基于国密算法的电网数据安全传输方案与实现

陈风帆[1]
CHEN Fengfan

摘 要 随着数字技术与能源领域的深度融合，电网数据安全传输已成为保障电网稳定运行的关键环节。文章提出一种基于国密算法的电网数据安全传输方案。首先，分析电网数据特征与传输场景；其次，设计包含数据转发服务程序、数据发送程序、数据接收程序及系统配置与密钥管理程序的分布式安全架构；最后，基于 C++ 与 OpenSSL 完成方案工程化实现，并通过安全性与性能测试验证。

关键词 国密算法；电网数据传输；国产化；网络安全；信息安全

doi：10.3969/j.issn.1672-9528.2026.06.033

0 引言

能源是国民经济的命脉，电网作为能源输送与分配的载体，其稳定运行关乎国家战略安全与民生保障。随着数字技术与能源领域深度融合，在新一代电力交易与电网调度中，电网模型数据、设备量测数据、出清数据、申报数据等，在终端、边缘节点、云端之间高频交互，这类数据的传输安全已成为关键环节。

当前我国很多信息基础设施领域仍存在“核心技术依赖进口”问题，电网数据传输安全领域亦是如此。部分电网企业曾在数据加密，身份认证等安全环节，广泛采用国际通用加密算法，这类方案虽满足了基础安全需求，但随着国际形势变化与网络安全威胁升级，潜在风险日益凸显：一方面，国外算法受应用场景限制，存在“技术卡脖子”隐患，一旦遭遇技术封锁，将直接威胁电网数据传输安全；另一方面，国外算法也难以适配我国电网特有业务场景与数据安全本土化法规要求。

在此背景下，“国产化替代”已成为保障国家关键信息基础设施安全的战略举措，相关法律法规与政策明确支持关键核心技术自主可控，优先选用安全可信产品与服务。对于电网领域而言，需要从“根技术”层面摆脱外部依赖。国密算法作为国产化安全体系的核心支撑，涵盖 SM2、SM3、SM4[1-5]。既能满足电网数据传输需求，也能为数据传输安全构建可靠屏障。

本文聚焦电网数据传输安全的国产化需求，围绕国密算法的工程化应用展开研究。首先，分析电网数据传输的业务场景与安全需求，明确国密算法在不同传输环节的应用场景与适配策略；其次，设计基于国密算法的电网数据传输安全架构，涵盖对称密钥协商模块和数据传输模块；最后，通过搭建实验平台进行方案验证。

1. 中国电力科学研究院有限公司南京分院 江苏南京 210003

1 电网数据与网络环境分析

1.1 电网核心数据特征

JSON（javascript object notation）数据是电网数据传输与交互中广泛采用的轻量级数据交换格式，因其简单高效的结构，在新一代电力交易与电网调度场景中广泛使用。采用键值对的结构化形式，可灵活封装各类电网数据。

CIM/E（common information model）数据是电网模型数据标准文件格式，遵循电力调度自动化系统相关技术规范，为电网拓扑结构、设备参数、逻辑关系等的规范化存储统一标准。是电网各个系统及平台的核心数据文件[6]。

QS 数据是电网在线仿真计算与继电保护定值校核的核心数据文件，是新一代电网调度技术支持平台的关键数据载体，主要用于记录电网实时运行状态与设备参数，为电网动态运行分析提供时序化数据支撑。

1.2 电网安全分区与数据传输场景

电网安全分区遵循“横向隔离，纵向贯通”的基本原则，其中生产控制大区（Ⅲ区）与信息管理大区（Ⅳ区）的边界防护是保障电网数据安全的关键环节。两者通过防火墙、隔离装置、网闸等专用安全设备构建而成的隔离屏障，仅开放基于业务需求的白名单访问规定协议的问权限，允许经认证的指定机器通过规范通道跨区访问。同时，Ⅲ区和Ⅳ区各所属单位均建立独立的防护体系，对单位内部核心业务系统与数据资源进行有效访问与控制。

