, 226, 73, 57, 238, 53, 3, 68, 178,]
S is e4f62fad1e6e079f
Y is 16498426705504765855
m is 8
C is [3, 0, 1, 9, 2, 2, 4, 7,]
B is [4, 4, 3, 7, 1, 1, 2, 0,]
A is [3, 0, 1, 9, 2, 2, 4, 7,]
Round #2
Q is [57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 2, 1, 204, 178, 119,]
R is [249, 164, 221, 143, 233, 137, 25, 134, 12, 40, 250, 0, 252, 100, 188, 74,]
S is f9a4dd8fe9891986
Y is 17988746421792741766
m is 8
C is [5, 1, 6, 2, 9, 3, 5, 4,]
B is [3, 0, 1, 9, 2, 2, 4, 7,]
A is [5, 1, 6, 2, 9, 3, 5, 4,]

```
Round #1
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 1, 3, 19, 205, 42, ]
R is    [ 26, 182, 209, 105, 255, 1, 93, 143, 128, 55, 221, 212, 141, 254, 204,
192, ]
S is    1ab6d169ff015d8f
Y is    1924956143927516559
m is    8
C is    [ 0, 2, 6, 7, 5, 6, 8, 8, ]
B is    [ 5, 1, 6, 2, 9, 3, 5, 4, ]
A is    [ 0, 2, 6, 7, 5, 6, 8, 8, ]
Round #0
Q is    [ 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 0, 0, 0, 40, 211, 232, ]
R is    [ 16, 49, 101, 255, 13, 221, 180, 133, 126, 35, 224, 21, 215, 226, 214,
184, ]
S is    103165ff0dddb485
Y is    1166825924589368453
m is    8
C is    [ 6, 2, 2, 6, 0, 9, 0, 1, ]
B is    [ 0, 2, 6, 7, 5, 6, 8, 8, ]
A is    [ 6, 2, 2, 6, 0, 9, 0, 1, ]

A || B is       [ 6, 2, 2, 6, 0, 9, 0, 1, 0, 2, 6, 7, 5, 6, 8, 8, ]
PT is   <6226090102675688>
```

2) 基于身份证号的 FF1 测试向量

```
Sample #1-2

FF1-SM4

Key is  <2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c > (hex)
Radix = 10
-----------------------------------

PT is   <110107197203192876>
FF1.Encrypt()

X is    [ 1, 1, 0, 1, 0, 7, 1, 9, 7, 2, 0, 3, 1, 9, 2, 8, 7, 6, ]
T is    <37 38 39 36 70 71 72 73 74 75 76 > (hex)
u is    9
v is    9
A is    [ 1, 1, 0, 1, 0, 7, 1, 9, 7, ]
B is    [ 2, 0, 3, 1, 9, 2, 8, 7, 6, ]
b is    4
d is    8
```

P is [1, 2, 1, 0, 0, 10, 10, 9, 0, 0, 0, 18, 0, 0, 0, 11,]
Round #0
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 12, 28, 122, 44,]
R is [130, 217, 127, 124, 248, 196, 220, 32, 50, 135, 212, 211, 81, 153, 37, 7,]
S is 82d97f7cf8c4dc20
Y is 9428707469603167264
m is 9
C is [7, 1, 3, 2, 7, 4, 4, 6, 1,]
A is [2, 0, 3, 1, 9, 2, 8, 7, 6,]
B is [7, 1, 3, 2, 7, 4, 4, 6, 1,]
Round #1
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 1, 42, 131, 180, 93,]
R is [19, 26, 87, 210, 84, 199, 197, 162, 251, 137, 230, 227, 87, 214, 179, 62,]
S is 131a57d254c7c5a2
Y is 1376509196992234914
m is 9
C is [1, 9, 5, 4, 2, 7, 7, 9, 0,]
A is [7, 1, 3, 2, 7, 4, 4, 6, 1,]
B is [1, 9, 5, 4, 2, 7, 7, 9, 0,]
Round #2
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 2, 11, 165, 253, 206,]
R is [66, 173, 103, 127, 207, 35, 145, 51, 79, 152, 249, 130, 119, 146, 5, 141,]
S is 42ad677fcf239133
Y is 4804610176107909427
m is 9
C is [8, 2, 1, 1, 8, 3, 8, 8, 8,]
A is [1, 9, 5, 4, 2, 7, 7, 9, 0,]
B is [8, 2, 1, 1, 8, 3, 8, 8, 8,]
Round #3
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 3, 48, 242, 69, 144,]
R is [168, 120, 195, 200, 233, 72, 231, 57, 123, 92, 103, 222, 47, 60, 245, 178,]
S is a878c3c8e948e739
Y is 12139668063251916601
m is 9
C is [4, 4, 7, 3, 4, 4, 3, 9, 1,]
A is [8, 2, 1, 1, 8, 3, 8, 8, 8,]
B is [4, 4, 7, 3, 4, 4, 3, 9, 1,]

Round #4
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 4, 26, 169, 239, 7,]
R is [145, 248, 122, 92, 227, 26, 97, 134, 237, 132, 174, 155, 141, 173, 124, 19,]
S is 91f87a5ce31a6186
Y is 10518291469089530246
m is 9
C is [9, 1, 0, 7, 1, 4, 1, 3, 4,]
A is [4, 4, 7, 3, 4, 4, 3, 9, 1,]
B is [9, 1, 0, 7, 1, 4, 1, 3, 4,]
Round #5
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 5, 54, 72, 101, 22,]
R is [63, 121, 251, 16, 194, 197, 82, 78, 154, 77, 47, 103, 126, 187, 26, 102,]
S is 3f79fb10c2c5524e
Y is 4573962945977209422
m is 9
C is [4, 2, 4, 5, 5, 3, 8, 1, 3,]
A is [9, 1, 0, 7, 1, 4, 1, 3, 4,]
B is [4, 2, 4, 5, 5, 3, 8, 1, 3,]
Round #6
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 6, 25, 78, 45, 85,]
R is [236, 191, 15, 128, 143, 97, 211, 107, 69, 205, 239, 222, 105, 254, 192, 101,]
S is ecbf0f808f61d36b
Y is 17059370958338511723
m is 9
C is [2, 4, 9, 2, 2, 5, 8, 5, 7,]
A is [4, 2, 4, 5, 5, 3, 8, 1, 3,]
B is [2, 4, 9, 2, 2, 5, 8, 5, 7,]
Round #7
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 7, 14, 218, 226, 129,]
R is [9, 86, 255, 139, 129, 240, 20, 152, 219, 7, 74, 105, 151, 147, 186, 104,]
S is 0956ff8b81f01498
Y is 0673006168983999640
m is 9
C is [4, 0, 8, 5, 5, 3, 4, 5, 3,]
A is [2, 4, 9, 2, 2, 5, 8, 5, 7,]
B is [4, 0, 8, 5, 5, 3, 4, 5, 3,]
Round #8

```
Q is    [ 55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 8, 24, 90, 7, 237, ]
R is    [ 156, 176, 21, 87, 195, 166, 33, 194, 237, 89, 158, 61, 181, 129, 174,
70, ]
S is    9cb01557c3a621c2
Y is    11290547732506616258
m is    9
C is    [ 7, 5, 5, 8, 4, 2, 1, 1, 5, ]
A is    [ 4, 0, 8, 5, 5, 3, 4, 5, 3, ]
B is    [ 7, 5, 5, 8, 4, 2, 1, 1, 5, ]
Round #9
Q is    [ 55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 9, 45, 13, 60, 67, ]
R is    [ 110, 39, 147, 20, 144, 253, 244, 240, 23, 59, 73, 132, 147, 167, 230,
195, ]
S is    6e27931490fdf4f0
Y is    7937474584804979952
m is    9
C is    [ 2, 1, 3, 5, 3, 3, 4, 0, 5, ]
A is    [ 7, 5, 5, 8, 4, 2, 1, 1, 5, ]
B is    [ 2, 1, 3, 5, 3, 3, 4, 0, 5, ]

A || B is       [ 7, 5, 5, 8, 4, 2, 1, 1, 5, 2, 1, 3, 5, 3, 3, 4, 0, 5, ]
CT is   <7558421152135334O5>

------------------------------------------------------------------------

FF1.Decrypt()

X is    [ 7, 5, 5, 8, 4, 2, 1, 1, 5, 2, 1, 3, 5, 3, 3, 4, 0, 5, ]
T is    <37 38 39 36 70 71 72 73 74 75 76 > (hex)
u is    9
v is    9
A is    [ 7, 5, 5, 8, 4, 2, 1, 1, 5, ]
B is    [ 2, 1, 3, 5, 3, 3, 4, 0, 5, ]
b is    4
d is    8
P is    [ 1, 2, 1, 0, 0, 10, 10, 9, 0, 0, 0, 18, 0, 0, 0, 11, ]
Round #9
Q is    [ 55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 9, 45, 13, 60, 67, ]
R is    [ 110, 39, 147, 20, 144, 253, 244, 240, 23, 59, 73, 132, 147, 167, 230,
195, ]
S is    6e27931490fdf4f0
Y is    7937474584804979952
m is    9
C is    [ 4, 0, 8, 5, 5, 3, 4, 5, 3, ]
```

