

摘要

鉴于ADS-B系统相对于现有的雷达航空监视系统的优越性，从雷达监视系统过渡到ADS-B系统是一个大势所趋的过程。根据美国联邦航空管理局(Federal Aviation Administration)关于下一代空中交通系统的实施规定，到2020年，所有经过美国的飞机必须配备ADS-B设备。由于ADS-B系统设计之初是没有考虑安全隐私方面的问题，因此，关于ADS-B安全隐私方面的研究也变得越来越迫切。目前的ADS-B系统可能面临的攻击大体可分为主动攻击和被动攻击，本课题主要解决被动攻击中的隐私保护问题。本课题采取的隐私保护的方法是对ADS-B数据进行加密的方法。说到加密第一个面临的问题就是算法的选择，ADS-B消息的加密算法需要满足符合ADS-B系统和ADS-B消息格式的特性。因此，针对这些特性和对各种算法的综合对比分析选出了认为是最适合的算法即格式保护算法中的FFX算法。根据ADS-B数据链的大小不可变特性，消息加密前和加密后的长度不能改变，且不能附加任何额外信息，故庞大臃肿的非对称性加密算法相对对称加密算法则没有优势。因此FFX算法是一种对称的具有格式保护特性的静态算法。对ADS-B消息进行加密还面临到的一个问题是ADS-B消息的哪些部分是可以加密的，哪些部分不能加密否则会导致无法解码等异常情况的出现。

经过仔细分析和实验，若要满足使飞行器对无关人员匿名以达到隐私保护的目的的话只对ADS-B消息中的ICAO24部分进行加密即可，对这部分进行加密即不会影响对ADS-B消息的解码也不会隐藏轨迹从而破坏航行安全性，且加解密的效率很高，因此这种方案可以投入到实际应用中。最后，为了对加解密的结果进行检验，本系统还实现了对ADS-B消息进行解码和可视化，使ADS-B消息包含的位置和飞行唯一标识信息等在地图上以直观的方式展现出来。

关键词： ADS-B，数据链隐私保护，格式保护算法，FFX算法，ADS-B解码

ABSTRACT

Given the ads-b system relative to the existing radar air monitoring system, the advantages of transition from radar surveillance system to the ads-b system is a process of the trend of The Times. According to the FAA (Federal Aviation Administration) implementation of the provisions on the next generation air transportation system, by 2020, all aircraft that want to fly through the US must be equipped with ads-b equipment . Due to the ads-b system haven't considered the safety and privacy issues in the beginning of design, therefore, study of ads-b security privacy is also becoming more and more urgent. The attacks ads-b system may face can be divided into active attack and passive attack, this topic mainly solve the problem of passive attack in privacy protection. This topic of privacy protection measures are of ads-b data encryption method. When it comes to encryption, the first problem is the choice of algorithm, ads-b message encryption algorithms need to meet and conform to the ads-b system ads-b message format features. Therefore, in view of these characteristics and comprehensive comparative analysis of various algorithms we have chosen FFX algorithm, one kind of the format preserving algorithm as the most suitable protection algorithm. According to the size of the chain of ads-b data immutable characteristics, message encryption can't change the length after encryption, and can't add any additional information to itself, so the bloated asymmetric encryption algorithm is relatively disadvantage when compare to symmetric encryption algorithms. So FFX algorithm is a kind of symmetrical static algorithm with format protection properties. The encryption of ads-b message also face to a problem is that which parts of ads-b message can be encrypted and which parts cannot be encrypted or it will lead to the situation that ADS-B message cannot be decoded or the presence of the other abnormal situation.

After careful analysis and experiment, to satisfy the purpose that to make aircraft anonymous which also means only encrypt ICAO24, one part of the ADS-B message which is the only Identity of the flight to irrelevant personnel in order to achieve the privacy protection , to encrypt this part of ads-b message does not affect decoding and won't

ABSTRACT

hide trajectory to damage navigation safety, and the encryption efficiency is very high, so the scheme can be put into practical application. Finally, in order to examine the results of encryption, the system also implements the ads-b message decoding and visualization, to make the position information and the only identity of flight which encode in the ADS-B message can be displayed on the map in an intuitive way.

Keywords: ADSB, Data Link privacy protection, Format Preserving Encryption, FFX encryption, ADS-B decoder

目 录

第1章 绪论	1
1.1 课题的背景	1
1.2 课题的研究价值和意义	2
1.3 课题的国内外研究现状	3
1.3.1 ADS-B 数据传输技术	4
1.3.2 ADS-B 数据链及仿真技术	4
1.3.3 ADS-B安全	5
1.4 课题难点、重点、核心问题及方向	6
1.4.1 课题的重点和难点	6
1.4.2 课题的核心问题及方向	6
第2章 相关概念和技术	7
2.1 ADS-B	7
2.1.1 ADS-B系统的组成和运行	7
2.1.2 ADS-B 数据链	9
2.1.2.1 MODE S 1090 ES	9
2.1.2.2 UAT	10
2.1.2.3 VDL MODE4	10
2.1.3 ADS-B消息格式	12
2.2 FPE	13
2.2.1 加密算法选择上的分析	15
2.2.2 FFX	16
2.3 OpenSky	17
2.4 本章小结	19
第3章 设计与实现	23
3.1 设计模型	23
3.1.1 系统模型	23
3.1.2 网络模型	23

目录

3.1.3 ADS-B安全模型	24
3.1.3.1 不考虑商用航空，只考虑通用航空	24
3.1.3.2 不考虑主动攻击如ghost plane; dos	24
3.1.3.3 遵守航空法不隐藏轨迹	25
3.2 系统流程图	25
3.3 模块设计	26
3.3.1 加解密模块	26
3.3.2 解码模块	28
3.3.2.1 广域位置解码（全球解码）	28
3.3.2.2 局域位置解码（本地解码）	29
3.3.3 可视化文件类型转换模块	31
3.4 算法编程	31
3.4.1 加解密部分	31
3.4.1.1 F函数部分代码	31
3.4.1.2 加密函数部分代码	33
3.4.1.3 解密函数部分代码	34
3.4.1.4 把加密部分的代码植入java的解码和可视化系统中	35
3.4.2 解码部分	48
3.4.2.1 全球解码	50
3.4.2.2 本地解码	53
3.4.3 可视化文件类型转换部分	54
3.5 本章小结	66
第4章 系统测试	67
4.1 测试环境	67
4.2 测试结果	67
4.2.1 加密	67
4.2.1.1 功能实现	67
4.2.1.2 稳定性	67
4.2.1.3 时效性	70
4.2.2 解密	70
4.2.2.1 功能实现	70
4.2.2.2 稳定性	70

目录

4.2.2.3 时效性	70
4.2.3 解码及解码结果的可视化	70
4.2.3.1 功能实现	70
4.3 总体分析	71
第5章 结束语	76
5.1 全文总结	76
5.2 后续工作展望	77
5.3 收获及体会	78
参考文献	79
致 谢	82
外文资料原文	83
外文资料译文	91

第1章 绪论

1.1 课题的背景

早期对飞机的监视，高交通密度区靠雷达监视，低交通密度区、边远地区或海洋只能靠驾驶员利用高频（HF）无线电台用电报或话音方式来发送飞机位置报告。高频无线电这种通信方式非常不可靠，极易受天气或其他突发事件影响。雷达监视是地面独立的对空监视，从一次监视雷达发展到二次监视雷达，从 A/C 模式发展到未来的 S 模式，是目前国际上普遍采用的技术。为了弥补变远地区和海洋等无法安装雷达的地区的监视之需，新航行系统(FANS)委员会推荐采用可靠空/地通信（如卫星通信）自动周期性用数字式数据报告飞机位置，虽是非独立监视，但仍然是有效的监视技术，为此推荐了自动相关监视 ADS(Automatic dependent surveillance)技术[1]。

国际民航组织 ICAO 定义 ADS 技术为：“ADS 是一种监视技术，由飞机将机上导航定位系统导出的数据通过数据链自动发送，这些数据至少包括飞机识别码，四维位置和所需附加数据”。ADS 技术是基于卫星定位和地/空数据链通信的航空器运行监视技术，是为越洋飞行的航空器在无法进行雷达监视的情况下，希望利用卫星实施监视所提出的解决方案。我国在沿青藏高原飞行的欧亚航路上建立的 L888 航路即采用了 ADS 的监视方式[2]。

ADS 可以分为 ADS-A (ADS-Addressing 选址式) 和 ADS-C (ADS-Contract 合约式) 两种模式。ADS-A/C 是飞机与航空管理单位之间首先建立起点对点的通信链接，在建立起链接之后，根据约定，飞机上导航设备自动地将飞机地有关信息传输给空中交通管制部门，同时地面也可以向飞行发送上行信息。通信链接的方式可以是甚高频数据链(VHF)、高频数据链(HF)、二次雷达(SSR)S 模式应答机、移动卫(AMSS)通信等。飞机向下报告的信息通过地面网络传到空管中心，经过数据解码转化等处理，飞机的位置等信息就可以显示在屏幕上。这种飞行员和管制员之间建立的数据链通信也叫做管制员飞行员数据链通信(CDPLC)。ADS-A 与 ADS-C 的区别是在建立起地空链接后，触发飞机向下报告的方式不同。ADS-A 是根据事先的约定来触发向下报告，这个约定可以是一定的时间间隔、过每个航路点等自动下传；ADS-C 是根据地面管制单位的询问来进行应答下传[3]。

ADS 是建立在地对空监视基础上的，八十年代后期发展的空中交通警告与防撞系统（TCAS）是建立在空对空监视基础上的，而用于机场场面活动监视是地对地监视。随着技术的发展，开始将这三种技术结合成一体，使活动着的飞机（包括滑行中的飞机）利用全球定位系统(Global Positioning System, GPS)和其他全球导航卫星系统(Global Navigation Satellite Systems ,GNSS) 获取飞机本身的监视数据。然后利用ADS-B OUT转发器将飞机自身的身份标识符、3D位置、速度、飞行目的地和其他空间信息以一定的周期广播出去。地面上的空中交通控制中心(Air Traffic Control, ATC) 和其他配备了ADS-B IN 的飞机都能接收到ADS-B信息。通过几年的试验。ICAO 的 ADS 专家组认可并定名该技术为广播式自动相关监视 ADS-B。广播式自动相关监视 ADS-B(Automatic dependent surveillance-Broadcast)，即航空器自动广播由机载导航设备和 GNSS 定位系统生成的精确定位信息，地面设备和其他航空器通过航空数据链接接收此信息，飞机以及地面系统通过高速数据链进行空对空、空对地以及地面的一体化协同监视。

ADS-B 与 ADS-A/C 最大的不同在于它不是采用点对点的通信方式，而是采用广播的方式。如此，不仅可以实现地面对飞机的监视，同时也可实现飞机与飞机之间的互相监视。由于 ADS 监视方式存在的诸多不足，如飞行信息处理时间过长不能满足终端区管制要求、监视方式费用较高等问题。因此，除了在洋区、偏远山区等不易安装地面设备的地区，考虑利用 ADS-A/C 对飞机进行监视之外，其他地区 ICAO 都推荐使用 ADS-B[4]。

1.2 课题的研究价值和意义

ADS-B技术是新航行系统中非常重要的通信和监视技术，把冲突探测、冲突避免、冲突解脱、ATC监视和ATC一致性监视以及CDTI综合信息显示有机的结合起来，为新航行系统增强和扩展了非常丰富的功能，同时也带来了潜在的经济效益和社会效益。ADS-B不仅可以应用于无雷达地区的远程航空器运行监视，而且与传统雷达监视技术相比，ADS-B技术具有使用成本低、精度误差小、监视能力强等明显优势，可以有效地提高了空管安全监视的水平。ADS-B技术能提高飞行中的航空器之间的相互监视能力。与TCAS相比，ADS-B的位置报告是自发广播式的，航空器之间无须发出问询即可接收和处理渐近航空器的位置报告。将ADS-B应用于TCAS，可以提高TCAS的性能，增强飞机之间的防撞能力，ADS-B系统的这一能力，使保持飞行安全间隔的责任更多地向空中转移，这是实现“自由飞

行”不可或缺的技术基础。ADS-B技术用于机场地面活动区，可以降低成本实现航空器的场面活动监视。利用ADS-B技术，通过接收和处理ADS-B广播信息，将活动航空器的监视从空中一直延伸到机场登机桥，因此能辅助场面监视雷达，实现“门到门”的空中交通管理。甚至可以不依赖场面监视雷达，实现机场地面移动目标的管理[3]。

FAA认为，ADS-B作为一种全新的监视技术，是实施自由飞行的奠基石。其主要功能（ADS-B、TIS-B、FIS-B）可以在监视、防撞、辅助进近上发挥作用，而且较原先一次雷达、二次雷达监视又具有明显的优势，对未来的飞行安全产生重大影响。ADS-B作为未来监视系统的发展方向必然是大势所趋，对于我国发展大型民机也有着良好的借鉴作用，C919飞机上也已确定安装ADS-B相关产品。因此，研究ADS-B技术有着极为重要的意义。

世界的空中交通管理正在从目前的非合作、独立的主监视雷达(primary surveillance radar, PSR)变成合作的、非独立的自动非独立监视广播(automatic dependent surveillance-broadcast ,ADS-B).根据美国联邦航空管理局(Federal Aviation Administration)关于下一代空中交通系统的实施规定，到2020年，所有经过美国的飞机必须配备ADS-B设备[2]，这意味着新一代现代化航空的到来。