Ⅲ区与Ⅳ区各自内部的数据传输，需在安全分区防护规则框架下进行。传输以各电网公司自主设定的防火墙策略为基础，通过严格的白名单进行访问控制，仅允许符合业务需求的特定数据或协议。传输过程中，依托各公司部署的边界安全设备，对数据来源、类型、流向进行全程校验，结合加

密传输、身份认证等机制防范窃取、篡改风险，在满足内部业务协同需求的同时，也能兼顾各个大区间跨数据交互的安全性与合规性[7]。

2 电网安全传输的国密技术

2.1 对称加密

SM4 是我国自主研发的对称分组加密算法，是国密算法体系中实现电网数据安全传输的核心技术之一。采用 128 位分组长度与密钥长度设计，通过 32 轮非线性迭代运算完成数据加解密。在电网场景中，SM4 凭借对称加密算法“加密效率高、适配大文件”的特性，可对 CIM/E、QS、JSON 格式的电网数据文件进行高效加密处理，既能保障数据在传输过程中数据的保密性，又能适配电网数据的传输需求。

2.2 非对称加密

SM2 是我国自主设计的非对称椭圆曲线加密算法，它基于椭圆曲线离散对数问题构建加密、解密、签名与验签机制，密钥长度为 256 位，相较于传统 RSA 算法，在相同安全强度下密钥长度更短、运算效率更高。在电网场景中，通过 SM2 加密 SM4 的密钥，实现对称密钥在电网不同主机之间的安全传输，发挥了 SM2 抗中间人攻击，身份可认证的特点。

2.3 签名及验签机制

SM3 与 SM2 的组合机制是国密算法在电网数据安全传输中保障完整性与不可抵赖性的核心技术。SM3 是我国自主设计的哈希算法，能将任意长度的电网数据，如 CIM/E、QS、JSON 映射为 256 位的固定长度哈希值。在电网数据传输场景中，该机制的应用逻辑为：对需传输的电网数据，先通过 SM3 生成数据的哈希摘要，再由数据发送方使用 SM2 私钥对该摘要进行签名；接收方通过发送方的 SM2 公钥验证签名，同时对接收数据重新计算 SM3 哈希值并与签名中的摘要比对，数据一致则验签成功。

3 国密算法下电网数据传输方案

3.1 架构设计

本方案基于 C++ 语言开发，集成 OpenSSL 3.5.2 支持库，构建了包含 4 个核心程序的安全传输架构[8-10]。实现电网数据，如 CIM/E、QS、JSON 数据的加密传输与全链路安全管控，具体组件及功能如下：

CentralServer（数据转发服务程序）：为数据传输的核心枢纽，负责接收 ClientSend 发送的加密数据，解密后按指定的 ClientReceive 对应的国密加密规则重新加密，再定向转发至目标接收端。该程序启动时会读取系统配置库中由 SystemController 维护的节点信息，包括 ClientSend 与 ClientReceive 的公私钥，IP 地址关联关系等。并且对连接请求进行合法性校验，仅允许已授权的节点接入。

ClientSend（数据发送程序）：由数据发送方部署，支持用户指定待传输的电网数据及 ClientReceive，通过国密算法，对数据进行安全处理后，将数据发送至 CentralServer。

ClientReceive（数据接收程序）：部署于数据接收端，负责接收 CentralServer 转发的加密数据，通过国密算法解密后，验证数据完整性与发送方身份，确保接收数据的安全性与可靠性。

SystemController（系统配置与密钥管理程序）：承担全系统的初始化配置与密钥生命周期管理功能，包括生成 CentralServer、ClientSend、ClientReceive 各自的 SM2 公私钥，关联节点 IP 地址与身份信息，设置密钥有效生存时间等，并将所有配置数据存储于系统配置库中，为整个传输架构提供统一的身份认证与密钥支撑。