B is [7, 5, 5, 8, 4, 2, 1, 1, 5,]
A is [4, 0, 8, 5, 5, 3, 4, 5, 3,]
Round #8
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 8, 24, 90, 7, 237,]
R is [156, 176, 21, 87, 195, 166, 33, 194, 237, 89, 158, 61, 181, 129, 174, 70,]
S is 9cb01557c3a621c2
Y is 11290547732506616258
m is 9
C is [2, 4, 9, 2, 2, 5, 8, 5, 7,]
B is [4, 0, 8, 5, 5, 3, 4, 5, 3,]
A is [2, 4, 9, 2, 2, 5, 8, 5, 7,]
Round #7
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 7, 14, 218, 226, 129,]
R is [9, 86, 255, 139, 129, 240, 20, 152, 219, 7, 74, 105, 151, 147, 186, 104,]
S is 0956ff8b81f01498
Y is 0673006168983999640
m is 9
C is [4, 2, 4, 5, 5, 3, 8, 1, 3,]
B is [2, 4, 9, 2, 2, 5, 8, 5, 7,]
A is [4, 2, 4, 5, 5, 3, 8, 1, 3,]
Round #6
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 6, 25, 78, 45, 85,]
R is [236, 191, 15, 128, 143, 97, 211, 107, 69, 205, 239, 222, 105, 254, 192, 101,]
S is ecbf0f808f61d36b
Y is 17059370958338511723
m is 9
C is [9, 1, 0, 7, 1, 4, 1, 3, 4,]
B is [4, 2, 4, 5, 5, 3, 8, 1, 3,]
A is [9, 1, 0, 7, 1, 4, 1, 3, 4,]
Round #5
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 5, 54, 72, 101, 22,]
R is [63, 121, 251, 16, 194, 197, 82, 78, 154, 77, 47, 103, 126, 187, 26, 102,]
S is 3f79fb10c2c5524e
Y is 4573962945977209422
m is 9
C is [4, 4, 7, 3, 4, 4, 3, 9, 1,]
B is [9, 1, 0, 7, 1, 4, 1, 3, 4,]
A is [4, 4, 7, 3, 4, 4, 3, 9, 1,]

Round #4
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 4, 26, 169, 239, 7,]
R is [145, 248, 122, 92, 227, 26, 97, 134, 237, 132, 174, 155, 141, 173, 124, 19,]
S is 91f87a5ce31a6186
Y is 10518291469089530246
m is 9
C is [8, 2, 1, 1, 8, 3, 8, 8, 8,]
B is [4, 4, 7, 3, 4, 4, 3, 9, 1,]
A is [8, 2, 1, 1, 8, 3, 8, 8, 8,]
Round #3
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 3, 48, 242, 69, 144,]
R is [168, 120, 195, 200, 233, 72, 231, 57, 123, 92, 103, 222, 47, 60, 245, 178,]
S is a878c3c8e948e739
Y is 12139668063251916601
m is 9
C is [1, 9, 5, 4, 2, 7, 7, 9, 0,]
B is [8, 2, 1, 1, 8, 3, 8, 8, 8,]
A is [1, 9, 5, 4, 2, 7, 7, 9, 0,]
Round #2
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 2, 11, 165, 253, 206,]
R is [66, 173, 103, 127, 207, 35, 145, 51, 79, 152, 249, 130, 119, 146, 5, 141,]
S is 42ad677fcf239133
Y is 4804610176107909427
m is 9
C is [7, 1, 3, 2, 7, 4, 4, 6, 1,]
B is [1, 9, 5, 4, 2, 7, 7, 9, 0,]
A is [7, 1, 3, 2, 7, 4, 4, 6, 1,]
Round #1
Q is [55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 1, 42, 131, 180, 93,]
R is [19, 26, 87, 210, 84, 199, 197, 162, 251, 137, 230, 227, 87, 214, 179, 62,]
S is 131a57d254c7c5a2
Y is 1376509196992234914
m is 9
C is [2, 0, 3, 1, 9, 2, 8, 7, 6,]
B is [7, 1, 3, 2, 7, 4, 4, 6, 1,]
A is [2, 0, 3, 1, 9, 2, 8, 7, 6,]

```
Round #0
Q is    [ 55, 56, 57, 54, 112, 113, 114, 115, 116, 117, 118, 12, 28, 122, 44, ]
R is    [ 130, 217, 127, 124, 248, 196, 220, 32, 50, 135, 212, 211, 81, 153,
37, 7, ]
S is    82d97f7cf8c4dc20
Y is    9428707469603167264
m is    9
C is    [ 1, 1, 0, 1, 0, 7, 1, 9, 7, ]
B is    [ 2, 0, 3, 1, 9, 2, 8, 7, 6, ]
A is    [ 1, 1, 0, 1, 0, 7, 1, 9, 7, ]

A || B is       [ 1, 1, 0, 1, 0, 7, 1, 9, 7, 2, 0, 3, 1, 9, 2, 8, 7, 6, ]
PT is   <1101071972031928 76>
```

3) 基于手机号的 FF1 测试向量

```
Sample #1-3

FF1-SM4

Key is  <2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c > (hex)
Radix = 10
----------------------------------

PT is   <13687260594>
FF1.Encrypt()

X is    [ 1, 3, 6, 8, 7, 2, 6, 0, 5, 9, 4, ]
T is    <> (hex)
u is    5
v is    6
A is    [ 1, 3, 6, 8, 7, ]
B is    [ 2, 6, 0, 5, 9, 4, ]
b is    3
d is    8
P is    [ 1, 2, 1, 0, 0, 10, 10, 5, 0, 0, 0, 11, 0, 0, 0, 0, ]
Round #0
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 249, 242, ]
R is    [ 205, 189, 217, 167, 210, 102, 8, 113, 58, 171, 73, 194, 152, 169, 179,
250, ]
S is    cdbdd9a7d2660871
Y is    14825244863186208881
m is    5
C is    [ 2, 2, 5, 6, 8, ]
A is    [ 2, 6, 0, 5, 9, 4, ]
```

```
B is    [ 2, 2, 5, 6, 8, ]
Round #1
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 88, 40, ]
R is    [ 18, 65, 187, 228, 42, 220, 225, 163, 58, 238, 132, 83, 34, 126, 27,
9, ]
S is    1241bbe42adce1a3
Y is    1315539154814951843
m is    6
C is    [ 2, 1, 2, 4, 3, 7, ]
A is    [ 2, 2, 5, 6, 8, ]
B is    [ 2, 1, 2, 4, 3, 7, ]
Round #2
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 3, 61, 213, ]
R is    [ 127, 204, 126, 220, 216, 24, 186, 252, 109, 76, 132, 60, 225, 17, 32,
55, ]
S is    7fcc7edcd818bafc
Y is    9208874825049225980
m is    5
C is    [ 4, 8, 5, 4, 8, ]
A is    [ 2, 1, 2, 4, 3, 7, ]
B is    [ 4, 8, 5, 4, 8, ]
Round #3
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 189, 164, ]
R is    [ 245, 152, 2, 250, 122, 153, 164, 68, 226, 23, 199, 130, 91, 203, 177,
104, ]
S is    f59802fa7a99a444
Y is    17696898010574332996
m is    6
C is    [ 5, 4, 5, 4, 3, 3, ]
A is    [ 4, 8, 5, 4, 8, ]
B is    [ 5, 4, 5, 4, 3, 3, ]
Round #4
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 8, 82, 153, ]
R is    [ 205, 238, 166, 57, 96, 112, 223, 129, 145, 120, 7, 231, 227, 127, 66,
177, ]
S is    cdeea6396070df81
Y is    148389805875937196 81
m is    5
C is    [ 6, 8, 2, 2, 9, ]
A is    [ 5, 4, 5, 4, 3, 3, ]
B is    [ 6, 8, 2, 2, 9, ]
Round #5
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 1, 10, 133, ]
```

```
R is    [ 11, 114, 100, 106, 115, 209, 24, 140, 203, 72, 96, 64, 107, 135, 139,
9, ]
S is    0b72646a73d1188c
Y is    0824832090134616204
m is    6
C is    [ 1, 6, 1, 6, 3, 7, ]
A is    [ 6, 8, 2, 2, 9, ]
B is    [ 1, 6, 1, 6, 3, 7, ]
Round #6
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 2, 119, 101, ]
R is    [ 55, 241, 247, 9, 226, 63, 246, 227, 78, 16, 151, 252, 12, 158, 139,
243, ]
S is    37f1f709e23ff6e3
Y is    4031274763295913699
m is    5
C is    [ 8, 1, 9, 2, 8, ]
A is    [ 1, 6, 1, 6, 3, 7, ]
B is    [ 8, 1, 9, 2, 8, ]
Round #7
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 64, 8, ]
R is    [ 181, 21, 65, 238, 22, 176, 38, 228, 51, 109, 242, 140, 4, 116, 34,
166, ]
S is    b51541ee16b026e4
Y is    13048407986214545124
m is    6
C is    [ 7, 0, 6, 7, 6, 1, ]
A is    [ 8, 1, 9, 2, 8, ]
B is    [ 7, 0, 6, 7, 6, 1, ]
Round #8
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 10, 200, 201, ]
R is    [ 221, 181, 74, 7, 81, 49, 238, 210, 217, 243, 217, 148, 37, 40, 156,
81, ]
S is    ddb54a075131eed2
Y is    15975756648454155986
m is    5
C is    [ 3, 7, 9, 1, 4, ]
A is    [ 7, 0, 6, 7, 6, 1, ]
B is    [ 3, 7, 9, 1, 4, ]
Round #9
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 148, 26, ]
R is    [ 21, 253, 136, 134, 110, 205, 184, 227, 120, 47, 173, 60, 244, 64, 145,
59, ]
S is    15fd88866ecdb8e3
Y is    1584572754870253795
```

```
m is    6
C is    [ 9, 6, 0, 5, 5, 6, ]
A is    [ 3, 7, 9, 1, 4, ]
B is    [ 9, 6, 0, 5, 5, 6, ]

A || B is       [ 3, 7, 9, 1, 4, 9, 6, 0, 5, 5, 6, ]
CT is   <37914960556>

------------------------------------------------------------------------

FF1.Decrypt()

X is    [ 3, 7, 9, 1, 4, 9, 6, 0, 5, 5, 6, ]
T is    <> (hex)
u is    5
v is    6
A is    [ 3, 7, 9, 1, 4, ]
B is    [ 9, 6, 0, 5, 5, 6, ]
b is    3
d is    8
P is    [ 1, 2, 1, 0, 0, 10, 10, 5, 0, 0, 0, 11, 0, 0, 0, 0, ]
Round #9
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 148, 26, ]
R is    [ 21, 253, 136, 134, 110, 205, 184, 227, 120, 47, 173, 60, 244, 64, 145,
59, ]
S is    15fd88866ecdb8e3
Y is    15845727548702537 95
m is    6
C is    [ 7, 0, 6, 7, 6, 1, ]
B is    [ 3, 7, 9, 1, 4, ]
A is    [ 7, 0, 6, 7, 6, 1, ]
Round #8
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 10, 200, 201, ]
R is    [ 221, 181, 74, 7, 81, 49, 238, 210, 217, 243, 217, 148, 37, 40, 156,
81, ]
S is    ddb54a075131eed2
Y is    15975756648454155986
m is    5
C is    [ 8, 1, 9, 2, 8, ]
B is    [ 7, 0, 6, 7, 6, 1, ]
A is    [ 8, 1, 9, 2, 8, ]
Round #7
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 64, 8, ]
```