但是ADS-B的通信协议设计之初并没有把安全问题考虑在内。根据它的开放式的协议，ADS-B消息都是没有考虑加密、数字签名等安全机制直接广播出去，任何具备ADS-B OUT设备的人都可以广播消息，具备ADS-B IN的人也可以接受任意飞机的消息。这样的开放式协议虽然拥有国际互操作性的优点，但是由于缺乏机密性、认证性也要承受各种被动攻击如信息泄露和主动攻击如欺骗假目标飞机(spoofing false target aircraft)。因此本课题将研究ADS-B数据链隐私保护的新方法，其主要思想是飞机假名的更新。

1.3 课题的国内外研究现状

对ADS-B理论技术的研究除了已制定并正在逐步完善的相关技术标准和规范之外（如SARPS, MASPS, MOPS, TSO等），还有对ADS-B数据传输，ADS-B数据链及仿真技术和ADS-B与雷达数据融合等方面的研究。

1.3.1 ADS-B 数据传输技术

自由飞行中，ADS-B 数据准确性对于飞行安全起到至关重要的作用，但是基于目前的设计，数据丢包，错误输入和数据欺骗都会降低数据的完整性，使得 ADS-B 无法从其它飞机上获得准确无误的状态和意图数据。鉴于此，K.Tysen Muellerh 和 Jimmy Krozel 等较早提出了“基于卡尔曼跟踪滤波的 ADS-B 意图信息完整性校验方法”[5]。之后，Jimmy Krozel 等还提出了“数据完整性校验的方法”，对 ADS-B 数据建立连续的实时状态估计系统，该系统采用卡尔曼滤波的方法平滑测量到的 ADS-B 数据中的噪声，识别错误的数据和数据丢包[6]，提供当前最佳的状态估计。此外，针对在二次雷达（SSR）无法覆盖的地方采用 ADS-B 监视的方法无法一直保证信号准确性的问题，Jimmy Krozel 等还提出了“对应于二次雷达跟踪监视的确认和授权批准技术（VV technique）”[7]，该技术不仅可以应用于机载系统，还可以引用于场面监视多点定位系统。上述研究都是从数据传输准确性的角度出发进行的。在国内，彭良福，郑超等人提出的基于 1090ES 数据链 ADS-B 的 CPR 算法[8]，对于飞机经度和纬度消息，采用简洁位置报告（Compact position reporting，CPR）的信源编码形式，针对全球位置和本地位置两种情况，实现飞机的经度和纬度消息的CPR 格式的编码和解码，提高了数据传输的效率。白松浩等人从工程实际中出发，为解决广播式自动相关监视信息和合约式自动相关监视信息相互转换提供了方法，实现了两种体制信息相互转换和信息共享，取得较好效果[9]。

1.3.2 ADS-B 数据链及仿真技术

ADS-B 技术可选的数据链技术有以下三种：1090 ES、UAT、VDL MODE 4。UAT 是专门为 ADS-B 设计的一种数据链系统，美国在其安装的通用飞机上采用的工作频率为 978MHz。陈志杰和朱晓辉提出了基于现有机载设备信道构建 UAT 数据链路的实现方案[10]，目的在于缩短设备研制周期、减少技术风险、节约投资。该方案在对标准 UAT 协议进行了剪裁和修改之后，利用了某型机载设备的发射机及其天线，通过增加 UAT 接收机和通信管理单元构成 UAT 机载端机，并单独配置地面端机后实现了方案的设计。通过仿真分析和原理性试验，验证得出该方案设计的 UAT 数据链系统与某型号系统可以协同工作，并且满足 UAT 系统设计要求，工作稳定可靠，有利于加快新一代空中交通管理系统的整体建设。黄飞、张军等人提出了基于 OPNET 的 UAT 数据链仿真模型[11]。在建立的该模型基础上，

通过比较在不同飞机数量情况下，端到端的时延、比特误速率、数据丢包率，通道利用率等参数的情况，得出结论：基于 UAT 的数据链可以为飞机和地面站之间提供了很好的通信服务，适宜为空管提供监视和态势感知服务。

1.3.3 ADS-B安全

Sampigethaya 和 Poovendran[12]第一次分析了ADS-B安全和隐私[13]。McCallie 等人[2]提出了目前的安全性分析，着重与分析可能遇到的攻击的本质和困难。他们都针对ADS-B的问题给出了一个系统级的高层次的和概括性的修补建议。Costin 、Francillon[14]和Schaefer也分析了ADS-B的安全性，主要研究让ADS-B利用目前的硬件和软件来提供一些可能的应对措施[15]。

Kovell等人[16]提及其他系统的数据融合和多侧面以及各种加密方案，他们也进一步的进行了一个对Kalman过滤和组认证的概念和可行操作的更加全面的分析。Nuseibeh等人进行了一个ADS-B的正式安全需求例子分析，提出了多侧面方案来处理可能的攻击情景[17]。Burbank等人提出了一个沟通网络通用概念来满足未来航空系统的要求，比如一个在终端区域和再规划路线的航空区域中的移动端的版本和无线网络的概念[18]。

Li 和Kamal[19]分析了以ADS-B为核心的整个FAA的下一代航空系统的安全性。他们建立了一个高层次的防御方案去分析下一代航空系统也提到一些通用的安全通信措施比如加密，认证和扩散范围等需要被深层次检验的ADS-B级别的安全方案。

与ADS-B安全性有关的工作的数量正在逐渐增加直到它在美国和其他欧洲国家被强制性使用，而且当时间越紧迫问题就会变得越迫切。当已经完成的工作属于安全必须和属于不同方面，尤其是多侧面和数据融合提供内在深层次的含义。目前的工作寻求去拓宽通信社区的理解和在一个更深更广的层面上强调这个问题。现在我们需要做的事是去整合目前考虑到的所有方面，在网络安全领域寻求一些可行的，有远见的想法，然后权衡利弊和检验应对他们的方法。

国内的Tso-Cho Chen 和 HsinChu[20]提出了一种通过通信双方共享一对私钥的的一种算法来为ADS-B提供机密性和认证性保护，虽然按照他们的算法确实是可以起到机密性和认证性的目的但是他们在算法中并没有考虑ADS-B消息的实际使用情况，例如他们在加密过程中采用的是把整个ADS-B消息加密的方式，但是ADS-B消息的前八位是不能加密的，否则接收者将无法辨认这是ADS-B消息因此导致无法解释和处理该消息；在例如他们的认证性的实现是通过在ADS-B消

息后面附加一个认证码的方式来为ADS-B消息提供认证性，但是他们没有考虑到ADS-B数据链路的带宽是一定的，只有112bits，因此，多出来的认证码部分将无法随原ADS-B消息一起发送出去。因此，本文将结合ADS-B消息的实际使用情况来解决ADS-B消息的机密性问题。

1.4 课题难点、重点、核心问题及方向

1.4.1 课题的重点和难点

课题的重点和难点都是要对ADS-B系统的运行以及对ADS-B消息的格式有很深入的认识，再针对ADS-B系统及网络传输的特点以及ADS-B的消息格式及含义选出适合且切合实际的解决方案。本课题要求对ADS-B数据链的隐私进行保护，即保护ADS-B消息的机密性，保护消息的机密性无外乎加密算法的选择问题。第二章会讨论到各种加密算法的异同以及把他们应用在ADS-B上的优势以及瓶颈。

1.4.2 课题的核心问题及方向

课题的核心是找到一种切合ADS-B应用实际的ADS-B加密算法并对该算法进行代码实现。然后进一步通过对消息进行解码和图像化显示来对用该加密算法加密后的ADS-B消息的实验结果进行验证和比较。

第2章 相关概念和技术

2.1 ADS-B

2.1.1 ADS-B系统的组成和运行

广播式自动相关监视(ADS-B, Automatic Dependent Surveillance Broadcast)是下一代空中交通管理的关键组成部分。现在使用的航空交通管理系统应用计算机和地面雷达实现飞机追踪和航空管理。虽然目前的航空系统表现的很好，但是它很容易受到干扰(例如天气)导致长时间的延误，更重要的是，这个航空运输管理系统已经快要达到它的工作处理速度的极限了。没有一个运输系统，它的航空交通预期增长将可能导致损失性巨大的航班延误和增加飞行危险系数的。因此，作为应对这个日益增长的担忧，美国国会建立了计划和发展工作小组来管理下一代航空管理系统的升级。ADS-B是自动的因为它不需要飞行员或其他控制者的干预就能自行广播消息。它是独立的监视系统因为飞行器需要依赖全球导航卫星系统(GNSS, Global Navigation Satellite System)来获取自身的的位置信息。更重要的是，它连续像附近的地面基站、附近的飞行器和地面交通工具(例如滑行中的飞行器)广播飞行器的位置信息和其他信息。其中需要涉及到标准信息格式和传输协议。ADS-B消息计划是在目前支持S模式的1090MHz的数据链路中传输。为了支持ADS-B, S模式的传播器将会结合一种叫拓展性应答机随机自发报告的特性。拓展性应答机随机自发报告提供了一个从传统S模式的升级和保证了在升级传输期间与现存系统的无缝整合。

ADS-B能提供连续的飞行器位置、身份、速度和其他信息的连续广播[2]。ADS-B设计之初并没有考虑安全问题，即没有安全机制去保护飞行器与航空管理者之间的消息传递的机密性、完整性和可靠性。因此，一个被动攻击者可以随意接收ADS-B消息并解码以获得飞行器的各种隐私信息以获取相关的利益。一个主动攻击者可以自行编码ADS-B消息并将其广播出去以制造并不存在的假飞行器又或者制造泛洪攻击使真正的ADS-B实时消息得不到接收。历史事件已经证明了不加密的数据链路会被有动机的对手利用。

ADS-B有两个功能性操作：(i) ADS-B OUT 和 (ii) ADS-B IN。ADS-B OUT 包含让飞行器或地面交通工具连续的产生ADS-B广播的功能，这个功能向航空控制系统提供了实时位置信息。ADS-B IN 具有接收和显示接收到的来自另一架飞行器的ADS-B OUT 消息，ADS-B IN 也允许飞行器接收地面控制中心提供的服务(例如天气预报更新)。需要注意的是，飞行器可以只装配ADS-B OUT 不装配ADS-B IN，这丝毫不会影响ADS-B OUT的工作。

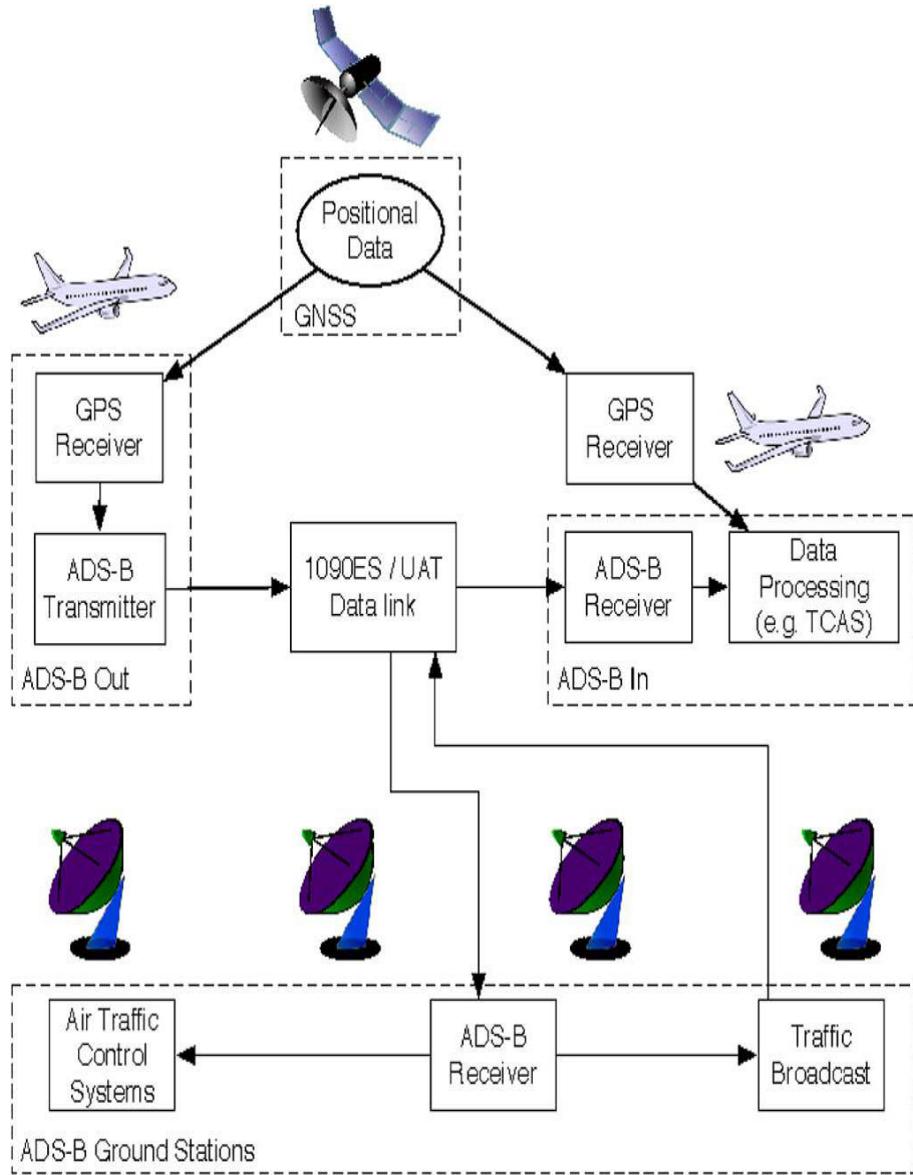


图 2-1 ADS-B系统的功能性组件及运行模型

图2-1介绍了ADS-B系统的功能性组件和模块以及提供了ADS-B运行的总的概括图。ADS-B数据在三个主要组件中交换，这三个组件分别为发送飞行器，接收

飞行器和地面基站。

发送飞行器：一架装有ADS-B OUT的飞行器通过ADS-B OUT中的GPS接收器从GNSS中获取自己的位置信息，随后把位置连同身份信息、海拔高度、速度(包括速率大小和速度方向)通过飞机上的S模式的传播器(transponder)在1090 ES数据链路中广播出去。