为直观呈现本方案的分布式部署与数据传输逻辑，设计了如图 1 所示的技术架构图。

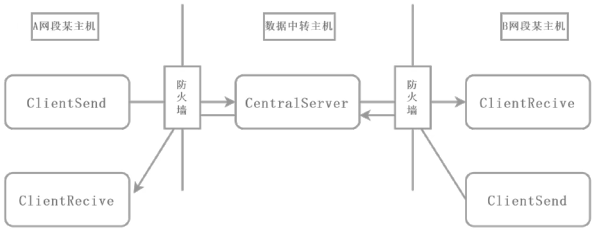

图 1 技术架构图

3.2 数据安全传输流程

3.2.1 SM4 密钥协商机制

先建立 TCP 连接以保障链路可靠性，再通过 IP 合法性验证确认通信方身份，最终通过 SM2 加密 SM4 密钥。具体流程为：ClientSend 与 ClientReceive 分别向 CentralServer 发起 TCP 连接请求。随后 CentralServer 对发起端 IP 进行合法性校验，验证 IP 是否在预设信任白名单内，校验通过后才允许后续密钥传输操作。

验证通过后，ClientSend 与 ClientReceive 各自随机生成 SM4 密钥，该密钥用于后续数据加解密。生成密钥后，使用自身 SM2 私钥对该 SM4 密钥进行签名；随后使用 CentralServer 的 SM2 公钥对 SM4 密钥进行加密。加密与签名完成后，发起端将加密后的 SM4 密钥与签名数据进行序列化处理，通过 TCP Socket 发送至 CentralServer。

CentralServer 接收数据后，先执行反序列化操作，解析出加密密钥块与签名信息块；再用自身 SM2 私钥解密密钥块，得到 SM4 密钥；随后使用 ClientSend 或 ClientReceive 的 SM2 公钥对签名数据进行验签。验签成功后，CentralServer 为当前通信线程分配独立存储空间，对该 SM4 密钥进行存储。密钥协商流程图如图 2 所示。

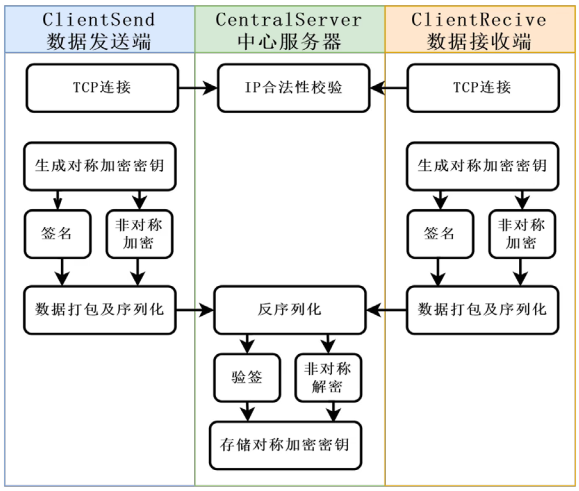

图 2 密钥协商流程图

3.2.2 数据传输流程

SM4 密钥协商完成后，将进行电网数据安全传输，数据传输交互流程图如图 3 所示。

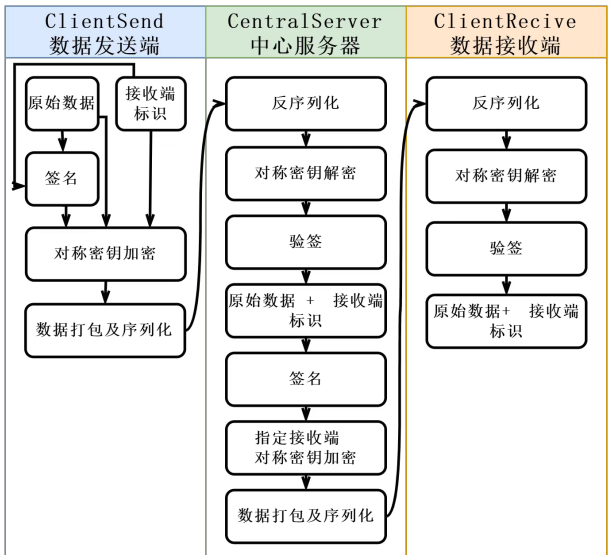

图 3 数据传输交互流程图

具体流程如下：

发送端处理：ClientSend 获取待传输的电网数据，如：CIM/E、JSON、QS 等。处理流程分为三步：首先将原始数据与目标接收端标识合并，再用 SM2 私钥对该合并信息进行签名；最后按规则整合数据，调用已协商的 SM4 密钥对该数据块进行对称加密。加密完成后进行序列化处理，通过 TCP 链路发送至 CentralServer。