R is [181, 21, 65, 238, 22, 176, 38, 228, 51, 109, 242, 140, 4, 116, 34, 166,]
S is b51541ee16b026e4
Y is 13048407986214545124
m is 6
C is [1, 6, 1, 6, 3, 7,]
B is [8, 1, 9, 2, 8,]
A is [1, 6, 1, 6, 3, 7,]
Round #6
Q is [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 2, 119, 101,]
R is [55, 241, 247, 9, 226, 63, 246, 227, 78, 16, 151, 252, 12, 158, 139, 243,]
S is 37f1f709e23ff6e3
Y is 4031274763295913699
m is 5
C is [6, 8, 2, 2, 9,]
B is [1, 6, 1, 6, 3, 7,]
A is [6, 8, 2, 2, 9,]
Round #5
Q is [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 1, 10, 133,]
R is [11, 114, 100, 106, 115, 209, 24, 140, 203, 72, 96, 64, 107, 135, 139, 9,]
S is 0b72646a73d1188c
Y is 0824832090134616204
m is 6
C is [5, 4, 5, 4, 3, 3,]
B is [6, 8, 2, 2, 9,]
A is [5, 4, 5, 4, 3, 3,]
Round #4
Q is [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 8, 82, 153,]
R is [205, 238, 166, 57, 96, 112, 223, 129, 145, 120, 7, 231, 227, 127, 66, 177,]
S is cdeea6396070df81
Y is 148389805875937196 81
m is 5
C is [4, 8, 5, 4, 8,]
B is [5, 4, 5, 4, 3, 3,]
A is [4, 8, 5, 4, 8,]
Round #3
Q is [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 189, 164,]
R is [245, 152, 2, 250, 122, 153, 164, 68, 226, 23, 199, 130, 91, 203, 177, 104,]
S is f59802fa7a99a444
Y is 17696898010574332996

```
m is    6
C is    [ 2, 1, 2, 4, 3, 7, ]
B is    [ 4, 8, 5, 4, 8, ]
A is    [ 2, 1, 2, 4, 3, 7, ]
Round #2
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 3, 61, 213, ]
R is    [ 127, 204, 126, 220, 216, 24, 186, 252, 109, 76, 132, 60, 225, 17, 32,
55, ]
S is    7fcc7edcd818bafc
Y is    9208874825049225980
m is    5
C is    [ 2, 2, 5, 6, 8, ]
B is    [ 2, 1, 2, 4, 3, 7, ]
A is    [ 2, 2, 5, 6, 8, ]
Round #1
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 88, 40, ]
R is    [ 18, 65, 187, 228, 42, 220, 225, 163, 58, 238, 132, 83, 34, 126, 27,
9, ]
S is    1241bbe42adce1a3
Y is    1315539154814951843
m is    6
C is    [ 2, 6, 0, 5, 9, 4, ]
B is    [ 2, 2, 5, 6, 8, ]
A is    [ 2, 6, 0, 5, 9, 4, ]
Round #0
Q is    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 249, 242, ]
R is    [ 205, 189, 217, 167, 210, 102, 8, 113, 58, 171, 73, 194, 152, 169, 179,
250, ]
S is    cdbdd9a7d2660871
Y is    14825244863186208881
m is    5
C is    [ 1, 3, 6, 8, 7, ]
B is    [ 2, 6, 0, 5, 9, 4, ]
A is    [ 1, 3, 6, 8, 7, ]

A || B is       [ 1, 3, 6, 8, 7, 2, 6, 0, 5, 9, 4, ]
PT is   <13687260594>
```