接收飞行器：另一架配备有ADS-B IN的在其附近的飞行器通过ADS-B IN组件中的ADS-B 接收器接收到第一架飞行器发送的ADS-B消息并把它传输到数据处理组件(TCAS)，在数据处理组件中对ADS-B 消息进行解码和可视化处理则可看到第一架飞行器所发送的ADS-B消息所包含的内容，即轨迹或者飞行器标识符等信息。

地面基站：同理，地面基站也配备有ADS-B接收器，同样可从1090ES数据链路中接收到ADS-B消息并同时把消息传输给空中交通控制系统和将接收到的消息广播出去。

2.1.2 ADS-B 数据链

ADS-B系统主要包括数据链、地面站、ATC系统、机载设备等。其中数据链是ADS-B技术非常重要的一个部分。实现 ADS-B 数据传输的技术称为 ADS-B 数据链，可以将数据链看作是一个专用的网络系统，包含了相关的通信协议、数据链设备和通信终端[21]。最初用于军事领域，目前已经逐步向民用领域扩展。目前，在 ADS-B 中使用的数据链一共有三种MODE S 1090 ES、UAT、VDL MODE 4[22]。

2.1.2.1 MODE S 1090 ES

1090 ES (Extended Squitter, 扩展电文)的命名中包含两层含义。1090 指的是该数据链的下行传输频带是 1090MHz。ES 指的是对原有 ADS-B 报文长度的扩展。原有的报文长度一般为 56-112 比特。1090 ES 是一种S模式的数据链，支持一对一的询问-应答机制[23]。S是 Selective 的意思，指的是可以对航空器进行选择通信。1090 ES 采用 PPM (脉冲位置调制) 编码。它的使用频1090MHz，信息格式是简单的脉冲位置编码，数据传输速率为1Mb/s。1090 ES用发射机和发射天线来传送不同的消息，包括24比特码、高度、呼号等。由于消息格式简单，承载信息能力较弱，所以在一个编码中只能传输一个特定类型的信息。而这些消息的更新率也有所不同，位置消息和速度消息每0.4-0.6s更新一次，标识消息和类型消息每4.8-5.2s更新一次，航路点消息每1.6-1.8s更新一次。1090ES采用扩展型断续振荡的方

式，由112个信息脉冲构成的S模式ADS-B长应答信号通过机载设备每隔1s广播一次。112位信息脉冲串的前88位为消息位、后24位为奇偶校验位。具体信息内容包括经度、纬度、方位和速度等信息，如图2-2所示。



图 2-2 1090ES 的消息格式

1090 ES 是由 ICAO 推荐的、唯一一个可以在全球范围内使用的数据链技术，得到了美国、欧洲、亚洲等大部分国家的承认和应用。在第11次国际航空会议上，已经将1090 ES 作为 ADS-B 主要数据链技术，并制定了相关的协议和标准[24]。

2.1.2.2 UAT

UAT 的全称是 Universal Access Transceiver，即通用访问收发机。UAT 是美国专门为 ADS-B 设计的数据链，它的特点是成本低、使用方便、通用性强、具有地面站向航空器上行广播数据的能力、传输带宽较大[25]。UAT 采用时分复用的方式传递数据，基本传输单元为 UAT 帧，每 1 秒钟为一帧。每一个 UTC 秒开始一帧数据，每一帧传输分为两个部分，前 188 毫秒由地面站发送上行数据，后 812 毫秒由机载 ADS-B OUT 设备下行发送 ADS-B 报文[26]。UAT 地面站上行传输采用固定时隙接入方式，机载 ADS-B OUT 设备下行传输采用随机接入方式。传输速率为 1Mbps，工作频带为 978MHz。UAT 的最长时间度量单位是 MSO，每个 MSO 时长为 250ms，一帧共 4000 个 MSO。如图2-3所示。地面站最小信息传输单位是时隙，一个时隙有 22 个 MSO。因此第一段由 32 个时隙构成，每个地面站被分配一个时隙。第二段被飞机和地面车辆所共享，每个飞机/车辆在 ADS-B 段中随机选择 MSO 进行传送。

2.1.2.3 VDL MODE4

VDL MODE4 又称为 VDL-4，全称是 VHF Data Link-Mode 4，是基于 STDMA (Self-Organized Time-Division Multiple Access，自组织时分多址) 的 ADS-B 数据链技术[27]。它以 ADS-B 应用为主，同时支持 TIS-B 和 FIS-B。VDL-4 由 ICAO

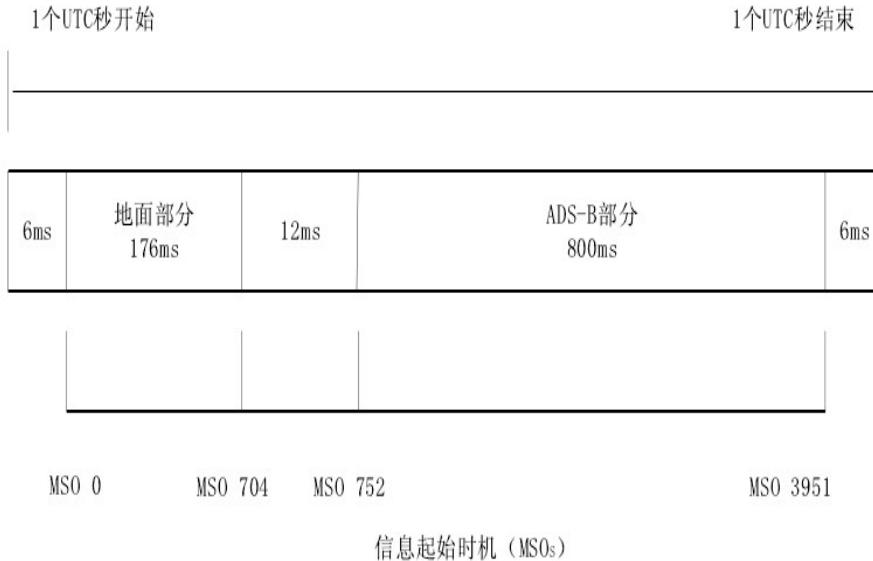


图 2-3 UAT的帧格式

和 ETSI (European Telecommunications Standards Institute, 欧洲电信标准协会) 共同推荐的数据链技术，有相应的 OSI 标准。VDL-4 利用 GNSS 进行定位和严格的时间同步，具有 2 个全领标示信道和一个附加信道，可以高效传输重复性信息，支持各种实时应用[28]。无需地面系统的支持，VDL-4便可建立空空通信或空地通信，使用VIP (VDL Mode-4 Interface Protocol, VDL 模式 4 接口协议) 进行工作。VDL-4 的帧时间长度达到了 60 秒，分为 4500 个等长时隙，每个时隙 13.33 毫秒。传输速率为 19.2kbps 。 VDL MODE 4采用自组织式时分多路(STDMA)和超长帧技术，参与通信的所有设备都由 GNSS 进行严格的时间同步，STDMA中使用者的共同时间标准是UTC。从而避免了数据发送的时隙冲突，并可以实现数据发送的计划性。表2-1是对三种数据链技术的比较[29]。

这三种数据链都能满足当前ADS-B应用的基本要求，但都不甚完美。由于欧洲和美国两大商用飞机制造基地的产品生产标准不同，在选用地空数据链时，出于兼容现有机载设备、兼顾终极发展目标的考虑，政策取向也各有侧重。根据民航总局关于ADS-B的技术政策，并考虑到我国未来空管系统与国际接轨问题及在全球范围内的相互操作性(目前只有S模式1090 ES数据链技术是被各个国家、地区和组织所接受的标准)，我国在实施ADS-B项目计划时优先考虑使用1090 ES作为数据链路技术。同时，考虑UAT机载设备和地面站的性价比、功能特性和适用范围，在通用航空飞行活动频繁的特殊区域可以考虑采用UAT作为支持ADS-B数据

链技术，暂不考虑采用VDL MODE 4作为我国ADS-B系统的数据链路。

2.1.3 ADS-B消息格式

图2-4所示，ADS-B消息由112 bits组成并分成5个区。

Bit from	Bit to	Abbr.	Name
1	5	DF	Downlink Format
6	8	CA	Message Subtype
9	32	ICAO24	ICAO aircraft address
33	88	DATA	Data frame
89	112	PC	Parity check

图 2-4 ADS-B 1090ES 消息格式

下面将对 ADS-B 报文结构的基本字段进行进一步的说明。

1. DF(Downlink Format)字段

DF 字段长度是 5 位，用于区分不同的下行链路格式（Downlink Format）。DF 的值可以是 17、18 或 19。DF=17 用于 S 模式应答机发出 ADS-B 报文。DF=18 用于非 S 模式应答机发出 ADS-B 报文或 TIS-B 报文。DF=19 用于军事用途，非军事应用不会涉及到该类型报文。

2. CA/CF(Capability/Code Format)字段

CA/CF 字段的长度是 3 位，在不同的 DF 值下有不同的含义。DF=17 时，该字段是 CA 字段，含义是 S 模式应答机的能力。DF=18 时，该字段是 CF 字段，含义是编码格式（Code Format），用于区分 ME 字段的内容、AA 地址的类型、以及两类特殊的报文。CF=0 或 1 时，表明该报文是 ADS-B 报文。

3. AA(ICAO24 Aircraft Address)字段

AA 字段的长度是 24 位，包含了发射装置的地址信息。地址的类型有两类：ICAO 地址和非 ICAO 地址。ICAO 地址是飞机的地址，非 ICAO 地址是匿名地址、地面车辆地址或表面障碍物地址。

4. DATA(Data frame)字段

DATA 字段的长度是 56 位，包含了 ADS-B 报文的业务数据，称之为 ADS-B 业务报文。关于 DATA 字段的格式将在 ADS-B 业务报文中进一步说明。5. PI(Parity Check)字段

PI字段的长度是 24 位，是一个下行链路字段，含义是奇偶性（Parity）和一致性(Identity)。该字段包含了编码标签（Code Label, CL）和询问器编码（Interrogator Code, IC）的奇偶性。在 ADS-B 报文中，CL 和 IC 的值都是 0，也就是说 PI 字段的内容填充了 24 个 0。

因此，可以总结出本系统应接收并处理的 ADS-B 报文格式为：DF=17。

下面具体介绍Data frame区域的各个字段的含义：如图2-5所示，Data frame(33bit-88bit)中的33bit-37bit为TC(Type Code),Type Code的值不同，则这条ADS-B消息所包含的消息也不同。如图2-6所示，当Type Code 为不同值时ADS-B消息所包含的不同的信息。

Bit from	Bit to	Abbr.	Name
33	37	TC	Type Code

图 2-5 Type Code位置

2.2 FPE

根据已有的研究成果，刘哲理等人[30]从两个方面对FPE进行了定义，包括基本FPE和一般化FPE。基本FPE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间；一般化FPE则强调待加密消息空间的复杂性决定FPE问题的复杂性。

定义1 基本FPE： FPE可简单描述为一个密码： $E: K \times X \rightarrow X$ (1) 其中K 为密钥空间，X 为消息空间[31]。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。例如，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。即明文与密文都处于消息空间 $/0, 1, \dots, 9/n$ 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，而

DF	TC	Content
17	1 to 4	Aircraft identification
17	5 to 8	Surface position
17	9 to 18	Airborne position (Baro Alt)
17	19	Airborne velocities
17	20 to 22	Airborne position (GNSS Height)
17	23	Test message
17	24	Surface system status
17	25 to 27	Reserved
17	28	Extended squitter AC status
17	29	Target state and status (V.2)
17	30	Reserved
17	31	Aircraft Operation status

图 2-6 Type Code值与ADS-B消息含义的对应

且还有特定位为‘-’，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 $\omega \in \Omega$ ，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 $X_\omega, \omega \in \Omega$ 有关。 X_ω 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^ω, t 。

定义2 一般化FPE：一般化FPE可描述为一个密码： $E: K \times \Omega \times T \times X \rightarrow X \cup \perp$ (2) 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 [32]。可通过算法三元组 $E_{FPE} = (\text{setup}, \text{encryption}, \text{decryption})$ 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：(1) 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；(2) 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 ；(3) 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 ，返回消息空间分片 X_ω 内的密文 y 或者 \perp 。该算法执行 $E_K^\Omega, T(X) = E(K, \Omega, T, X), E_K \Omega, T(\cdot)$ 是 X_Ω 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 \perp 。

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 \perp 。该算法是**encryption**的逆运算：如果 $y \in X_\omega$ ，则返回 $x = D_t^\omega, t(y)$ ，否则返回 \perp 。