中心服务器转发处理：CentralServer 接收数据后，先执行反序列化操作。随后通过对应线程存储的 SM4 密钥解密数据，得到原始数据、接收端标识、发送端签名；随后使用 ClientSend 的 SM2 公钥验签。验签通过后，CentralServer 用自身 SM2 私钥对原始数据和接收端标识进行签名；最后再调用与 ClientReceive 预先协商存储的 SM4 密钥，对数据包整体进行加密，序列化后向目标 ClientReceive 发起数据传输。

接收端处理：ClientReceive 接收 CentralServer 转发的数据后，先执行反序列化操作，通过自身存储的 SM4 密钥解密，得到原始数据与接收方标识信息。随后使用 CentralServer 的 SM2 公钥验签。验签通过后，ClientRecive 对原始电网数据进行存储，最终完成整个数据传输过程。

4 方案实现与效果验证

4.1 功能模块实现

4.1.1 对称密钥协商模块

ClientSend 与 ClientReceive 在完成 TCP 连接建立及 IP 白名单校验后，各自随机生成 SM4 密钥。再通过自身的私钥完成签名。随后使用 CentralServer 的 SM2 公钥对原始 SM4 密钥加密，最后将加密后的密钥与签名数据序列化后通过 TCP 链路发送 [8-10]。

```
    // 随机生成 16 字节密钥
    unsigned char *key = sm4->randKey();
    // 存储 sm4key
    m_sm4Key = string(reinterpret_cast<char*>(key), KEY_
LENGTH);
    // 使用私钥对 key 进行签名
    string sign = sm2->sign(PRIVATE_KEY_PEM, key, KEY_
LENGTH);
    ......
    // 使用服务端公钥对 key 进行加密
    string keySm4 = sm2->encrypt(SERVER_PUBLIC_
KEY, string(reinterpret_cast<char*>(key), KEY_LENGTH));
    ......
    TcpData data;
    // 对称密钥 0x01
    data.op = OperationType::SymmetricKey;
    data.content = QByteArray(keySm4.data(), keySm4.
size());
    ......
    // 中心服务端名称
    data.receive = QByteArray(SERVER_NAME);
    data.digest = QByteArray(hash.data(), hash.size());
    // 数据序列化
    QByteArray packet = TcpTransTools::serialize(data);
    ......
```

CentralServer 接收客户端传输的序列化数据后，先执行反序列化解析密钥块与签名信息，通过自身 SM2 私钥解密密钥块，获取 SM4 密钥；再使用 SM2 进行验签，验签通过后在当前线程存储 SM4 密钥。

```
    // 得到传输过来加密后的密钥数据
    string keySm4Ori = string(data->content.data(), data->
content.size());
    // 使用自己的私钥进行解密
    string key = sm2->decrypt(PRIVATE_KEY_
PEM, keySm4Ori)
    ......
```

```
    // 获取签名数据
    string sign = string(data->sign.data(), data->sign.size());
    // 获取对方公钥数据
    string peerPubKeyFile = this->getPeerPubKeyFile();
    // 验签
    bool verify = sm2-> verify(peerPubKeyFile, (const unsigned c
har *)key.data(), key.size(), sign);
    if(verify){
    this->m_sm4Key = key;
    return true;
    }
    ......
```