6.4.2 FF3-1 示例

1) 基于银行卡号 FF3-1 的测试向量

```
Sample #2-1

FF3-1-SM4
```

```
Key is  <ef 43 59 d8 d5 80 aa 4f 7f 03 6d 6f 04 fc 6a 94 > (hex)
Radix = 10
-----------------------------------

PT is   <6226090102675688>
FF3-1.Encrypt()

X is    [ 6, 2, 2, 6, 0, 9, 0, 1, 0, 2, 6, 7, 5, 6, 8, 8, ]
T is    <d8 e7 92 0a fa 33 0a > (hex)
T is    <1101 1000 1110 0111 1001 0010 0000 1010 1111 1010 0011 0011 0000 1010 >
(bit string)
u is    8
v is    8
A is    [ 6, 2, 2, 6, 0, 9, 0, 1, ]
B is    [ 0, 2, 6, 7, 5, 6, 8, 8, ]
T_L is  D8E79200
T_R is  FA330AA0
Round #0
W is    FA330AA0
P is    [ 250, 51, 10, 160, 0, 0, 0, 0, 0, 0, 0, 0, 5, 72, 206, 212, ]
S is    35EA1E1F FBCD55F2 39AA581C E8E85FFF
Y is    71664692247787811548882045628978847743
m is    8
C is    [ 9, 6, 9, 3, 5, 7, 9, 8, ]
A is    [ 0, 2, 6, 7, 5, 6, 8, 8, ]
B is    [ 9, 6, 9, 3, 5, 7, 9, 8, ]
Round #1
W is    D8E79201
P is    [ 216, 231, 146, 1, 0, 0, 0, 0, 0, 0, 0, 0, 5, 89, 137, 113, ]
S is    FF828820 CFC61FA4 A2DED7C7 705428A6
Y is    33963089852407328604148140249441449795 8
m is    8
C is    [ 8, 7, 5, 5, 5, 1, 3, 0, ]
A is    [ 9, 6, 9, 3, 5, 7, 9, 8, ]
B is    [ 8, 7, 5, 5, 5, 1, 3, 0, ]
Round #2
W is    FA330AA2
P is    [ 250, 51, 10, 162, 0, 0, 0, 0, 0, 0, 0, 0, 0, 48, 38, 122, ]
S is    E2334D5A 9FD4EADE 485FDFCE FE06E423
Y is    30067190311271588891722995443986745859 5
m is    8
C is    [ 4, 6, 5, 2, 1, 2, 7, 5, ]
A is    [ 8, 7, 5, 5, 5, 1, 3, 0, ]
B is    [ 4, 6, 5, 2, 1, 2, 7, 5, ]
```

```
Round #3
W is    D8E79203
P is    [ 216, 231, 146, 3, 0, 0, 0, 0, 0, 0, 0, 0, 3, 104, 254, 148, ]
S is    4D8942B2 68680568 D44CA0D6 B6AAB48D
Y is    10306325311901675700025327928198264948 5
m is    8
C is    [ 3, 6, 0, 5, 0, 8, 5, 8, ]
A is    [ 4, 6, 5, 2, 1, 2, 7, 5, ]
B is    [ 3, 6, 0, 5, 0, 8, 5, 8, ]
Round #4
W is    FA330AA4
P is    [ 250, 51, 10, 164, 0, 0, 0, 0, 0, 0, 0, 0, 5, 29, 72, 7, ]
S is    AACAD627 32191CB4 CDF8725B EA2D23A8
Y is    22702194678991786228646506246063817616 8
m is    8
C is    [ 2, 3, 7, 8, 8, 3, 5, 9, ]
A is    [ 3, 6, 0, 5, 0, 8, 5, 8, ]
B is    [ 2, 3, 7, 8, 8, 3, 5, 9, ]
Round #5
W is    D8E79205
P is    [ 216, 231, 146, 5, 0, 0, 0, 0, 0, 0, 0, 0, 5, 175, 132, 60, ]
S is    DB11EF80 2C0B25AA 65B2687D 96AC6170
Y is    29119405777422255929767463709027731902 4
m is    8
C is    [ 7, 8, 0, 4, 2, 1, 3, 6, ]
A is    [ 2, 3, 7, 8, 8, 3, 5, 9, ]
B is    [ 7, 8, 0, 4, 2, 1, 3, 6, ]
Round #6
W is    FA330AA6
P is    [ 250, 51, 10, 166, 0, 0, 0, 0, 0, 0, 0, 0, 3, 195, 50, 119, ]
S is    AC67E51F 44AEFAAE 88967D43 34C41963
Y is    22916666900056340913157579296025163197 1
m is    8
C is    [ 3, 0, 7, 0, 2, 0, 7, 4, ]
A is    [ 7, 8, 0, 4, 2, 1, 3, 6, ]
B is    [ 3, 0, 7, 0, 2, 0, 7, 4, ]
Round #7
W is    D8E79207
P is    [ 216, 231, 146, 7, 0, 0, 0, 0, 0, 0, 0, 0, 2, 205, 122, 159, ]
S is    32021778 F1070AE8 FAE38C75 3A98C1C8
Y is    66472260460357650369657041547010032072
m is    8
C is    [ 9, 5, 1, 6, 5, 1, 3, 7, ]
A is    [ 3, 0, 7, 0, 2, 0, 7, 4, ]
```

```
B is    [ 9, 5, 1, 6, 5, 1, 3, 7, ]

CT is   <3070207495165137>

------------------------------------------------------------------------

FF3-1.Decrypt()

X is    [ 3, 0, 7, 0, 2, 0, 7, 4, 9, 5, 1, 6, 5, 1, 3, 7, ]
T is    <d8 e7 92 0a fa 33 0a > (hex)
u is    8
v is    8
A is    [ 3, 0, 7, 0, 2, 0, 7, 4, ]
B is    [ 9, 5, 1, 6, 5, 1, 3, 7, ]
T_L is  D8E79200
T_R is  FA330AA0
Round #7
W is    D8E79207
P is    [ 216, 231, 146, 7, 0, 0, 0, 0, 0, 0, 0, 0, 2, 205, 122, 159, ]
S is    32021778 F1070AE8 FAE38C75 3A98C1C8
Y is    66472260460357650369657041547010032072
m is    8
C is    [ 7, 8, 0, 4, 2, 1, 3, 6, ]
B is    [ 3, 0, 7, 0, 2, 0, 7, 4, ]
A is    [ 7, 8, 0, 4, 2, 1, 3, 6, ]
Round #6
W is    FA330AA6
P is    [ 250, 51, 10, 166, 0, 0, 0, 0, 0, 0, 0, 0, 3, 195, 50, 119, ]
S is    AC67E51F 44AEFAAE 88967D43 34C41963
Y is    22916666900056340913157579296025163197l
m is    8
C is    [ 2, 3, 7, 8, 8, 3, 5, 9, ]
B is    [ 7, 8, 0, 4, 2, 1, 3, 6, ]
A is    [ 2, 3, 7, 8, 8, 3, 5, 9, ]
Round #5
W is    D8E79205
P is    [ 216, 231, 146, 5, 0, 0, 0, 0, 0, 0, 0, 0, 5, 175, 132, 60, ]
S is    DB11EF80 2C0B25AA 65B2687D 96AC6170
Y is    29119405777422255929767463709027731902l
m is    8
C is    [ 3, 6, 0, 5, 0, 8, 5, 8, ]
B is    [ 2, 3, 7, 8, 8, 3, 5, 9, ]
A is    [ 3, 6, 0, 5, 0, 8, 5, 8, ]
Round #4
```

```
W is    FA330AA4
P is    [ 250, 51, 10, 164, 0, 0, 0, 0, 0, 0, 0, 0, 5, 29, 72, 7, ]
S is    AACAD627 32191CB4 CDF8725B EA2D23A8
Y is    22702194678991786228646506246063817616 8
m is    8
C is    [ 4, 6, 5, 2, 1, 2, 7, 5, ]
B is    [ 3, 6, 0, 5, 0, 8, 5, 8, ]
A is    [ 4, 6, 5, 2, 1, 2, 7, 5, ]
Round #3
W is    D8E79203
P is    [ 216, 231, 146, 3, 0, 0, 0, 0, 0, 0, 0, 0, 3, 104, 254, 148, ]
S is    4D8942B2 68680568 D44CA0D6 B6AAB48D
Y is    10306325311901675700025327928198264948 5
m is    8
C is    [ 8, 7, 5, 5, 5, 1, 3, 0, ]
B is    [ 4, 6, 5, 2, 1, 2, 7, 5, ]
A is    [ 8, 7, 5, 5, 5, 1, 3, 0, ]
Round #2
W is    FA330AA2
P is    [ 250, 51, 10, 162, 0, 0, 0, 0, 0, 0, 0, 0, 0, 48, 38, 122, ]
S is    E2334D5A 9FD4EADE 485FDFCE FE06E423
Y is    30067190311271588891722995443986745859 5
m is    8
C is    [ 9, 6, 9, 3, 5, 7, 9, 8, ]
B is    [ 8, 7, 5, 5, 5, 1, 3, 0, ]
A is    [ 9, 6, 9, 3, 5, 7, 9, 8, ]
Round #1
W is    D8E79201
P is    [ 216, 231, 146, 1, 0, 0, 0, 0, 0, 0, 0, 0, 5, 89, 137, 113, ]
S is    FF828820 CFC61FA4 A2DED7C7 705428A6
Y is    33963089852407328604148140249441449795 8
m is    8
C is    [ 0, 2, 6, 7, 5, 6, 8, 8, ]
B is    [ 9, 6, 9, 3, 5, 7, 9, 8, ]
A is    [ 0, 2, 6, 7, 5, 6, 8, 8, ]
Round #0
W is    FA330AA0
P is    [ 250, 51, 10, 160, 0, 0, 0, 0, 0, 0, 0, 0, 5, 72, 206, 212, ]
S is    35EA1E1F FBCD55F2 39AA581C E8E85FFF
Y is    71664692247787811548882045628978847743
m is    8
C is    [ 6, 2, 2, 6, 0, 9, 0, 1, ]
B is    [ 0, 2, 6, 7, 5, 6, 8, 8, ]
A is    [ 6, 2, 2, 6, 0, 9, 0, 1, ]
```

PT is <6226090102675688>

2) 基于身份证号 FF3-1 测试向量

Sample #2-2

FF3-1-SM4

Key is <ef 43 59 d8 d5 80 aa 4f 7f 03 6d 6f 04 fc 6a 94 > (hex)
Radix = 10

PT is <110107197203192876>
FF3-1.Encrypt()