2.2.1 加密算法选择上的分析

根据选题‘ADS-B数据链隐私保护算法的设计与实现’，本文只考虑被动攻击的防范问题即ADS-B信息隐私的保护问题。对于公共航班和货机，由于它们具有固定时刻表、公开的航线和永久性的身份标识符，所以并不存在位置隐私的保护需求。本文主要考虑的是航班以外的民航专用航空(general aviation)的位置隐私安全的保护问题。专用航空包括很大范围的飞行器，包括私人飞机、执法和紧急服务飞机和商务飞机。

本文采用加密ADS-B报文的方式来保护飞机的位置隐私。加密算法大体分为两种：对称加密算法和非对称加密算法。对称加密算法使用相同的密钥进行加密和解密。非对称加密算法使用唯一的公私钥对分别进行加密和解密。下面分析一下这两算不同的算法在ADS-B数据链加密应用上的好处与瓶颈。

公共密钥基础设施(The public key infrastructure ,PKI) 使用的就是非对称算法。PKI常常用于保证未知结构网络的安全性，但是关联节点间的身份标识符是预先决定并持续不变的。而且PKI不仅能提供机密性服务也能提供认证性服务。但是PKI需要一个可信的第三方来对密钥对的产生和认证进行统一的管理。而且公钥算法要求使用目标接收方的公钥进行加密，如果这个消息需要同时发送给很多不同的人，那么这条消息则需要被用不同的公钥加密并分别发送给对应的接收方。而且ADS-B消息的长度固定且很小，要装下一个庞大的证书在具体实践上是几乎不可能的。虽然飞机能够知道它附近有哪些飞机和ATC，但是这种广播式的逐一发送效率太低的缺陷足以掩盖ADS-B相对于过时的方案PSR的优点，且不说全球的飞行器的密钥对存储和管理的问题。

对称密钥算法相对公钥算法来说效率更高，规模性也更能得到保证因为只有一个密钥被同时用于加密和解密。关于密钥的分发和管理，可以使用飞行员控制数据链路通信(the controller pilot data link communication ,CPDLC)来初次协商或者更新密钥。在初始化飞行计划期间飞机会和ATC进行一起身份确认以进行同步。因此，密钥交换可以包含在CPDLC登录中，这为密钥的安全交换提供了一个机会。至于密钥的管理，可以由ATC控制员随机产生并通过CPDLC传输给每个相关飞行员。由于ADS-B广播消息的日期信息只有很小的改变，为了保护系统免受已知明文攻击，块加密(e.g., AES and 3DES)比流加密更适用于ADS-B的实际操作环境。下行链路格式区和CA字段固定为8 bits, 用于向接收方表明消息的处理方式，不能加密。这意味着只能加密剩下的104 bits。但是目前的快加密算法一般以64或128 bits 分块。通常情况下如果消息不能满足分块大小的话可以通过填充的方式来解决，但是对于ADS-B消息而言不行，因为下层的ADS-B广播框架已经设计成112 bits的固定数据长度，因此需要一个能支持任意块大小的加密算法。

2.2.2 FFX

Bellare等人在2010年提出了FFX模型[34][33]，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 $Char^n$ 上的FPE问题， $Char^n$ 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 $Char^n$ 上的保留格式加密。格式保护加密不仅应该做到传统块加密(例如AES)能够做到的事，如使输入明文如输出密文的大小一样，而且还

能比传统加密更加普遍化，而不是仅仅能加密大小为64bits 或128bits的信息(例如AES)。实现格式保护加密的一种方式使使用Feistel-based模式的操作。作为基础的轮功能是基于传统的块加密算法，例如AES。两种实现了FPE要求的模式分别是FFX和BPS。但是FFX比BPS在模式上要更加的开放和更实用，因此本文选择的是FFX模式的FPE算法。 FFX又分为A2和A10两种模式， FFX-A2模式用于加密8-128bits的二进制比特串而FFX-A10用于加密4-36bits的十进制数字。对于长度充足的字符串，两种模式都使用12轮的Feistel。但是轮数会随着消息变短而迅速增长，对于允许的最短消息长度FFX-A10可以达到24轮或者FFX-A2可以达到36轮。 FFX[radix]模式。这是一种新的FFX参数收集方式。首先，这样做拓展了允许的radix值，任何可行的radix值都可以被使用，而不仅仅是2或者10。其次，还扩大了允许的加密消息长度，允许有效的任意长度字符串被加密。最后，在轮数的选择上FFX[radix]模式比FFX-A2或FFX-A10要更进步一些， FFX[radix]的轮数是常量而不是随着消息的长度变化而变化。 Radix 的可选值介于2到 2^{16} 之间。 Radix的数值等于被加密的字符串的字符种类的个数。例如，被加密字符串是一个二进制比特串的话， radix就等于2，因为比特串里只有0或者1两种类型的字符。 FFX[radix]是使用基于AES的Feistel模型。 FFX[radix]有效的统一和拓展了FFX-A2和FFX-A10模式。所以即使加密轮数一样， FFX[2]与FFX[10]也和FFX-A2和FFX-A10不一样。块加密加功能由 $a_1 \cdots a_n$ 由 $b_1 \cdots b_n = c_1 \cdots c_n$ 定义, $c_1 \cdots c_n$ 是唯一满足

$\sum c_i[\text{radix}]^{(n-i)} = (\sum a_i[\text{radix}]^{(n-i)} + \sum b_i[\text{radix}]^{(n-i)})$ 的字符串。块加密曰功能的运算特性是当 $X \oplus Y = Z$ ，则 $Y \oplus Z = X$ 。 $[s]^i$ 表示s是由i个字节编码的字符串，例如i=1,则s是由一个字节即四个比特编码的字符串，即s是一个16进制的数字。

2.3 OpenSky

Opensky(<https://opensky-network.org>)是一个专门的在它的感知范围内收集ADS-B数据并用于飞行研究的感知网络。目前的传感器部署范围是欧洲中部，如图2-10所示：

这些传感器由志愿者们提供和部署，然后志愿者们将他们的信息发送到一个数据处理中心，在那里所有的数据被解码、评估和存储在一个大型数据库里。这些数据不仅包括ADS-B原始数据，还包括时间戳、传感器位置等可用于深层次研究的数据。它的部件组成细节描述如图2-11：

<pre> 10 algorithm FFX.Encrypt(K, T, X) 11 if $K \notin \text{Keys}$ or $T \notin \text{Tweaks}$ or 12 $X \notin \text{Chars}^*$ or $X \notin \text{Lengths}$ 13 then return \perp 14 $n \leftarrow X$; $\ell \leftarrow \text{split}(n)$; $r \leftarrow \text{rnds}(n)$ 15 $A \leftarrow X[1.. \ell]$; $B \leftarrow X[\ell+1.. n]$ 16 for $i \leftarrow 0$ to $r-1$ do 17 $C \leftarrow A \boxplus F_K(n, T, i, B)$ 18 $A \leftarrow B$; $B \leftarrow C$ 19 return $A \parallel B$ </pre>	<pre> 20 algorithm FFX.Decrypt(K, T, Y) 21 if $K \notin \text{Keys}$ or $T \notin \text{Tweaks}$ or 22 $Y \notin \text{Chars}^*$ or $Y \notin \text{Lengths}$ 23 then return \perp 24 $n \leftarrow Y$; $\ell \leftarrow \text{split}(n)$; $r \leftarrow \text{rnds}(n)$ 25 $A \leftarrow Y[1.. \ell]$; $B \leftarrow Y[\ell+1.. n]$ 26 for $i \leftarrow r-1$ downto 0 do 27 $C \leftarrow B$; $B \leftarrow A$ 28 $A \leftarrow C \boxminus F_K(n, T, i, B)$ 29 return $A \parallel B$ </pre>
---	---

图 2-7 FFX EncryptDecrypt

<pre> 30 algorithm $F_K(n, T, i, B)$ 31 vers $\leftarrow 1$; $t \leftarrow T _8$; $\beta \leftarrow \lceil n/2 \rceil$; $b \leftarrow \lceil \lceil \beta \log_2(\text{radix}) \rceil / 8 \rceil$; $d \leftarrow 4\lceil b/4 \rceil$ 32 if EVEN(i) then $m \leftarrow \lfloor n/2 \rfloor$ else $m \leftarrow \lceil n/2 \rceil$ 33 $P \leftarrow [\text{vers}]^1 \parallel [\text{method}]^1 \parallel [\text{addition}]^1 \parallel [\text{radix}]^3 \parallel [\text{rnds}(n)]^1 \parallel [\text{split}(n)]^1 \parallel [n]^4 \parallel [t]^4$ 34 $Q \leftarrow T \parallel [0]^{(-t-b-1) \bmod 16} \parallel [i]^1 \parallel [\text{NUM}_{\text{radix}}(B)]^b$ 35 $Y \leftarrow \text{CBC-MAC}_K(P \parallel Q)$ 36 $Y \leftarrow \text{first } d+4 \text{ bytes of } (Y \parallel \text{AES}_K(Y \oplus [1]^{16}) \parallel \text{AES}_K(Y \oplus [2]^{16}) \parallel \text{AES}_K(Y \oplus [3]^{16}) \dots)$ 37 $y \leftarrow \text{NUM}_2(Y)$ 38 $z \leftarrow y \bmod \text{radix}^m$ 39 return $\text{STR}_{\text{radix}}^m(z)$ </pre>

图 2-8 F函数

radix	a number $\text{radix} \in [2..2^{16}]$	alphabet is $\text{Chars} = \{0, 1, \dots, \text{radix} - 1\}$
Lengths	$[\text{minlen} .. \text{ maxlen}]$ where $\text{minlen} = 2$ if $\text{radix} \geq 10$ and $\text{minlen} = 8$ otherwise; and $\text{ maxlen} = 2^{32} - 1$.	permitted message lengths
Keys	$\{0, 1\}^{128}$	128-bit AES keys
Tweaks	$\text{BYTE}^{\leq \text{ maxlen}}$ where $\text{ maxlen} = 2^{32} - 1$	tweaks are arbitrary byte strings
addition	1	blockwise addition
method	2	alternating Feistel
$\text{split}(n)$	$\lfloor n/2 \rfloor$	maximally balanced Feistel
$\text{rnds}(n)$	10	number of rounds
F	given below	AES-based round function

图 2-9 FFX加密算法参数

2.4 本章小结

本章对该课题接下来章节中需要用到的概念和技术做了详细的讲解。首先介绍了ADS-B系统的概念、组成、运行和设计，ADS-B系统的网络和ADS-B消息的格式，使读者对本课题的主要研究对象ADS-B有系统和深入的认识。然后介绍了格式保持加密这种算法，并具体分析了为什么要选择这种算法，也介绍了所选择的格式保护算法的具体的某一种模式(FFX)。最后，向读者介绍一个对本次毕业设计提供了重要帮助的一个非常实用的公益研究性质的网站。该网站为本次课题提供了研究所必须的ADS-B原始数据以及对处理这些数据很实用的一些具体函数的实现。