4.1.2 数据传输模块

ClientSend 获取待传输数据后，用自己的私钥将原始数据和接收端标识进行签名，随后打包为二进制数据块；调用已协商的 SM4 密钥对数据块加密，序列化后通过 TCP 链路发送至 CentralServer。

```
    // 对原始数据与接收端标识进行签名
    string waitingSig = fileContent + receive;
    string sign = sm2->sign(PRIVATE_KEY_PEM,
(const unsigned char *) waitingSig.data(), waitingSig.size());
    // 填充发送结构体
    TcpData data;
    data.op = OperationType::FileTransfer;
    data.content = QByteArray(fileContent.data(), fileContent.
size());
     ......
    // 序列化，并使用商议好的密钥对发送数据进行 sm4 加
密
    QByteArray packet = TcpTransTools::serialize(data, &m_
sm4Key);
    ......
```

CentralServer 接收序列化数据后，用对应 SM4 密钥解密数据，使用 ClientSend 的 SM2 公钥验签；验签通过后用自身 SM2 私钥重新签名，调用与目标 ClientReceive 协商的 SM4 密钥加密数据，序列化后转发至 ClientReceive。

```
    // 反序列化并解密数据
    bool deserializeStatus = TcpTransTools::deserialize(fullPack
et, data, &m_sm4Key);
    ......
    // 判断接收端是否在线或存在
    qintptr socketDescriptor;
    bool receiveStatus = isOnlineOrExist(data.receive, socketDe-
scriptor);
    ......
    // 找到对应线程，发送数据到接收端
    m_handlerMap[socketDescriptor]->sendData(data);
```

ClientReceive 接收 CentralServer 转发的序列化数据后，先反序列化并通过自身存储的 SM4 密钥解密，得到原始电网数据与中心服务器签名；再通过 CentralServer 的 SM2 公钥验签，验签通过后对原始数据进行存储，完成整个传输流程。

```
    // 反序列化并解密数据
    bool deserializeStatus = TcpTransTools::deserialize(fullPack
et, data, &m_sm4Key);
    ......
    // 获取签名数据
    string sign = string(data->sign.data(), data->sign.size());
    // 获取中心服务端公钥数据
    string serverPubKeyFile = this->getServerPubKeyFile();
    // 验签
    bool verify = sm2-> verify(serverPubKeyFile , (const unsigne
d char *)content.data(), content.size(), sign);
    if(verify){
    // 存储密钥
    ......
    }
```

4.2 安全性与性能测试分析

4.2.1 安全性测试分析

测试内容：聚焦国密算法合规性，核心安全属性（机密性、完整性、身份认证）及抗攻击能力，参考相关电力行业安全标准设计测试用例。

测试结果如下：

（1）SM2、SM3、SM4 算法通过官方 KAT 测试，实现与国密标准完全兼容；

（2）传输数据经 SM4 加密无明文泄露，SM3 哈希校验可阻断篡改数据，IP 白名单 +SM2 签名验签能拦截非法终端接入；

（3）可有效抵御暴力破解、差分攻击、重放攻击等常见威胁，满足电网数据传输安全要求。

4.2.2 性能测试分析

测试内容：以电网典型数据，如：1 kB JSON 数据、500 kB QS 数据、10 MB CIM/E 数据。为测试对象，考核传输时延，并发处理能力及资源占用率。

测试结果如下：

（1）传输时延：小数据包性能损耗 ≤5 ms，大数据包损耗占比 ≤21%，均低于电网业务实时性阈值；

（2）并发能力：支持 20 台终端并发稳定传输，服务器 CPU 及内存占用率可控，无数据丢失；

（3）资源占用：终端侧 CPU 占用 ≤15%，服务器侧资源消耗适中，不影响其他业务运行。

5 结论与展望

本文针对电网数据传输的安全需求，设计并实现了基于国密算法的安全传输方案。实现了数据传输的机密性、完整性、身份认证的功能；经安全性测试验证，方案符合国密标准与电力行业安全规范，可抵御常见密码攻击；性能测试表明，方案传输时延与资源占用可控，适配电网典型业务的实时性与并发需求。