X is [1, 1, 0, 1, 0, 7, 1, 9, 7, 2, 0, 3, 1, 9, 2, 8, 7, 6,]
T is <37 38 39 36 70 71 72 > (hex)
T is <0011 0111 0011 1000 0011 1001 0011 0110 0111 0000 0111 0001 0111 0010 >
(bit string)
u is 9
v is 9
A is [1, 1, 0, 1, 0, 7, 1, 9, 7,]
B is [2, 0, 3, 1, 9, 2, 8, 7, 6,]
T_L is 37383930
T_R is 70717260
Round #0
W is 70717260
P is [112, 113, 114, 96, 0, 0, 0, 0, 0, 0, 0, 0, 40, 109, 231, 102,]
S is F07E4391 C356FDAC F28303D8 C2E50F56
Y is 319670318862536949493040000951757639510
m is 9
C is [1, 2, 5, 0, 4, 3, 9, 4, 5,]
A is [2, 0, 3, 1, 9, 2, 8, 7, 6,]
B is [1, 2, 5, 0, 4, 3, 9, 4, 5,]
Round #1
W is 37383931
P is [55, 56, 57, 49, 0, 0, 0, 0, 0, 0, 0, 0, 32, 190, 69, 105,]
S is 6560BBB0 7E12396E C59BB2E0 B241E61E
Y is 134754294866465475109029220322935694878
m is 9
C is [0, 8, 1, 6, 8, 9, 3, 1, 6,]
A is [1, 2, 5, 0, 4, 3, 9, 4, 5,]
B is [0, 8, 1, 6, 8, 9, 3, 1, 6,]
Round #2

```
W is    70717262
P is    [ 112, 113, 114, 98, 0, 0, 0, 0, 0, 0, 0, 0, 36, 152, 175, 132, ]
S is    C6F98438 5E9D00B6 BA597897 B378C6CC
Y is    26448270682731469338137800880973996206O
m is    9
C is    [ 1, 8, 5, 2, 0, 3, 9, 8, 2, ]
A is    [ 0, 8, 1, 6, 8, 9, 3, 1, 6, ]
B is    [ 1, 8, 5, 2, 0, 3, 9, 8, 2, ]
Round #3
W is    37383933
P is    [ 55, 56, 57, 51, 0, 0, 0, 0, 0, 0, 0, 0, 17, 62, 104, 53, ]
S is    FE7E624A 350F2813 76FF3552 A8EAAD28
Y is    338280133888990232579571955712565882152
m is    9
C is    [ 2, 3, 3, 8, 6, 8, 9, 7, 1, ]
A is    [ 1, 8, 5, 2, 0, 3, 9, 8, 2, ]
B is    [ 2, 3, 3, 8, 6, 8, 9, 7, 1, ]
Round #4
W is    70717264
P is    [ 112, 113, 114, 100, 0, 0, 0, 0, 0, 0, 0, 0, 10, 184, 146, 172, ]
S is    19F3C351 CEB24106 E7E1E372 9A75B0A3
Y is    34496389582570143567771327199547076771
m is    9
C is    [ 2, 5, 3, 9, 7, 3, 6, 3, 8, ]
A is    [ 2, 3, 3, 8, 6, 8, 9, 7, 1, ]
B is    [ 2, 5, 3, 9, 7, 3, 6, 3, 8, ]
Round #5
W is    37383935
P is    [ 55, 56, 57, 53, 0, 0, 0, 0, 0, 0, 0, 0, 49, 218, 34, 216, ]
S is    98A76324 BE2B7032 D5774748 DEBCCF5E
Y is    20291177980430180644329550882362129187O
m is    9
C is    [ 2, 0, 2, 0, 6, 1, 1, 0, 8, ]
A is    [ 2, 5, 3, 9, 7, 3, 6, 3, 8, ]
B is    [ 2, 0, 2, 0, 6, 1, 1, 0, 8, ]
Round #6
W is    70717266
P is    [ 112, 113, 114, 102, 0, 0, 0, 0, 0, 0, 0, 0, 47, 192, 188, 10, ]
S is    7DBE110A 441AA01E 65043CF1 CEC67C10
Y is    167140381490558157461917296169645931536
m is    9
C is    [ 8, 8, 8, 0, 1, 3, 2, 8, 4, ]
A is    [ 2, 0, 2, 0, 6, 1, 1, 0, 8, ]
B is    [ 8, 8, 8, 0, 1, 3, 2, 8, 4, ]
```

```
Round #7
W is    37383937
P is    [ 55, 56, 57, 55, 0, 0, 0, 0, 0, 0, 0, 0, 28, 191, 122, 232, ]
S is    EE0539E5 967D2595 D97A9D47 B24A731D
Y is    31638339876827332049344553964305609193 3
m is    9
C is    [ 5, 3, 1, 2, 5, 2, 7, 5, 8, ]
A is    [ 8, 8, 8, 0, 1, 3, 2, 8, 4, ]
B is    [ 5, 3, 1, 2, 5, 2, 7, 5, 8, ]

CT is   <888013284531252758>

------------------------------------------------------------------------

FF3-1.Decrypt()

X is    [ 8, 8, 8, 0, 1, 3, 2, 8, 4, 5, 3, 1, 2, 5, 2, 7, 5, 8, ]
T is    <37 38 39 36 70 71 72 > (hex)
u is    9
v is    9
A is    [ 8, 8, 8, 0, 1, 3, 2, 8, 4, ]
B is    [ 5, 3, 1, 2, 5, 2, 7, 5, 8, ]
T_L is  37383930
T_R is  70717260
Round #7
W is    37383937
P is    [ 55, 56, 57, 55, 0, 0, 0, 0, 0, 0, 0, 0, 28, 191, 122, 232, ]
S is    EE0539E5 967D2595 D97A9D47 B24A731D
Y is    31638339876827332049344553964305609193 3
m is    9
C is    [ 2, 0, 2, 0, 6, 1, 1, 0, 8, ]
B is    [ 8, 8, 8, 0, 1, 3, 2, 8, 4, ]
A is    [ 2, 0, 2, 0, 6, 1, 1, 0, 8, ]
Round #6
W is    70717266
P is    [ 112, 113, 114, 102, 0, 0, 0, 0, 0, 0, 0, 0, 47, 192, 188, 10, ]
S is    7DBE110A 441AA01E 65043CF1 CEC67C10
Y is    16714038149055815746191729616964593 1536
m is    9
C is    [ 2, 5, 3, 9, 7, 3, 6, 3, 8, ]
B is    [ 2, 0, 2, 0, 6, 1, 1, 0, 8, ]
A is    [ 2, 5, 3, 9, 7, 3, 6, 3, 8, ]
Round #5
W is    37383935
```

```
P is    [ 55, 56, 57, 53, 0, 0, 0, 0, 0, 0, 0, 0, 49, 218, 34, 216, ]
S is    98A76324 BE2B7032 D5774748 DEBCCF5E
Y is    202911779804301806443295508823621291870
m is    9
C is    [ 2, 3, 3, 8, 6, 8, 9, 7, 1, ]
B is    [ 2, 5, 3, 9, 7, 3, 6, 3, 8, ]
A is    [ 2, 3, 3, 8, 6, 8, 9, 7, 1, ]
Round #4
W is    70717264
P is    [ 112, 113, 114, 100, 0, 0, 0, 0, 0, 0, 0, 0, 10, 184, 146, 172, ]
S is    19F3C351 CEB24106 E7E1E372 9A75B0A3
Y is    34496389582570143567771327199547076771
m is    9
C is    [ 1, 8, 5, 2, 0, 3, 9, 8, 2, ]
B is    [ 2, 3, 3, 8, 6, 8, 9, 7, 1, ]
A is    [ 1, 8, 5, 2, 0, 3, 9, 8, 2, ]
Round #3
W is    37383933
P is    [ 55, 56, 57, 51, 0, 0, 0, 0, 0, 0, 0, 0, 17, 62, 104, 53, ]
S is    FE7E624A 350F2813 76FF3552 A8EAAD28
Y is    33828013388899023257957195571256588215 2
m is    9
C is    [ 0, 8, 1, 6, 8, 9, 3, 1, 6, ]
B is    [ 1, 8, 5, 2, 0, 3, 9, 8, 2, ]
A is    [ 0, 8, 1, 6, 8, 9, 3, 1, 6, ]
Round #2
W is    70717262
P is    [ 112, 113, 114, 98, 0, 0, 0, 0, 0, 0, 0, 0, 36, 152, 175, 132, ]
S is    C6F98438 5E9D00B6 BA597897 B378C6CC
Y is    26448270682731469338137800880973996 2060
m is    9
C is    [ 1, 2, 5, 0, 4, 3, 9, 4, 5, ]
B is    [ 0, 8, 1, 6, 8, 9, 3, 1, 6, ]
A is    [ 1, 2, 5, 0, 4, 3, 9, 4, 5, ]
Round #1
W is    37383931
P is    [ 55, 56, 57, 49, 0, 0, 0, 0, 0, 0, 0, 0, 32, 190, 69, 105, ]
S is    6560BBB0 7E12396E C59BB2E0 B241E61E
Y is    13475429486646547510902922032293569 4878
m is    9
C is    [ 2, 0, 3, 1, 9, 2, 8, 7, 6, ]
B is    [ 1, 2, 5, 0, 4, 3, 9, 4, 5, ]
A is    [ 2, 0, 3, 1, 9, 2, 8, 7, 6, ]
Round #0
```

```
W is    70717260
P is    [ 112, 113, 114, 96, 0, 0, 0, 0, 0, 0, 0, 0, 40, 109, 231, 102, ]
S is    F07E4391 C356FDAC F28303D8 C2E50F56
Y is    319670318862536949493040000951757639510
m is    9
C is    [ 1, 1, 0, 1, 0, 7, 1, 9, 7, ]
B is    [ 2, 0, 3, 1, 9, 2, 8, 7, 6, ]
A is    [ 1, 1, 0, 1, 0, 7, 1, 9, 7, ]

PT is   <1101071972031928 76>
```