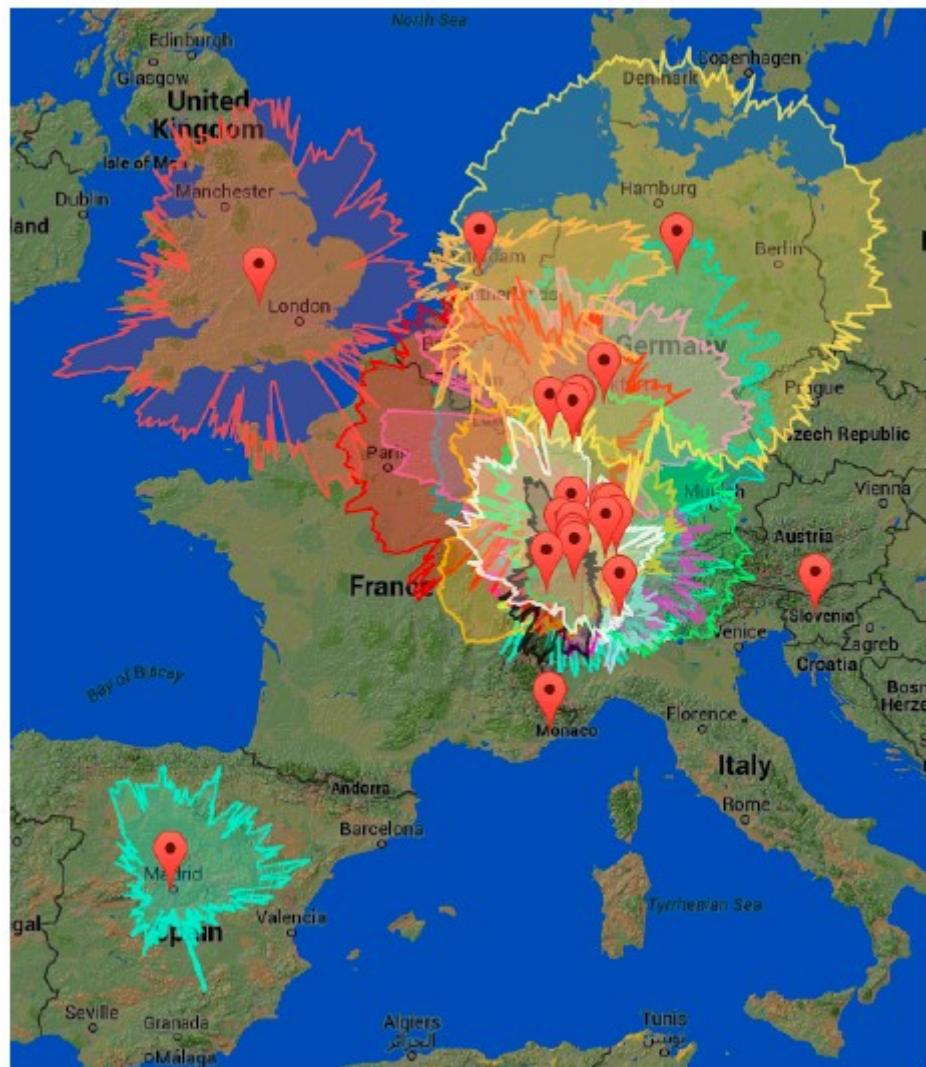


图 2-10 OpenSky 传感器分布图

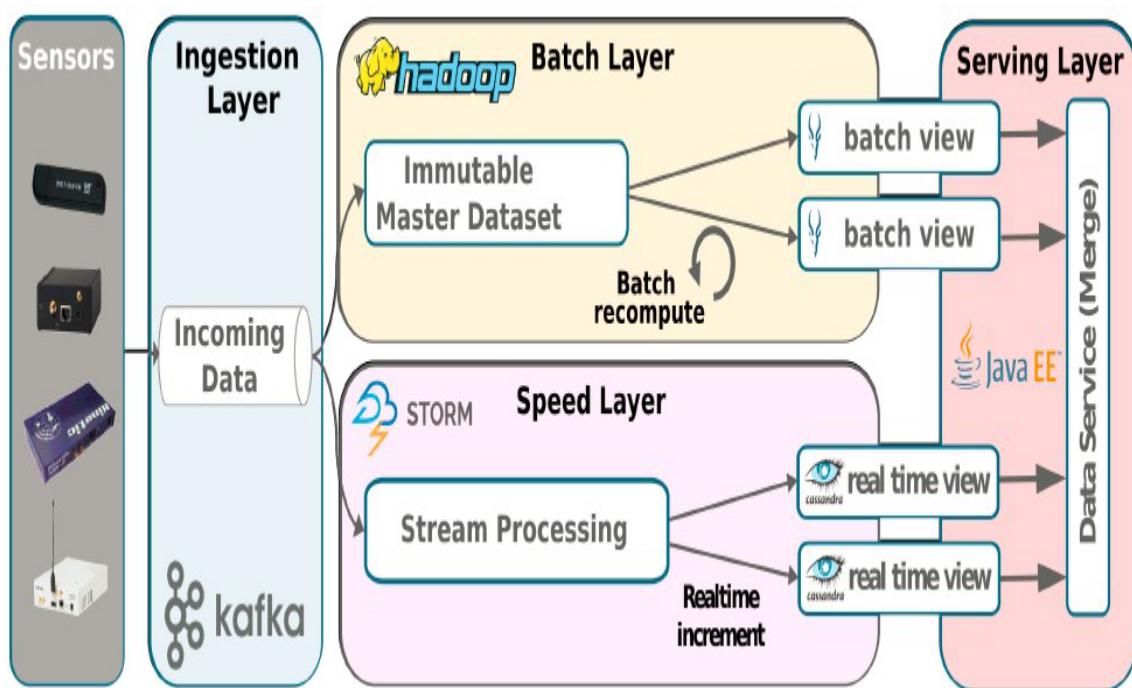


图 2-11 OpenSky数据处理中心组成部件

表 2-1 三种数据链技术比较

	1090MHz Mode S	UAT	VDL Mode 4
使用频率	1090MHz	建议使用 DME 频段, 没有达成世界范围内的标准, 在美国使用 978 MHz	建议使用 VHF 频段, 需要多信道。没有世界范围内的标准或共识。
码速率	1Mb/s	1Mb/s	19.2Kb/s
访问方式	随机访问	下行: 有分配ADS-B 块的随机方式; 上行: 固定分配	自组织时隙: 时隙由 GPS 同步
地-空通信距离	>200Nm 取决于地面系统的天线增益和灵敏度	>200Nm 取决于地面系统的天线增益和灵敏度	>200Nm 取决于地面系统的天线增益和灵敏度
ICAO 标准	Mode S SARPS Annex 10 Amendment 77 via SCRSP	目前还不是 ICAO SARPS	Annex 10 AMCP
主要文件	D0260, D0260A, D0181C, ED73A, ED86	D0282	Eurocae ED108
实施方法	升级现有的应答机软件, 使用现有的天线; 通用航空器需要加装新的机载设备。	加装新的机载电子设备、收发信机、天线。	加装新的机载电子设备、收发信机、天线。

第3章 设计与实现

3.1 设计模型

3.1.1 系统模型

本系统假设的应用模型如图一所示，GNSS把飞机的位置信息发送给飞机的GPS Receiver,再由ADS-B Transmitter将飞机自身的身份标识符、3D位置、速度、飞行目的地和其他空间信息编码成ADS-B消息并对其进行加密然后以一定的周期广播出去。地面上的空中交通控制中心(Air Traffic Control, ATC) 和其他附近一定范围内飞机的ADS-B Receiver都能接收到ADS-B信息并对其进行处理或广播。只有预先与发送ADS-B消息飞行器进行密钥协商的其他飞行器或地面基站的ADS-B IN才能解码出真实的发送方的ICAO24。本课题由于只研究ADS-B的数据链隐私保护即ADS-B消息的加解密，故并没有模拟ADS-B消息的发送和接收过程。加密过程相当于在ADS-B OUT 中实现，解密过程和解码显示过程相当于在ADS-B IN中实现。

3.1.2 网络模型

本课题的网络模型设计两种通信模式，其中GPS与ADS-B Out的GPS Receiver之间通过GNSS数据链路通信，而ADS-B Out、ADS-B In 和ADS-B Ground Stations 之间通过MODE S 1090 ES数据链路通信。本课题重点讨论ADS-B Out、ADS-B In 和ADS-B Ground Stations 之间的通信模式。在本课题中这三者之间通信采用的是MODE S 1090 ES 的ADS-B数据链技术。1090MHz拓展性应答机随机自发报告消息是为商用航空设计的而且也是ADS-B OUT的国际化标准。1090ES采用扩展型断续振荡的方式，由112个信息脉冲构成的S模式ADS-B长应答信号通过机载设备每隔1s广播一次。112位信息脉冲串的前88位为消息位、后24位为奇偶校验位。具体信息内容包括经度、纬度、方位和速度等信息。下面简单介绍下1090MHz拓展性应答机随机自发报告(extended squitter)消息的格式。这个拓展性应答机随机自发报告消息长112bits,其中包含了56bits的ADS-B信息。前同步码(Preamble)包含了一个特定的用于实现同步的比特序列。下行链路(Downlink Format)格式域为5bits,这

5bits表明了消息的类型—这个域被设置为17的话则代表了这是一条拓展性应答机随机自发报告消息。3bits的容量域(Capability)代表了S模式传输器的通信容量能力。飞行器地址(Aircraft Address)是24bits的传播器也及装载了该传播器的飞行器的唯一标识符。不存在两架不同的飞行器拥有统一个统一标识符。ADS-B数据(ADS-B Data)长56bits也包含了相关的监视数据(例如：身份信息，位置信息，速度信息，紧急码和质量等级)。奇偶校验值(Parity Check)是一个24bits的用于让接受者在接收到消息时检测传输错误的语。脉冲式调整方案被用于消息的编码和传输。ADS-B使用了一个广播式的通信范例。飞行器之间传输信息时并没有考虑敌人会接收信息，任何距离飞行器一定范围的敌人都可以接收并解码传输的消息。另外，传输的协议不要求消息接收的确认也没有实现任何的保持在线功能(例如：如果一架飞行器的ADS-B消息在一个确定的时间段内没有被地面基站或其他飞行器所接收到，这个系统将不会再询问这架飞行器的状态)。

本课题重点在于加解密算法的研究和实现故并未考虑ADS-B具体网络传输过程的模拟，只做原理性的介绍。

3.1.3 ADS-B安全模型

3.1.3.1 不考虑商用航空，只考虑通用航空

商用航空主要指固定的航班，根据航空飞行发，商用航班不允许匿名或隐藏轨迹，不仅因为这样做不利于飞行安全而且也不便于旅客查询航班的信息，并且商用航空每架飞机都有指定的航行路线，根据航行路线就可以推出航班号，因此根本没有匿名的必要。而通用航空则完全不同，通用航空包含一大部分领域的飞机，包括私人飞机、法院或其他紧急服务的飞机到商业飞机。这些飞机出于个人、政治、医疗或商业愿意不愿意暴露飞行动机和飞行目的地及喜好，因此存在保证保证他们的身份匿名的需求。

3.1.3.2 不考虑主动攻击如ghost plane; dos

针对ADS-B的攻击种类繁多，统分为主动攻击和被动攻击。本课题主要解决ADS-B的数据链隐私保护即保护ADS-B消息的机密性，防止ADS-B消息被无关人员窃听的被动攻击防御，故在本文中不考虑ADS-B假消息的注入及泛洪攻击等主动攻击的情形。

3.1.3.3 遵守航空法不隐藏轨迹

出于航行安全的考虑，航空安全法不建议任何通过ADS-B通信的飞行器隐藏轨迹，但是隐藏飞行器的唯一标识符并不会影响飞行安全且可以达到保护飞行器的隐私的效果，因此本课题通过加密ICAO24(通用的航空的唯一标识)，ADS-B消息的一部分，且加密ICAO24并不会改变ADS-B消息中的飞行轨迹也不会影响ADS-B消息的解码而且可以达到保护通用航空隐私的目的。

3.2 系统流程图

第一步：从OPenSky的网站上下载到包含ADS-B的原始消息的.avro文件。.avro文件是apache下的一个数据序列化系统的文件，它拥有丰富的数据结构类型。本次用的的.avro文件的数据结构如下：

```

01 {
02 "name": "ModeSEncodedMessage",
03 "type": "record",
04 "namespace": "org.opensky.avro.v2",
05 "fields": [
06 {"name": "sensorType", "type": "string"},
07 {"name": "sensorLatitude", "type": ["double", "null"]},
08 {"name": "sensorLongitude", "type": ["double", "null"]},
09 {"name": "sensorAltitude", "type": ["double", "null"]},
10 {"name": "timeAtServer", "type": "double"},
11 {"name": "timeAtSensor", "type": ["double", "null"]},
12 {"name": "timestamp", "type": ["double", "null"]},
13 {"name": "rawMessage", "type": "string"},
14 {"name": "sensorSerialNumber", "type": "int"},
15 {"name": "RSSIPacket", "type": ["double", "null"]},
16 {"name": "RSSIPreamble", "type": ["double", "null"]},
17 {"name": "SNR", "type": ["double", "null"]},
18 {"name": "confidence", "type": ["double", "null"]}}
19 ]
20 }
```

第二步：对原始的.avro文件按时间顺序进行排序，因为ADS-B消息解码位置信息时需要用到两条连续的ADS-B消息，所以要先对消息进行排序，否则无法对ADS-B消息的位置信息进行解码。

第三步：整个.avro包包含了大量的ADS-B消息，故也包含了若干条轨迹信息，本课题主要研究ADS-B消息的加解密，故只需从若干条轨迹中抽出其中的一两条来进行试验分析即可。

第四步：对选取的一条轨迹的ADS-B消息在ADS-B OUT 中用FFX算法进行加密。

第五步：把加密过的若干条在公益轨迹上的ADS-B消息发送给ADS-B IN，本课题中没有模拟发送过程。

第六步：在ADS-B IN1中对加密过的ADS-B消息用FFX算法进行解密。

第七步：在ADS-B IN1中对解密的ADS-B消息进行解码。

第八步：把解码后的信息转换为.kml文件并在Google Earth上显示与未经加密的轨迹进行对比以验证加解密效果。.kml文件是google公司创建的一种地理信息文件，用于描述和保存时间、地点、经纬度、海拔高度等轨迹参数。可用Google Earth打开。

第九步：在另一个ADS-B IN，即ADS-B IN2中直接对接收到的经过加密的ADS-B消息进行解码并转换为.kml文件在Google Earth上显示，与原始的未经过加密的ADS-B轨迹信息和经过解密的ADS-B轨迹信息进行对比以验证试验结果。

总的流程概括如下：

`sample.avro → sorted.avro → select.avro → FFXEncrypt/Decrypt(ICAO24) → decoder → select.kml`

3.3 模块设计

3.3.1 加解密模块

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

- (1)字母表 $Chars = \{char_0, char_1, \dots, char_{(radix-1)}\}$ 及其基数 radix；
- (2)确定非平衡Feistel网络类型；

(3)消息空间 $Char^n$ 中的字符串长度（字符个数）n；

(4)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t，输出为密文字符串y，字符串x和y同属于消息空间 $Char^n$ ，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 $Chars' = \{0, 1, 2, \dots, radix - 1\}$ 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) $\sum c_i radix^{(n-i)} = (\sum a_i radix^{(n-i)} + \sum b_i radix^{(n-i)}) \bmod radix$ ；

(3)连接L'与R得到输出 $L' \parallel R$ ，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模型的工作原理如下图所示。