基于自动化动态库劫持结合 Datalog 的堆漏洞检测系统设计

罗治祥[1] 赖怡聪[1] 曹佳伟[3] 李乐言[1,2*]
LUO Zhixiang LAI Yicong CAO Jiawei LI Leyan

摘 要 堆漏洞作为二进制程序中的重要安全威胁，长期以来受到安全研究者广泛关注。传统的静态分析与动态测试在检测精度、扩展性与适配性方面仍存在诸多不足。基于此，文章提出一种结合自动化动态库劫持与 Datalog 规则匹配的数据流分析方法，实现对堆漏洞（包括 Use-After-Free、Double Free、堆溢出等）的高效、精准检测。通过运行时劫持内存操作函数（如 malloc、free），实时收集堆内存变化信息，并使用可扩展的 Datalog 规则进行漏洞检测分析。实验结果表明，所提方法具有良好的检测精度与可移植性，能有效适应多平台环境并支持新型漏洞规则的扩展。

关键词 堆漏洞；动态库劫持；Datalog；Use-After-Free；动态分析；自动化漏洞检测

doi：10.3969/j.issn.1672-9528.2026.06.034

0 引言

近年来，随着软件系统复杂度的不断提高以及软件部署场景的日益多样化，软件漏洞的数量与影响范围呈现出爆发式增长趋势。尤其是在操作系统核心组件、浏览器引擎、车载系统、智能硬件固件等领域，由于开发周期紧迫、安全测试不完善以及安全意识缺失等问题，导致诸如堆栈溢出、Use-After-Free（UAF）、Double-Free 等内存相关漏洞[1]频发。堆内存漏洞作为攻击者实现远程代码执行、提权、沙箱逃逸等攻击的重要入口，其检测与修复已成为软件安全领域的重要课题。

为此，研究人员提出了诸多软件漏洞检测技术，整体

1. 工业和信息化部电子第五研究所 广东广州 511370
2. 智能产品质量评价与可靠性保障技术工业和信息化部重点实验室 广东广州 511370
3. 一汽大众汽车有限公司 吉林长春 130011

未来可以从两方面进一步优化：一是引入非对称密钥动态更新机制，结合业务场景定期更新 SM2 公钥与私钥，提升长期运行的安全性；二是引入类 CA 认证机制完成客户端身份校验。

参考文献：

[1] 国家密码管理局 .SM2 密码算法加密签名消息语法规范 :GM/T 0010—2023[S] 北京 : 国家密码管理局 ,2023.
[2] 李露 , 谢映宏 , 李蔚凡 , 等 . 基于国密算法的配电网终端通信安全架构研究 [J]. 浙江电力 ,2022,41(12):79-87.
[3] 康剑萍 , 王沈敏 , 杜竹青 .PKI 技术在信息安全中的应用 [J]. 自动化仪表 ,2020,41(4):107-110.
[4] 国家市场监督管理总局 , 中国国家标准化管理委员会 . 电力监控系统网络安全防护导则 :GB/T 36572-2018 [S]. 北京 : 中国国家标准化管理委员会 ,2018.
[5] 国家密码管理局 .SM4 分组密码算法 :GM/T 0002-2012[S]. 北京 : 国家密码管理局 ,2012.
[6] 邓大为 , 李可 , 陆俊 . 基于 CIM/E 文件的电网全景建模技术研究 [J]. 广东电力 ,2013,26(11):49-53.
[7] 申昕怡 , 宋广彦 . 电力调度数据的网络传输技术与安全渗透分析 [J]. 电气技术与经济 ,2024(2):112-114.
[8]Khlebnikov A, Adolfsen J. Demystifying cryptography with OpenSSL 3.0: discover the best techniques to enhance your network security with OpenSSL 3.0[M]. Birmingham—Mumbai: Packt Publishing, 2022.
[9]Dey N. Cross-platform development with Qt 6 and modern C++: design and build applications with modern graphical user interfaces without worrying about platform dependency[M]. Birmingham: Packt Publishing, 2021.
[10]Lazarus G. Mastering Qt 5:create stunning cross-platform applications using C++ with Qt widgets and QML with Qt quick[M]. Birmingham: Packt Publishing, 2016.

【作者简介】

陈风帆（1995—），男，湖南怀化人，本科，高级工程师，研究方向：信息化技术在电网中的应用。

（收稿日期：2025-11-25 修回日期：2026-06-02）