3) 基于手机号 FF3-1 测试向量

```
Sample #2-3

FF3-1-SM4

Key is  <ef 43 59 d8 d5 80 aa 4f 7f 03 6d 6f 04 fc 6a 94 > (hex)
Radix = 10
-----------------------------------

PT is   <13687260594>
FF3-1.Encrypt()

X is    [ 1, 3, 6, 8, 7, 2, 6, 0, 5, 9, 4, ]
T is    <39 38 37 36 35 34 33 > (hex)
T is    <0011 1001 0011 1000 0011 0111 0011 0110 0011 0101 0011 0100 0011 0011 >
(bit string)
u is    6
v is    5
A is    [ 1, 3, 6, 8, 7, 2, ]
B is    [ 6, 0, 5, 9, 4, ]
T_L is  39383730
T_R is  35343360
Round #0
W is    35343360
P is    [ 53, 52, 51, 96, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 193, 98, ]
S is    18735433 82540EED AFBA670A 50BBBA47
Y is    325002938409471526380633453898390390 47
m is    6
C is    [ 8, 7, 6, 7, 1, 3, ]
A is    [ 6, 0, 5, 9, 4, ]
B is    [ 8, 7, 6, 7, 1, 3, ]
Round #1
W is    39383731
```

```
P is    [ 57, 56, 55, 49, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 216, 238, ]
S is    35735E39 1C4E5F50 AF3C14E6 33B7AD01
Y is    710481089866003796133798670935077470 73
m is    5
C is    [ 9, 7, 5, 6, 9, ]
A is    [ 8, 7, 6, 7, 1, 3, ]
B is    [ 9, 7, 5, 6, 9, ]
Round #2
W is    35343362
P is    [ 53, 52, 51, 98, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 121, 67, ]
S is    3A39E05F 377A8015 62441C4D 9CA905BB
Y is    773957354800580319800732953934535735 63
m is    6
C is    [ 1, 4, 2, 1, 9, 8, ]
A is    [ 9, 7, 5, 6, 9, ]
B is    [ 1, 4, 2, 1, 9, 8, ]
Round #3
W is    39383733
P is    [ 57, 56, 55, 51, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13, 153, 105, ]
S is    EA9B824D 746D246A 8062F866 7A226351
Y is    311846799876592406880835694258330690385
m is    5
C is    [ 4, 6, 9, 6, 8, ]
A is    [ 1, 4, 2, 1, 9, 8, ]
B is    [ 4, 6, 9, 6, 8, ]
Round #4
W is    35343364
P is    [ 53, 52, 51, 100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 83, 180, ]
S is    4C72C709 6C8CE92B 58D2435B 77C6D543
Y is    10161728646768589778758728839189697875 5
m is    6
C is    [ 6, 9, 9, 9, 6, 8, ]
A is    [ 4, 6, 9, 6, 8, ]
B is    [ 6, 9, 9, 9, 6, 8, ]
Round #5
W is    39383735
P is    [ 57, 56, 55, 53, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13, 70, 108, ]
S is    BD3750EC 795AD233 B05D2149 0F1D9776
Y is    25151130885874064316100807019485079743 0
m is    5
C is    [ 4, 9, 3, 4, 8, ]
A is    [ 6, 9, 9, 9, 6, 8, ]
B is    [ 4, 9, 3, 4, 8, ]
Round #6
```

```
W is    35343366
P is    [ 53, 52, 51, 102, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 73, 170, ]
S is    A8E246CA 0B559742 BBF471FA 0BE220A4
Y is    22448519815816362930128298521621496643 6
m is    6
C is    [ 2, 3, 4, 6, 3, 8, ]
A is    [ 4, 9, 3, 4, 8, ]
B is    [ 2, 3, 4, 6, 3, 8, ]
Round #7
W is    39383737
P is    [ 57, 56, 55, 55, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12, 195, 80, ]
S is    21A2FF3C 2D9A5438 87B2985D 0BDCDB2D
Y is    44710852734236945653227390986125892397
m is    5
C is    [ 1, 9, 7, 6, 7, ]
A is    [ 2, 3, 4, 6, 3, 8, ]
B is    [ 1, 9, 7, 6, 7, ]

CT is   <23463819767>

------------------------------------------------------------------------

FF3-1.Decrypt()

X is    [ 2, 3, 4, 6, 3, 8, 1, 9, 7, 6, 7, ]
T is    <39 38 37 36 35 34 33 > (hex)
u is    6
v is    5
A is    [ 2, 3, 4, 6, 3, 8, ]
B is    [ 1, 9, 7, 6, 7, ]
T_L is  39383730
T_R is  35343360
Round #7
W is    39383737
P is    [ 57, 56, 55, 55, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12, 195, 80, ]
S is    21A2FF3C 2D9A5438 87B2985D 0BDCDB2D
Y is    44710852734236945653227390986125892397
m is    5
C is    [ 4, 9, 3, 4, 8, ]
B is    [ 2, 3, 4, 6, 3, 8, ]
A is    [ 4, 9, 3, 4, 8, ]
Round #6
W is    35343366
P is    [ 53, 52, 51, 102, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 73, 170, ]
```

```
S is    A8E246CA 0B559742 BBF471FA 0BE220A4
Y is    22448519815816362930128298521621496643 6
m is    6
C is    [ 6, 9, 9, 9, 6, 8, ]
B is    [ 4, 9, 3, 4, 8, ]
A is    [ 6, 9, 9, 9, 6, 8, ]
Round #5
W is    39383735
P is    [ 57, 56, 55, 53, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13, 70, 108, ]
S is    BD3750EC 795AD233 B05D2149 0F1D9776
Y is    25151130885874064316100807019485079743 0
m is    5
C is    [ 4, 6, 9, 6, 8, ]
B is    [ 6, 9, 9, 9, 6, 8, ]
A is    [ 4, 6, 9, 6, 8, ]
Round #4
W is    35343364
P is    [ 53, 52, 51, 100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 83, 180, ]
S is    4C72C709 6C8CE92B 58D2435B 77C6D543
Y is    10161728646768589778758728839189697875 5
m is    6
C is    [ 1, 4, 2, 1, 9, 8, ]
B is    [ 4, 6, 9, 6, 8, ]
A is    [ 1, 4, 2, 1, 9, 8, ]
Round #3
W is    39383733
P is    [ 57, 56, 55, 51, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13, 153, 105, ]
S is    EA9B824D 746D246A 8062F866 7A226351
Y is    31184679987659240688083569425833069038 5
m is    5
C is    [ 9, 7, 5, 6, 9, ]
B is    [ 1, 4, 2, 1, 9, 8, ]
A is    [ 9, 7, 5, 6, 9, ]
Round #2
W is    35343362
P is    [ 53, 52, 51, 98, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 121, 67, ]
S is    3A39E05F 377A8015 62441C4D 9CA905BB
Y is    77395735480058031980073295393453573563
m is    6
C is    [ 8, 7, 6, 7, 1, 3, ]
B is    [ 9, 7, 5, 6, 9, ]
A is    [ 8, 7, 6, 7, 1, 3, ]
Round #1
W is    39383731
```

```
P is    [ 57, 56, 55, 49, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 216, 238, ]
S is    35735E39 1C4E5F50 AF3C14E6 33B7AD01
Y is    710481089866003796133798670935077470 73
m is    5
C is    [ 6, 0, 5, 9, 4, ]
B is    [ 8, 7, 6, 7, 1, 3, ]
A is    [ 6, 0, 5, 9, 4, ]
Round #0
W is    35343360
P is    [ 53, 52, 51, 96, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 193, 98, ]
S is    18735433 82540EED AFBA670A 50BBBA47
Y is    325002938409471526380633453898390390 47
m is    6
C is    [ 1, 3, 6, 8, 7, 2, ]
B is    [ 6, 0, 5, 9, 4, ]
A is    [ 1, 3, 6, 8, 7, 2, ]

PT is   <13687260594>
```