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

FFX算法的三个主要步骤：

step1:赋值

把加密原文长度赋值给n,把分段长度赋值给l,把加密轮数赋值给r,把两个分段分别赋值给A和B。

step2:Feistel转化

把两段消息放入Feistel轮变化中经过r轮转化后输出。

step3:合并

把从Feisel轮变化中输出的两段消息合并。

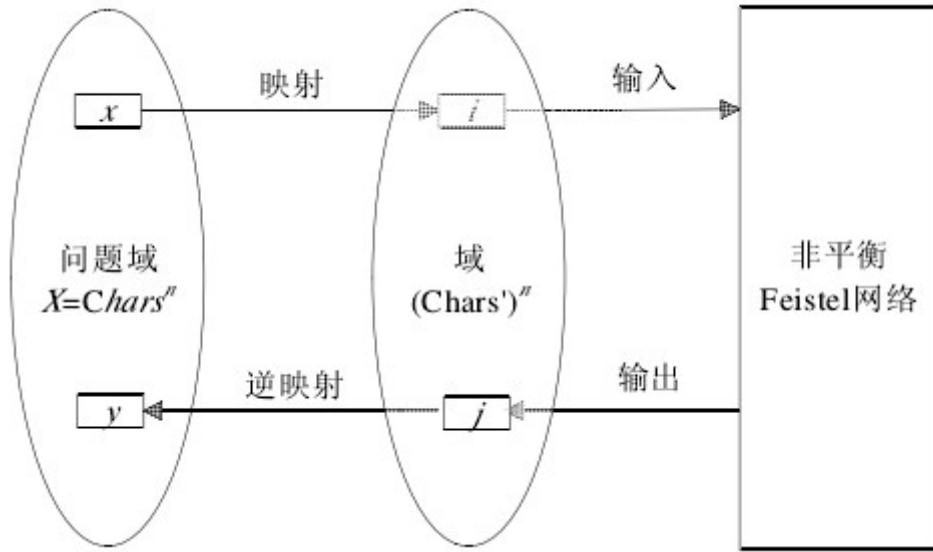


图 3-1 FFX模型

3.3.2 解码模块

step1:建立从.avro文件中提取所需的ADS-B rawMessage各区域的类;
 step2:先判断downlink format(DF)是否为17, 是则继续step2;
 step3:再判断第33-第37位的Type Code(TC)的值根据TC的不同判断ADS-B rawMessage包含的是什么信息, 再针对rawMessage的类型对rawMessage分别进行不同的解码;

step4:把解码出来的ADS-B信息放在.kml文件的对应的位置。

下面对解码的最重要部分位置解码进行详细的讲解, ADS-B位置解码分为广域位置解码（全球解码）和局域位置解码（本地解码）

3.3.2.1 广域位置解码（全球解码）

广域解码方式, 全球解码需要两组编码数据来完成解码, 即一组奇编码数据和一组偶编码数据, 与局域解码方式相比, 不需要提供基准位置信息, 任何位置都可以解码。采用接收到的偶编码（由YZ0, XZ0表示）和奇编码（由YZ1, XZ1表示）的两个位置消息, 共同来产生全球位置的纬度Rlat 和经度Rlon。对于空中位置, 偶编码和奇编码位置消息之间的时间间隔不超过 10s, 这是由 3 n mile 的最大允许间距决定的。

解码步骤主要分为以下几个部分：

(1) 将经纬度的分区的数量 $Dlat_i$ 计算出来。

对于空中位置和地面位置 CPR 分别计算分区的数量，CPR 奇编码计算的为 $Dlat_{i1}$ ，CPR 偶编码计算的为 $Dlat_{i0}$ 。

$$\text{空中位置: } Dlat_{i1} = \frac{360}{4 \times NZ - i} = \begin{cases} 6.00, & \text{偶编码} \\ 6.10, & \text{奇编码} \end{cases}$$

$$\text{地面位置: } Dlat_{i0} = \frac{90}{4 \times NZ - i} = \begin{cases} 0, & \text{偶编码} \\ 1, & \text{奇编码} \end{cases}$$

(2) 计算纬度区域索引值 j 。

$$j = \text{floor}\left(\frac{59*YZ_0 - 60*YZ_1}{2^{17}} + \frac{1}{2}\right)$$

当 $j \geq 0$ 时， j = 偶纬度Zone的编号减 60， j = 奇纬度Zone的编号减 59；

当 $j < 0$ or $j = 0$ 时， j = 偶纬度Zone的编号， j = 奇纬度Zone的编号。

(3) 计算纬度值 $Rlat_i$ 。

$$Rlat_i = Dlat_i \left(\text{MOD}(j, 60 - i) + \frac{YZ_i}{2^{17}} \right)$$

若求出的纬度绝对值大于 90° ，则应减去 360° 。因为纬度的取值范围为 $-90^\circ + 90^\circ$ 。

(4) 如果 $\text{NL}(Rlat_0)$ 不等于 $\text{NL}(Rlat_1)$ ，则表示两次收到的位置信息不再同一个纬度区域内，这种情况说明飞行器在跨越纬度区域，不能解码，需等待接收到位置信息中经纬度区域值相等时解码。如果相等，则计算 $Dlon_i$ 的值。 $Dlon_i = \frac{360}{n_i}$, n_i 为 $[\text{NL}(Rlat_i) - 1]$ 中大的那个数。

(5) 然后计算经度索引值 m 。

$$m = \text{floor}\left(\frac{XZ_0 * (NL - 1) - XZ_1 * NL}{2^{17}} + \frac{1}{2}\right) NL = \text{NL}(Rlat_i)$$

(6) 最后计算经度 $Rlon_i$ 。

$$Rlon_i = Dlon_i \left(\frac{\text{Mod}(m, ni) + XZ_i}{2^{17}} \right) n_i = \max([\text{NL}(Rlat_i) - 1], 1)$$

若求出的经度绝对值大于 180° ，则应减去 360° 。又因为无论是广域位置解码还是局域位置解码都仍然存在两种情况，即：空中位置和地面位置。但上述算法两种对于两种类型解码均适用。因为经度的取值范围为 $-180^\circ + 180^\circ$ 。

3.3.2.2 局域位置解码（本地解码）

对于某个参考点（参考点可以是由全球解码确认的以前跟踪的某个位置，也可以是本机的位置，假定其纬度和经度分别为 $lats$ 、 $lons$ ），CPR 算法将通过解码

获得本地明确的地理位置 (纬度 $Rlat_i$ 和经度 $Rlon_i$)。对于空中位置, 该位置在真实位置的 180 n mile 内; 对于地面位置, 该位置在真实位置的 45 n mile(180/4 n mile)以内。

解码步骤:

(1)首先计算纬度 Zone 的尺寸 $Dlat_i$ 。

$$\begin{aligned} \text{空中位置: } Dlat_i &= \frac{360}{4 \times NZ - i} = \begin{cases} 6.00, & \text{偶编码} \\ 6.10, & \text{奇编码} \end{cases} \\ \text{地面位置: } Dlat_i &= \frac{90}{4 \times NZ - i} = \begin{cases} 0, & \text{偶编码} \\ 1, & \text{奇编码} \end{cases} \end{aligned}$$

(2)计算纬度索引 j。

$j = floor(\frac{lats}{Dlat_i}) + floor[\frac{1}{2} + \frac{MOD(lats, Dlat_i)}{Dlat_i} - \frac{YZ_i}{2^{17}}]$ 纬度索引 j 的值与参考点的纬度 lats 的取值有关。

(3)解码纬度位置 $Rlat_i$ 。

$Rlat_i = Dlat_i \times (j + \frac{YZ_i}{2^{17}})$ 本地解码后的纬度位置 $Rlat_i$ 与参考点的纬度 lats 的取值有关。

(4)由 $Rlat_i$ 确定东西向经度 Zone 的尺寸 $Dlon_i$ 。

$$\begin{aligned} \text{空中位置: } Dlat_i &= \begin{cases} \frac{360}{NL(Rlat_i) - i}, & NL(Rlat_i) - i > 0 \\ 360, & NL(Rlat_i) - i = 0 \end{cases} \\ \text{地面位置: } Dlat_i &= \begin{cases} \frac{90}{NL(Rlat_i) - i}, & NL(Rlat_i) - i > 0 \\ 90, & NL(Rlat_i) - i = 0 \end{cases} \end{aligned}$$

(5)采用参考点的经度 lons、 $Dlon_i$ 和 XZi 计算经度索引 m。

$m = floor(\frac{lons}{Dlon_i}) + floor[\frac{1}{2} + \frac{MOD(lons, Dlon_i)}{Dlon_i} - \frac{XZ_i}{2^{17}}]$; 经度索引 m 的值与参考点的经度 lons 的取值有关。

(6)解码经度位置 $Rlon_i$ 。

$Rlon_i = Dlon_i \times (m + \frac{XZ_i}{2^{17}})$; 本地解码后的经度位置 $Rlon_i$ 与参考点的经度 lons 的取值有关。

3.3.3 可视化文件类型转换模块

step1: 创建装换类中的飞机消息格式内部类;
step2: 在转换的类中编写创建.kml文件的函数且将飞机消息格式内部类作为函数的输入参数;
step3: 通过添加函数把解码出来的信息对应放在创建.kml文件的函数的参数的飞机消息格式内部类的各个部分;

3.4 算法编程

3.4.1 加解密部分

3.4.1.1 F函数部分代码

```
01 def F(self, n, T, i, B):
02     if T == 0:
03         t = 0
04     else:
05         t = len(T)
06
07     beta = math.ceil(n / 2.0)
08     b = int(math.ceil(math.ceil(beta * math.log(self._radix, 2))
/ 8.0))
09     d = 4 * int(math.ceil(b / 4.0))
10
11     if self.isEven(i):
12         m = int(math.floor(n / 2.0))
13     else:
14         m = int(math.ceil(n / 2.0))
15
16     if not self._P.get(n):
17         P = '\x01' # vers
18         P += '\x02' # method
19         P += '\x01' # addition
```

```
20      P += long_to_bytes(self._radix, 3)
21      P += '\x0a' # always ten
22      P += long_to_bytes(self.split(n)%256, 1)
23      P += long_to_bytes(n, 4)
24      P += long_to_bytes(t, 4)
25      self._P[n] = P
26
27      if T == 0:
28          Q = ''
29      else:
30          Q = str(T)
31
32      Q += '\x00' * (((-1 * t) - b - 1) % 16)
33      Q += long_to_bytes(i, blocksize=1)
34
35      _B_as_bytes = long_to_bytes(B)
36      Q += '\x00' * (b - len(_B_as_bytes))
37      Q += _B_as_bytes[-b:]
38
39      _cbc = AES.new(self._K, AES.MODE_CBC, '\x00' * 16)
40
41      assert len(self._P[n]) % 16 == 0
42      assert len(Q) % 16 == 0
43
44      Y = _cbc.encrypt(self._P[n] + Q)[-16:]
45
46      i = 1
47      TMP = Y
48      while len(TMP) < (d + 4):
49          left = FFXInteger(bytes_to_long(Y), radix=self._radix,
blocksize=32)
50          right = FFXInteger(str(i), radix=10, blocksize=16)
```

```

51      X = self.add(left, right)
52      TMP += self._ecb.encrypt(X.to_bytes(16))
53      i += 1
54
55      y = bytes_to_long(TMP[:d + 4])
56      z = y % (self._radix ** m)
57
58      return z

```

F函数的步骤是首先对各个变量进行赋值: $t = |T|$; $\text{beta} = \lceil n/2 \rceil$; $b = \lceil \lceil \text{beta} \log_2(\text{radix}) \rceil / 8 \rceil$; $d = 4 * \lceil b/4 \rceil$; 如果i是偶数: $m = \lfloor n/2 \rfloor$; 如果i是奇数: $m = \lceil n/2 \rceil$; P是vers、method、addition、radix、rnds(n)、split(n)、n、t的连接; Q是T、0、i、B的连接; $Y = \text{CBC-MAC}_K(P \parallel Q)$; $Y = \text{first } d + 4 \text{ bytes of}(Y \parallel \text{AES}_K(Y + [1]^{16}) \parallel \text{AES}_K(Y + [2]_K^{16}(Y + [3]^{16}) \dots))$; $y = \text{NUM}_2(Y)$; $z = y \bmod \text{radix}^m$; $z = \text{STR}_{\text{radix}}^m(z)$; F函数的作用是对输入的其中的一半消息A或者B进行基于AES加密和参数连接的变化, 且输出长度与输入长度一致的输出值, 让同样长度的输出值进入下一轮迭代, 每一轮的输出的长度都相等。

3.4.1.2 加密函数部分代码

```

01 def encrypt(self, T, X):
02     retval = ''
03
04     n = len(X)
05     l = self.split(n)
06     r = 10 #self.rnds(n)
07     A = X[:l]
08     B = X[l:]
09     for i in range(r):
10         C = self.add(A, self.F(n, T, i, B))
11         A = B
12         B = C
13
14     retval = FFXInteger(str(A) + str(B), radix=self._radix)

```

```
15
16     return retval
```

为加密过程中需要用到的各个参数赋值： n = X的长度; l = 分开的块的块大小， 这里为输入原文的一半; A、 B分别为输入原文的前半段和后半段; 然后对A、 B进行r轮的迭代变换， 最后再把经过r轮变换的A、 B连接在一起组成和输入原文长度一致的输出值。

3.4.1.3 解密函数部分代码

```
01 def decrypt(self, T, Y):
02     retval = ''
03
04     n = len(Y)
05     l = self.split(n)
06     r = 10 #self.rnds(n)
07
08     A = Y[:l]
09     B = Y[l:]
10    for i in range(r - 1, -1, -1):
11        C = B
12        B = A
13        A = self.sub(C, self.F(n, T, i, B))
14
15    retval = FFXInteger(str(A) + str(B), radix=self._radix)
16
17    return retval
```

同理对n、 l、 r三个变量赋值,在解密中， 与加密不同的是A这时被赋值为密文的后半段， B被赋值为前半段;循环也是与加密中的循环流程相反.