6.4.3 TE-FPE 示例

1)基于银行卡号 TE-FPE 的测试向量

```
Sample #3-1

Key is <ef 43 59 d8 d5 80 aa 4f 7f 03 6d 6f 04 fc 6a 94>
Radix = 10

-----------------------------------

PT is   <6226090102675688>
TE-FPE.Encrypt()

X is    [ 6, 2, 2, 6, 0, 9, 0, 1, 0, 2, 6, 7, 5, 6, 8, 8, ]
T is    <d8 e7 92 0a fa 33 0a > (hex)

u is <8> v is <8>
h is <53>
r is <8>
D is FF00090140D8E7920AFA330A00100808
Z is D9592ABF65D0584CBC31F358445BC20A
T_L is D9592ABF65D05800
T_R is BC31F358445BC200
Round #0
W is BC31F358445BC200
P is [188, 49, 243, 88, 68, 91, 194, 0, 0, 0, 0, 0, 5, 72, 206, 212]
```

S is 8289ED64902CBF33CEC316B8948CF06B
Y is 173515799020170588812945517102270050411
m is 8
c: 80956637
Round #1
W is D9592ABF65D05800
P is [217, 89, 42, 191, 101, 208, 88, 1, 0, 0, 0, 0, 4, 211, 76, 221]
S is F5751D8B95257456ED5D1B05F530FD34
Y is 32626895694850460519189803422164772382 8
m is 8
c: 36381448
Round #2
W is BC31F358445BC200
P is [188, 49, 243, 88, 68, 91, 194, 2, 0, 0, 0, 0, 2, 43, 35, 8]
S is 44B4C9E44C80BB14465EA0C1DC5D0ED2
Y is 913262119999384222764892970040959341 62
m is 8
c: 76890799
Round #3
W is D9592ABF65D05800
P is [217, 89, 42, 191, 101, 208, 88, 3, 0, 0, 0, 0, 4, 149, 66, 175]
S is C5E0FC7F8090B808B354999BDE07D945
Y is 263026110934926023534444129268261181765
m is 8
c: 97563213
Round #4
W is BC31F358445BC200
P is [188, 49, 243, 88, 68, 91, 194, 4, 0, 0, 0, 0, 5, 208, 178, 77]
S is FBAC715C89C62893ABA4E7CF1ACD21DC
Y is 33453160124559704367854671211079068 1052
m is 8
c: 67571851
Round #5
W is D9592ABF65D05800
P is [217, 89, 42, 191, 101, 208, 88, 5, 0, 0, 0, 0, 4, 7, 16, 139]
S is 34F94B701FDF790889DCFBE388FD2DD1
Y is 70414267762729470269477898327244484049
m is 8
c: 42047262
Round #6
W is BC31F358445BC200
P is [188, 49, 243, 88, 68, 91, 194, 6, 0, 0, 0, 0, 2, 129, 151, 30]
S is 6D4DBF4286EB2969FEBEC016148F9EB8
Y is 14528953760971131974036899567921264 1976

```
m is 8
c: 80213827
Round #7
W is D9592ABF65D05800
P is [217, 89, 42, 191, 101, 208, 88, 7, 0, 0, 0, 0, 4, 199, 247, 67]
S is F8A23A2A188394C345FCF8BC922F4ACB
Y is 330490874760668327982432395547953285835
m is 8
c: 95333097

CT is <7 2 8 3 1 2 0 8 7 9 0 3 3 3 5 9>
--------------------------------------------------------------------------
TE-FPE.Decrypt()
u is <8> v is <8>
h is <53>
r is <8>
D is FF00090140D8E7920AFA330A00100808
Z is D9592ABF65D0584CBC31F358445BC20A
T_L is D9592ABF65D05800
T_R is BC31F358445BC200
Round #7
W is D9592ABF65D05800
P is [217, 89, 42, 191, 101, 208, 88, 7, 0, 0, 0, 0, 4, 199, 247, 67]
S is F8A23A2A188394C345FCF8BC922F4ACB
Y is 330490874760668327982432395547953285835
m is 8
c: 42047262
Round #6
W is BC31F358445BC200
P is [188, 49, 243, 88, 68, 91, 194, 6, 0, 0, 0, 0, 2, 129, 151, 30]
S is 6D4DBF4286EB2969FEBEC016148F9EB8
Y is 145289537609711319740368995679212641976
m is 8
c: 67571851
Round #5
W is D9592ABF65D05800
P is [217, 89, 42, 191, 101, 208, 88, 5, 0, 0, 0, 0, 4, 7, 16, 139]
S is 34F94B701FDF790889DCFBE388FD2DD1
Y is 704142677627294702694778983272444484049
m is 8
c: 97563213
Round #4
W is BC31F358445BC200
P is [188, 49, 243, 88, 68, 91, 194, 4, 0, 0, 0, 0, 5, 208, 178, 77]
```

S is FBAC715C89C62893ABA4E7CF1ACD21DC
Y is 3345316012455970436785467121107906810 52
m is 8
c: 76890799
Round #3
W is D9592ABF65D05800
P is [217, 89, 42, 191, 101, 208, 88, 3, 0, 0, 0, 0, 4, 149, 66, 175]
S is C5E0FC7F8090B808B354999BDE07D945
Y is 263026110934926023534444129268261181765
m is 8
c: 36381448
Round #2
W is BC31F358445BC200
P is [188, 49, 243, 88, 68, 91, 194, 2, 0, 0, 0, 0, 2, 43, 35, 8]
S is 44B4C9E44C80BB14465EA0C1DC5D0ED2
Y is 913262119999384222764892970040959341 62
m is 8
c: 80956637
Round #1
W is D9592ABF65D05800
P is [217, 89, 42, 191, 101, 208, 88, 1, 0, 0, 0, 0, 4, 211, 76, 221]
S is F5751D8B95257456ED5D1B05F530FD34
Y is 326268956948504605191898034221647723828
m is 8
c: 88657620
Round #0
W is BC31F358445BC200
P is [188, 49, 243, 88, 68, 91, 194, 0, 0, 0, 0, 0, 5, 72, 206, 212]
S is 8289ED64902CBF33CEC316B8948CF06B
Y is 173515799020170588812945517102270050411
m is 8
c: 10906226
PT is <6 2 2 6 0 9 0 1 0 2 6 7 5 6 8 8>