3.4.1.4 把加密部分的代码植入java的解码和可视化系统中

```
01 public class ModeSEncodedMessage extends org.apache.avro.specific.SpecificRecord
02     implements org.apache.avro.specific.SpecificRecord {
03
04     @Deprecated public java.lang.CharSequence sensorType;
05     @Deprecated public java.lang.Double sensorLatitude;
06     @Deprecated public java.lang.Double sensorLongitude;
07     @Deprecated public java.lang.Double sensorAltitude;
08     @Deprecated public double timeAtServer;
09     @Deprecated public java.lang.Double timeAtSensor;
10     @Deprecated public java.lang.Double timestamp;
11     @Deprecated public java.lang.CharSequence rawMessage;
12     @Deprecated public int sensorSerialNumber;
13     @Deprecated public java.lang.Double RSSIPacket;
14     @Deprecated public java.lang.Double RSSIPreamble;
15     @Deprecated public java.lang.Double SNR;
16     @Deprecated public java.lang.Double confidence;
17
18     /**
19      * Default constructor. Note that this does not initialize fields
20      * to their default values from the schema. If that is desired
then
21      * one should use <code>newBuilder()</code>.
22      */
23     public ModeSEncodedMessage() {}
24     private CharSequence FFXEncrypt(CharSequence rawMessage) {
25         System.out.println("unencrypted raw message is " + rawMessage);
26         try {
27             Process proc = Runtime.getRuntime().exec("python \"C:\\\\Users\\\\Serlina\\\\out\\\\libffx\\\\example.py\" " + rawMessage);
```

```
28     BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(
29         rawMessage = bufferedReader.readLine();
30         proc.waitFor();
31     }
32     catch (Exception e){
33         e.printStackTrace();
34     }
35     System.err.println("encrypted raw message is " + rawMessage);
36     return rawMessage;
37 }
38 private CharSequence FFXDecrypt(CharSequence encryptedMessage){
39     System.err.println("encrypted raw message is " + encryptedMessage);
40     try {
41         Process proc = Runtime.getRuntime().exec("python \"C:\\\\Users\\\\Serlina\\\\in\\\\libffx\\\\example.py\" " + encryptedMessage);
42         BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(
43             rawMessage = bufferedReader.readLine();
44             proc.waitFor();
45         }
46         catch (Exception e){
47             e.printStackTrace();
48         }
49         System.err.println("decrypted raw message is " + rawMessage);
50         return rawMessage;
51     }
52 /**
53 * All-args constructor.
54 */
55 public ModeSEncodedMessage(java.lang.CharSequence sensorType,
java.lang.Double sensorLatitude, java.lang.Double sensorLongitude, java.lang.Double
sensorAltitude, java.lang.Double timeAtServer, java.lang.Double timeAtSensor,
java.lang.Double timestamp, java.lang.CharSequence rawMessage, java.lang.Integer
```

```
sensorSerialNumber, java.lang.Double RSSIPacket, java.lang.Double RSSIPreamble,
java.lang.Double SNR, java.lang.Double confidence) {
    56     this.sensorType = sensorType;
    57     this.sensorLatitude = sensorLatitude;
    58     this.sensorLongitude = sensorLongitude;
    59     this.sensorAltitude = sensorAltitude;
    60     this.timeAtServer = timeAtServer;
    61     this.timeAtSensor = timeAtSensor;
    62     this.timestamp = timestamp;
    63     this.rawMessage = rawMessage;
    64     this.sensorSerialNumber = sensorSerialNumber;
    65     this.RSSIPacket = RSSIPacket;
    66     this.RSSIPreamble = RSSIPreamble;
    67     this.SNR = SNR;
    68     this.confidence = confidence;
    69 }
    70
    71     public org.apache.avro.Schema getSchema() { return SCHEMA$; }
    72     // Used by DatumWriter. Applications should not call.
    73     public java.lang.Object get(int field$) {
    74         switch (field$) {
    75             case 0: return sensorType;
    76             case 1: return sensorLatitude;
    77             case 2: return sensorLongitude;
    78             case 3: return sensorAltitude;
    79             case 4: return timeAtServer;
    80             case 5: return timeAtSensor;
    81             case 6: return timestamp;
    82             case 7: return rawMessage;
    83             case 8: return sensorSerialNumber;
    84             case 9: return RSSIPacket;
    85             case 10: return RSSIPreamble;
```

```
86     case 11: return SNR;
87     case 12: return confidence;
88     default: throw new org.apache.avro.AvroRuntimeException("Bad
index");
89   }
90 }
91 // Used by DatumReader. Applications should not call.
92 @SuppressWarnings(value="unchecked")
93 public void put(int field$, java.lang.Object value$) {
94   switch (field$) {
95     case 0: sensorType = (java.lang.CharSequence)value$; break;
96     case 1: sensorLatitude = (java.lang.Double)value$; break;
97     case 2: sensorLongitude = (java.lang.Double)value$; break;
98     case 3: sensorAltitude = (java.lang.Double)value$; break;
99     case 4: timeAtServer = (java.lang.Double)value$; break;
100    case 5: timeAtSensor = (java.lang.Double)value$; break;
101    case 6: timestamp = (java.lang.Double)value$; break;
102    case 7: rawMessage = (java.lang.CharSequence)value$; break;
103    case 8: sensorSerialNumber = (java.lang.Integer)value$; break;
104    case 9: RSSIPacket = (java.lang.Double)value$; break;
105    case 10: RSSIPreamble = (java.lang.Double)value$; break;
106    case 11: SNR = (java.lang.Double)value$; break;
107    case 12: confidence = (java.lang.Double)value$; break;
108    default: throw new org.apache.avro.AvroRuntimeException("Bad
index");
109  }
110 }
111
112 /**
113  * Gets the value of the 'sensorType' field.
114 */
115 public java.lang.CharSequence getSensorType() {
```

```
116     return sensorType;
117 }
118
119 /**
120  * Sets the value of the 'sensorType' field.
121  * @param value the value to set.
122 */
123 public void setSensorType(java.lang.CharSequence value) {
124     this.sensorType = value;
125 }
126
127 ...
128 ...
129 ...
130
131 /**
132  * Gets the value of the 'rawMessage' field.
133 */
134 public java.lang.CharSequence getRawMessage() {
135     return FFXEncrypt(rawMessage);
136 }
137
138 /**
139  * Sets the value of the 'rawMessage' field.
140  * @param value the value to set.
141 */
142 public void setRawMessage(java.lang.CharSequence value) {
143     this.rawMessage = value;
144 }
145
146 ...
147 ...
```

```
148 ...
149
150 /**
151 * Gets the value of the 'confidence' field.
152 */
153 public java.lang.Double getConfidence() {
154     return confidence;
155 }
156
157 /**
158 * Sets the value of the 'confidence' field.
159 * @param value the value to set.
160 */
161 public void setConfidence(java.lang.Double value) {
162     this.confidence = value;
163 }
164
165 /** Creates a new ModeSEncodedMessage RecordBuilder */
166 public static org.opensky.example.ModeSEncodedMessage.Builder
newBuilder() {
167     return new org.opensky.example.ModeSEncodedMessage.Builder();
168 }
169
170 /** Creates a new ModeSEncodedMessage RecordBuilder by copying
an existing Builder */
171 public static org.opensky.example.ModeSEncodedMessage.Builder
newBuilder(org.opensky.example.ModeSEncodedMessage.Builder other) {
172     return new org.opensky.example.ModeSEncodedMessage.Builder(other);
173 }
174
175 /** Creates a new ModeSEncodedMessage RecordBuilder by copying
an existing ModeSEncodedMessage instance */
```

```
176     public static org.opensky.example.ModeSEncodedMessage.Builder  
newBuilder(org.opensky.example.ModeSEncodedMessage other) {  
177         return new org.opensky.example.ModeSEncodedMessage.Builder(other);  
178     }  
179  
180     /**  
181      * RecordBuilder for ModeSEncodedMessage instances.  
182      */  
183     public static class Builder extends org.apache.avro.specific.SpecificReco  
184         implements org.apache.avro.data.RecordBuilder<ModeSEncodedMessage>  
{  
185  
186         private java.lang.CharSequence sensorType;  
187         private java.lang.Double sensorLatitude;  
188         private java.lang.Double sensorLongitude;  
189         private java.lang.Double sensorAltitude;  
190         private double timeAtServer;  
191         private java.lang.Double timeAtSensor;  
192         private java.lang.Double timestamp;  
193         private java.lang.CharSequence rawMessage;  
194         private int sensorSerialNumber;  
195         private java.lang.Double RSSIPacket;  
196         private java.lang.Double RSSIPreamble;  
197         private java.lang.Double SNR;  
198         private java.lang.Double confidence;  
199  
200         /** Creates a new Builder */  
201         private Builder() {  
202             super(org.opensky.example.ModeSEncodedMessage.SCHEMA$);  
203         }  
204  
205         /** Creates a Builder by copying an existing Builder */
```

```
206     private Builder(org.opensky.example.ModeSEncodedMessage.Builder  
other) {  
207         super(other);  
208         if (isValidValue(fields()[0], other.sensorType)) {  
209             this.sensorType = data().deepCopy(fields()[0].schema(),  
other.sensorType);  
210             fieldSetFlags()[0] = true;  
211         }  
212     }  
213     ...  
214     ...  
215     ...  
216     if (isValidValue(fields()[7], other.rawMessage)) {  
217         this.rawMessage = data().deepCopy(fields()[7].schema(),  
other.rawMessage);  
218         fieldSetFlags()[7] = true;  
219     }  
220     ...  
221     ...  
222     ...  
223     if (isValidValue(fields()[12], other.confidence)) {  
224         this.confidence = data().deepCopy(fields()[12].schema(),  
other.confidence);  
225         fieldSetFlags()[12] = true;  
226     }  
227 }  
228  
229     /** Creates a Builder by copying an existing ModeSEncodedMessage  
instance */  
230     private Builder(org.opensky.example.ModeSEncodedMessage other)  
{  
231         super(org.opensky.example.ModeSEncodedMessage.SCHEMA$);
```

```
232     if (isValidValue(fields()[0], other.sensorType)) {
233         this.sensorType = data().deepCopy(fields()[0].schema(),
234                                         other.sensorType);
235         fieldSetFlags()[0] = true;
236     }
237 ...
238 ...
239 ...
240     if (isValidValue(fields()[7], other.rawMessage)) {
241         this.rawMessage = data().deepCopy(fields()[7].schema(),
242                                         other.rawMessage);
243         fieldSetFlags()[7] = true;
244     }
245 ...
246 ...
247     if (isValidValue(fields()[12], other.confidence)) {
248         this.confidence = data().deepCopy(fields()[12].schema(),
249                                         other.confidence);
250         fieldSetFlags()[12] = true;
251     }
252
253     /** Gets the value of the 'sensorType' field */
254     public java.lang.CharSequence getSensorType() {
255         return sensorType;
256     }
257
258     /** Sets the value of the 'sensorType' field */
259     public org.opensky.example.ModeSEncodedMessage.Builder setSensorType(ja
value) {
```

```
260     validate(fields()[0], value);
261     this.sensorType = value;
262     fieldSetFlags()[0] = true;
263     return this;
264 }
265
266 /** Checks whether the 'sensorType' field has been set */
267 public boolean hasSensorType() {
268     return fieldSetFlags()[0];
269 }
270
271 /** Clears the value of the 'sensorType' field */
272 public org.opensky.example.ModeSEncodedMessage.Builder clearSensorType()
{
273     sensorType = null;
274     fieldSetFlags()[0] = false;
275     return this;
276 }
277
278 ...
279 ...
280 ...
281
282 /** Gets the value of the 'rawMessage' field */
283 public java.lang.CharSequence getRawMessage() {
284     return rawMessage;
285 }
286
287 /** Sets the value of the 'rawMessage' field */
288 public org.opensky.example.ModeSEncodedMessage.Builder setRawMessage(ja
value) {
289     validate(fields()[7], value);
```

```
290     this.rawMessage = value;
291     fieldSetFlags()[7] = true;
292     return this;
293 }
294
295     /** Checks whether the 'rawMessage' field has been set */
296     public boolean hasRawMessage() {
297         return fieldSetFlags()[7];
298     }
299
300     /** Clears the value of the 'rawMessage' field */
301     public org.opensky.example.ModeSEncodedMessage.Builder clearRawMessage()
{
302         rawMessage = null;
303         fieldSetFlags()[7] = false;
304         return this;
305     }
306
307 ...
308 ...
309 ...
310
311     /** Gets the value of the 'confidence' field */
312     public java.lang.Double getConfidence() {
313         return confidence;
314     }
315
316     /** Sets the value of the 'confidence' field */
317     public org.opensky.example.ModeSEncodedMessage.Builder setConfidence(ja
value) {
318         validate(fields()[12], value);
319         this.confidence = value;
```

```
320     fieldSetFlags()[12] = true;
321     return this;
322 }
323
324 /** Checks whether the 'confidence' field has been set */
325 public boolean hasConfidence() {
326     return fieldSetFlags()[12];
327 }
328
329 /** Clears the value of the 'confidence' field */
330 public org.opensky.example.ModeSEncodedMessage.Builder clearConfidence()
{
331     confidence = null;
332     fieldSetFlags()[12] = false;
333     return this;
334 }
335
336 @Override
337 public ModeSEncodedMessage build() {
338     try {
339         ModeSEncodedMessage record = new ModeSEncodedMessage();
340         record.sensorType = fieldSetFlags()[0] ? this.sensorType
: (java.lang.CharSequence) defaultValue(fields()[0]);
341         record.sensorLatitude = fieldSetFlags()[1] ? this.sensorLatitude
: (java.lang.Double) defaultValue(fields()[1]);
342         record.sensorLongitude = fieldSetFlags()[2] ? this.sensorLongitude
: (java.lang.Double) defaultValue(fields()[2]);
343         record.sensorAltitude = fieldSetFlags()[3] ? this.sensorAltitude
: (java.lang.Double) defaultValue(fields()[3]);
344         record.timeAtServer = fieldSetFlags()[4] ? this.timeAtServer
: (java.lang.Double) defaultValue(fields()[4]);