2) 基于手机号 TE-FPE 的测试向量

Sample #3-2

Key is <ef 43 59 d8 d5 80 aa 4f 7f 03 6d 6f 04 fc 6a 94>
Radix = 10

PT is <13687260594>
TE-FPE.Encrypt()

```
X is  [1, 3, 6, 8, 7, 2, 6, 0, 5, 9, 4]
T is  <d8 e7 92 0a fa 33 0a > (hex)

u is <5> v is <6>
h is <36>
r is <8>
D is FF00090140D8E7920AFA330A00100808
Z is D9592ABF65D0584CBC31F358445BC20A
T_L is 3ABAF096AE071900
T_R is 2AC919905ABAC900
Round #0
W is 2AC919905ABAC900
P is [42, 201, 25, 144, 90, 186, 201, 0, 0, 0, 0, 0, 0, 7, 141, 214]
S is E0D4D7998F24D5A9F190A8470E7C8199
Y is 29885221087410507551221171594804674600 9
m is 5
c: 24640
Round #1
W is 3ABAF096AE071900
P is [58, 186, 240, 150, 174, 7, 25, 1, 0, 0, 0, 0, 0, 0, 96, 64]
S is 71FCE133EC10E71B82AE4EC9D77615D5
Y is 151515789987902276558933941952657626581
m is 6
c: 121643
Round #2
W is 2AC919905ABAC900
P is [42, 201, 25, 144, 90, 186, 201, 2, 0, 0, 0, 0, 0, 1, 219, 43]
S is F92C9A079F937C9F64ECAA40916F5DB6
Y is 331209356107282101896624725544135515574
m is 5
c: 40214
Round #3
W is 3ABAF096AE071900
P is [58, 186, 240, 150, 174, 7, 25, 3, 0, 0, 0, 0, 0, 0, 157, 22]
S is FDEA6B94FDB18D2524FA9AADE379BDC3
Y is 337511862420590862243336967892057570755
m is 6
c: 692398
Round #4
W is 2AC919905ABAC900
P is [42, 201, 25, 144, 90, 186, 201, 4, 0, 0, 0, 0, 0, 10, 144, 174]
S is F6D62DD6F6123750D348CE06A7E15916
Y is 328102168230230034526504064715130493206
```

```
m is 5
c: 33420
Round #5
W is 3ABAF096AE071900
P is [58, 186, 240, 150, 174, 7, 25, 5, 0, 0, 0, 0, 0, 0, 130, 140]
S is 9B95A29813DA6F10C8E04B38ACB46101
Y is 20680728937776442886311283305488085017 7
m is 6
c: 542575
Round #6
W is 2AC919905ABAC900
P is [42, 201, 25, 144, 90, 186, 201, 6, 0, 0, 0, 0, 0, 8, 71, 111]
S is E3C42E1684EB588F87356173DB752295
Y is 302753380002446609734915791111780115093
m is 5
c: 48513
Round #7
W is 3ABAF096AE071900
P is [58, 186, 240, 150, 174, 7, 25, 7, 0, 0, 0, 0, 0, 0, 189, 129]
S is A5F1126D3F3D13C1C36DCD5A0B24C242
Y is 22057433658623198566549259316340855661 0
m is 6
c: 99185
CT is <3 1 5 8 4 5 8 1 9 9 0>
------------------------------------------------------------------------
TE-FPE.Decrypt()
u is <5> v is <6>
h is <36>
r is <8>
D is FF00090140D8E7920AFA330A00100808
Z is D9592ABF65D0584CBC31F358445BC20A
T_L is 3ABAF096AE071900
T_R is 2AC919905ABAC900
Round #7
W is 3ABAF096AE071900
P is [58, 186, 240, 150, 174, 7, 25, 7, 0, 0, 0, 0, 0, 0, 189, 129]
S is A5F1126D3F3D13C1C36DCD5A0B24C242
Y is 22057433658623198566549259316340855661 0
m is 6
c: 542575
Round #6
W is 2AC919905ABAC900
P is [42, 201, 25, 144, 90, 186, 201, 6, 0, 0, 0, 0, 0, 8, 71, 111]
S is E3C42E1684EB588F87356173DB752295
```

```
Y is 3027533800024466097349157911117801150 93
m is 5
c: 33420
Round #5
W is 3ABAF096AE071900
P is [58, 186, 240, 150, 174, 7, 25, 5, 0, 0, 0, 0, 0, 0, 130, 140]
S is 9B95A29813DA6F10C8E04B38ACB46101
Y is 20680728937776442886311283305488085 0177
m is 6
c: 692398
Round #4
W is 2AC919905ABAC900
P is [42, 201, 25, 144, 90, 186, 201, 4, 0, 0, 0, 0, 0, 10, 144, 174]
S is F6D62DD6F6123750D348CE06A7E15916
Y is 32810216823023003452650406471513049 3206
m is 5
c: 40214
Round #3
W is 3ABAF096AE071900
P is [58, 186, 240, 150, 174, 7, 25, 3, 0, 0, 0, 0, 0, 0, 157, 22]
S is FDEA6B94FDB18D2524FA9AADE379BDC3
Y is 337511862420590862243336967892057570755
m is 6
c: 121643
Round #2
W is 2AC919905ABAC900
P is [42, 201, 25, 144, 90, 186, 201, 2, 0, 0, 0, 0, 0, 1, 219, 43]
S is F92C9A079F937C9F64ECAA40916F5DB6
Y is 331209356107282101896624725544135515574
m is 5
c: 24640
Round #1
W is 3ABAF096AE071900
P is [58, 186, 240, 150, 174, 7, 25, 1, 0, 0, 0, 0, 0, 0, 96, 64]
S is 71FCE133EC10E71B82AE4EC9D77615D5
Y is 151515789987902276558933941952657626581
m is 6
c: 495062
Round #0
W is 2AC919905ABAC900
P is [42, 201, 25, 144, 90, 186, 201, 0, 0, 0, 0, 0, 0, 7, 141, 214]
S is E0D4D7998F24D5A9F190A8470E7C8199
Y is 29885221087410507551221171594804674 6009
m is 5
```

c: 78631
PT is <1 3 6 8 7 2 6 0 5 9 4>

参考文献

[1] 刘哲理，贾春福，李经纬.保留格式加密技术研究.软件学报，2012,23(1):152~170

[2] Spies T. Format Preserving encryption. Unpublished Voltage White Paper. 2008.

[3] Spies T. Feistel finite set encryption mode. 2008.

[4] Bellare M, Ristenpart T, Rogaway P, Stegers T. Format-Preserving encryption. In: Jacobsn MJ, eds. Proc. of the Selected Areas in Cryptography (SAC 2009). LNCS 5867, Calgary: Springer-Verlag, 2009. 295-312.

[5] Bellare M, Rogaway P, Spies T. The FFX mode of operation for format-preserving encryption. 2010.

[6] Liu ZL, Jia CF, Li JW, Cheng XC. Format-Preserving encryption for datetime. In: Chen W, ed. Proc. of the Intelligent Computing and Intelligent Systems 2010,Vol.2.Xiamen:IEEE Press,2010.201-205.

[7] Stütz T, Uhl A. Efficient format-compliant encryption of regular languages: Block-Based Cycle-Walking. In: De Decker B, ed. Proc. of the 11th IFIP TC 6/TC 11 Int'l Conf. LNCS 6109, Linz: Springer-Verlag, 2010. 81-92.

[8] JR/T 0008—2000 银行卡发卡行标识代码及卡号[S].

[9] JR/T 0098.8—2012 中国金融移动支付 检测规范 第8部分:个人信息保护[S].

[10] 银发[2011]17号 人民银行关于银行业金融机构做好个人金融信息保护工作的通知

[11] GB 11643-1999 公民身份号码[S].

[12] 主席令第五十一号《中华人民共和国居民身份证法》
http://www.gov.cn/zhengce/2011-10/29/content_2602263.htm

[13] 工业和信息化部令 第24号《电信和互联网用户个人信息保护规定》
http://www.miit.gov.cn/n1146285/n1146352/n3054355/n3057724/n3057729/c4700145/content.html

[14] GB/T 35273-2017 信息安全技术 个人信息安全规范.

[15] 中华人民共和国网络安全法
http://www.cac.gov.cn/2016-11/07/c_1119867116.htm

[16] 中华人民共和国国务院令 第697号《快递暂行条例》
http://www.gov.cn/zhengce/content/2018-03/27/content_5277801.htm

[17] Black J, Rogaway P. Ciphers with arbitrary finite domains. In: Preneel B, ed. Proc. of the Topics in Cryptology—CT-RSA 2002. LNCS 2271, San Jose: Springer-Verlag, 2002. 114 130.

[18] Liskov M, Rivest RL, Wagner D. Tweakable block ciphers. In: Advances in Cryptology—CRYPTO 2002. LNCS 2442, Santa Barbara: Springer-Verlag, 2002. 31-46.

[19] M. Dworkin. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. NIST Special Publication 800-38G, Mar. 2016. http://dx.doi.org/10.6028/NIST.SP.800-38G.

[20] M. Bellare, V. T. Hoang, and S. Tessaro. Message-recovery attacks on feistel-based format preserving encryption. In ACM CCS 16, pages 444{455. ACM Press, 2016.

[21] F. B. Durak and S. Vaudenay. Breaking and repairing the FF3 format preserving encryption over small domain. In CRYPTO 2017, pages 679{707. Springer, 2017.

[22] Hoang V.T., Tessaro S., Trieu N. (2018) The Curse of Small Domains: New Attacks on Format-Preserving Encryption. In: Shacham H., Boldyreva A. (eds) Advances in Cryptology – CRYPTO 2018. CRYPTO 2018. Lecture Notes in Computer Science, vol 10991. Springer, Cham.

[23] Recent Cryptanalysis of FF3. April 12, 2017. https://csrc.nist.gov/News/2017/Recent-Cryptanalysis-of-FF3

[24] Methods for Format-Preserving Encryption: NIST Requests Public Comments on Draft Special Publication 800-38G Revision 1. February 28, 2019. https://csrc.nist.gov/News/2019/nist-requests-comments-on-draft-sp-800-38g-rev-1

[25] Viet Tung Hoang and David Miller and Ni Trieu. Attacks Only Get Better: How to Break FF3 on Large Domains. EUROCRYPT-2019 https://eprint.iacr.org/2019/244

[26] 李敏 保留格式加密技术应用研究 南开大学 博士学位论文 2012

[27] GB/T 17964:2008 《信息安全技术 分组密码算法的工作模式》

[28] ISO/IEC 10116:2017 Information technology-Security techniques-Modes of operation for an n-bit blokc cipher

[29] 陈佳,彭长根等,SM4-FPE:基于 SM4 的数字型数据保留格式加密算法,Journal of Chinese Computer Systems[J],2009

[30] 王鹏. 多类型数据保留格式加密技术的研究与实现[D].北京邮电大学,2017.

[31] Ohad Amon1,Orr Dunkelman,Nathan Keller,Eyal Ronen,and AdiShamir. Three Third Generation Attacks on the Format Preserving Encryption Scheme FF3. EUROCRYPT-2021

[32] Tim Beyne. Linear Cryptanalysis of FF3-1 and FEA. CRYPTO 2021.

[33] 深圳奥联信息安全技术有限公司. 格式保留加密方案 TE-FPE 及在地理位置数据保护中的应用方法. 技术报告，2022