```

```
345         record.timeAtSensor = fieldSetFlags()[5] ? this.timeAtSensor
: (java.lang.Double) defaultValue(fields()[5]);
346         record.timestamp = fieldSetFlags()[6] ? this.timestamp
: (java.lang.Double) defaultValue(fields()[6]);
347         record.rawMessage = fieldSetFlags()[7] ? this.rawMessage
: (java.lang.CharSequence) defaultValue(fields()[7]);
348         record.sensorSerialNumber = fieldSetFlags()[8] ? this.sensorSerialNumber
: (java.lang.Integer) defaultValue(fields()[8]);
349         record.RSSIPacket = fieldSetFlags()[9] ? this.RSSIPacket
: (java.lang.Double) defaultValue(fields()[9]);
350         record.RSSIPreamble = fieldSetFlags()[10] ? this.RSSIPreamble
: (java.lang.Double) defaultValue(fields()[10]);
351         record.SNR = fieldSetFlags()[11] ? this.SNR : (java.lang.Double)
defaultValue(fields()[11]);
352         record.confidence = fieldSetFlags()[12] ? this.confidence
: (java.lang.Double) defaultValue(fields()[12]);
353         return record;
354     } catch (Exception e) {
355         throw new org.apache.avro.AvroRuntimeException(e);
356     }
357 }
358 }
359 }
```

3.4.2 解码部分

step1: 确定输入的ADS-B消息是奇数帧还是偶数帧，只有连续的两个ADS-B消息一个为奇数帧一个为偶数帧才利用这两条ADS-B消息进行位置解码，这就是为什么.avro文件在用来解码前要先利用OpenSky提供的函数对.avro文件中的数据进行排序的原因。

step2: 再判断该ADS-B消息适用于全球解码或者本地解码，根据其特征对其使用不同的解码方式。

```

01 public class Decoder {
02
03     public static ModeSReply genericDecoder (String raw_message) throws
04         BadFormatException, UnspecifiedFormatError {
05
06     ModeSReply modes = new ModeSReply(raw_message);
07
08     if (modes.getDownlinkFormat() == 4) {
09         return new AltitudeReply(modes);
10     }
11
12     else if (modes.getDownlinkFormat() == 5) {
13         return new IdentifyReply(modes);
14     }
15     else if (modes.getDownlinkFormat() == 11) {
16         return new AllCallReply(modes);
17     }
18     else if (modes.getDownlinkFormat() == 17 || modes.getDownlinkFormat()
19 == 18) {
20
21         ExtendedSquitter es1090 = new ExtendedSquitter(modes);
22
23         // what kind of extended squitter?
24         byte ftc = es1090.getFormatTypeCode();
25
26         if (ftc >= 1 && ftc <= 4) // identification message
27             return new IdentificationMsg(es1090);
28
29     }
30
31     return null;
32 }
```

```
23
24     if (ftc >= 5 && ftc <= 8) // surface position message
25         return new SurfacePositionMsg(es1090);
26
27     if ((ftc >= 9 && ftc <= 18) || (ftc >= 20 && ftc <= 22))
// airborne position message
28         return new AirbornePositionMsg(es1090);
29
30     if (ftc == 19) { // possible velocity message, check subtype
31         int subtype = es1090.getMessage()[0]&0x7;
32
33         if (subtype == 1 || subtype == 2) // velocity over ground
34             return new VelocityOverGroundMsg(es1090);
35         else if (subtype == 3 || subtype == 4) // airspeed & heading
36             return new AirspeedHeadingMsg(es1090);
37     }
38
39     if (ftc == 28) { // aircraft status message, check subtype
40         int subtype = es1090.getMessage()[0]&0x7;
41
42         if (subtype == 1) // emergency/priority status
43             return new EmergencyOrPriorityStatusMsg(es1090);
44         if (subtype == 2) // TCAS resolution advisory report
45             return new TCASResolutionAdvisoryMsg(es1090);
46     }
47
48     if (ftc == 31) { // operational status message
49         int subtype = es1090.getMessage()[0]&0x7;
50
51         if (subtype == 0 || subtype == 1) // airborne or surface?
52             return new OperationalStatusMsg(es1090);
53     }
```

```

54
55     return es1090; // unknown extended squitter
56 }
57 else if (modes.getDownlinkFormat() == 20) {
58     return new CommBALtitudeReply(modes);
59 }
60 else if (modes.getDownlinkFormat() == 21) {
61     return new CommBIDeidentifyReply(modes);
62 }
63
64 return modes; // unknown mode s reply
65 }
66
67 }

```

3.4.2.1 全球解码

```

01 public Position getGlobalPosition(SurfacePositionMsg other) throws
MissingInformationException,
02 PositionStraddleError, BadFormatException {
03     if (!tools.areEqual(other.getIcao24(), getIcao24()))
04         throw new IllegalArgumentException(
05             String.format("Transmitter of other message (%s) not equal
to this (%s).",
06             tools.toHexString(other.getIcao24()), tools.toHexString(this.getIcao24())
07
08     if (other.isOddFormat() == this.isOddFormat())
09         throw new BadFormatException("Expected "+(isOddFormat()?"even":"odd")+
message format.", other.toString());
10
11     if (!horizontal_position_available)
12         throw new MissingInformationException("No position information
available!");

```

```
13 if (!other.hasPosition())
14     throw new MissingInformationException("Other message has no position
information.");
15
16 SurfacePositionMsg even = isOddFormat()?other:this;
17 SurfacePositionMsg odd = isOddFormat()?this:other;
18
19 // Helper for latitude single(Number of zones NZ is set to 15)
20 double Dlat0 = 90.0/60.0;
21 double Dlat1 = 90.0/59.0;
22
23 // latitude index
24 double j = Math.floor((59.0*even.getCPREncodedLatitude())-60.0*odd.getCPREn
25
26 // global latitudes
27 double Rlat0 = Dlat0 * (mod(j,60)+even.getCPREncodedLatitude())/((double)(1<
28 double Rlat1 = Dlat1 * (mod(j,59)+odd.getCPREncodedLatitude())/((double)(1<
29
30 // Southern hemisphere?
31 if (Rlat0 >= 270 && Rlat0 <= 360) Rlat0 -= 360;
32 if (Rlat1 >= 270 && Rlat1 <= 360) Rlat1 -= 360;
33
34 // Northern hemisphere?
35 if (Rlat0 <= -270 && Rlat0 >= -360) Rlat0 += 360;
36 if (Rlat1 <= -270 && Rlat1 >= -360) Rlat1 += 360;
37
38 // ensure that the number of even longitude zones are equal
39 if (NL(Rlat0) != NL(Rlat1))
40     throw new org.opensky.libadsb.exceptions.PositionStraddleError(
41         "The two given position straddle a transition latitude "+
42         "and cannot be decoded. Wait for positions where they are equal.");
43
```

```
44 // Helper for longitude
45 double Dlon0 = 90.0/Math.max(1.0, NL(Rlat0));
46 double Dlon1 = 90.0/Math.max(1.0, NL(Rlat1)-1);
47
48 // longitude index
49 double NL_helper = NL(isOddFormat()?Rlat1:Rlat0); // assuming
that this is the newer message
50 double m = Math.floor((even.getCPREncodedLongitude()*(NL_helper-1)-odd.get
51
52 // global longitude
53 double Rlon0 = Dlon0 * (mod(m,Math.max(1.0, NL(Rlat0))) + even.getCPREncod
54 double Rlon1 = Dlon1 * (mod(m,Math.max(1.0, NL(Rlat1)-1)) + odd.getCPREncod
55
56 // correct longitude
57 if (Rlon0 < -180 && Rlon0 > -360) Rlon0 += 360;
58 if (Rlon1 < -180 && Rlon1 > -360) Rlon1 += 360;
59 if (Rlon0 > 180 && Rlon0 < 360) Rlon0 -= 360;
60 if (Rlon1 > 180 && Rlon1 < 360) Rlon1 -= 360;
61
62 return new Position(isOddFormat()?Rlon1:Rlon0,
63           isOddFormat()?Rlat1:Rlat0,
64           0.0);
65 }
```

3.4.2.2 本地解码

```
01 public Position getLocalPosition(Position ref) throws MissingInformationException {
02     if (!horizontal_position_available)
03         throw new MissingInformationException("No position information
available!");
04
05     // latitude zone size
06     double Dlat = isOddFormat() ? 90.0/59.0 : 90.0/60.0;
07
08     // latitude zone index
09     double j = Math.floor(ref.getLatitude()/Dlat) + Math.floor(0.5+(mod(ref.g
Dlat))/Dlat-getCPREncodedLatitude()/(((double)(1<<17))));
10
11    // decoded position latitude
12    double Rlat = Dlat*(j+getCPREncodedLatitude()/(((double)(1<<17)));
13
14    // longitude zone size
15    double Dlon = 90.0/Math.max(1.0, NL(Rlat)-(isOddFormat()?1.0:0.0));
16
17    // longitude zone coordinate
18    double m =
19        Math.floor(ref.getLongitude()/Dlon) +
20        Math.floor(0.5+(mod(ref.getLongitude(),Dlon))/Dlon-(float)getCPREncodedL
21
22    // and finally the longitude
23    double Rlon = Dlon * (m + getCPREncodedLongitude()/(((double)(1<<17)));
24
25 // System.out.println("Loc: EncLon: "+getCPREncodedLongitude()+
26 //      " m: "+m+" Dlon: "+Dlon+" Rlon2: "+Rlon2);
27
28    return new Position(Rlon, Rlat, 0.0);
29 }
```

3.4.3 可视化文件类型转换部分

```
01 public class Avro2Kml {  
02     /**  
03      * This class is a container for all information  
04      * about flights that are relevant for the KML  
05      * generation  
06     */  
07     private class Flight {  
08         public String icao24;  
09         public char[] callsign;  
10         public double first; // first message seen  
11         public double last; // last message seen  
12         public List<Coordinate> coords; // ordered list of coordinates  
13         public List<Integer> serials; // flight seen by these sensors  
14         public PositionDecoder dec; // stateful position decoder  
15         boolean contains_unreasonable; // true if there was one position  
with unreasonable flag  
16  
17         public Flight () {  
18             coords = new ArrayList<Coordinate>();  
19             dec = new PositionDecoder();  
20             callsign = new char[0];  
21             contains_unreasonable = false;  
22             serials = new ArrayList<Integer>();  
23         }  
24     }  
25  
26     /**  
27      * Class for generating the kml  
28     */  
29     private class OskyKml {
```

```
30  private Kml kml;
31  private Document doc;
32  private Style style;
33  private SimpleDateFormat date_formatter;
34  private Folder unreasonable;
35  private Folder reasonable;
36  private Folder empty;
37  private int num_flights;
38
39  public OskyKml () {
40      // prepare KML
41      kml = new Kml();
42      doc = kml.createAndSetDocument()
43          .withName("OpenSky Network");
44
45      style = doc.createAndAddStyle()
46          .withId("reasonable");
47      style.createAndSetLineStyle()
48          .withColor("ffffffff")
49          .withColorMode(ColorMode.NORMAL)
50          .withWidth(1);
51
52      style = doc.createAndAddStyle()
53          .withId("unreasonable");
54      style.createAndSetLineStyle()
55          .withColor("ffd5d5ff")
56          .withColorMode(ColorMode.NORMAL)
57          .withWidth(1);
58
59      unreasonable = doc.createAndAddFolder()
60          .withName("Unreasonable Flights");
61      reasonable = doc.createAndAddFolder()
```

```
62     .withName("Reasonable Flights");
63     empty = doc.createAndAddFolder()
64         .withName("No Positions");
65
66     date_formatter = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ssZ");
67 }
68
69 public void addFlight(Flight flight) {
70     Date begin = new Date((long)(flight.first*1000));
71     Date end = new Date((long)(flight.last*1000));
72
73     Folder which;
74     if (flight.coords.size()>0)
75         which = flight.contains_unreasonable ? unreasonable : reasonable;
76     else which = empty;
77
78     String description = "ICAO: "+flight.icao24+"<br />\n"+
79             "Callsign: "+new String(flight.callsign)+"<br />\n"+
80             "First seen: "+begin.toString()+"<br />\n"+
81             "Last seen: "+end.toString()+"<br />\n"+
82             "Seen by serials: "+StringUtils.join(flight.serials, ","));
83
84     Placemark placemark = which.createAndAddPlacemark()
85         .withName(flight.icao24)
86         .withTimePrimitive(new TimeSpan()
87             .withBegin(date_formatter.format(begin))
88             .withEnd(date_formatter.format(end)))
89         .withDescription(description)
90         .withStyleUrl(flight.contains_unreasonable ? "#unreasonable"
91 : "#reasonable");
92     placemark.createAndSetLineString()
```

```
93     .withCoordinates(flight.coords)
94     .withAltitudeMode(AltitudeMode.fromValue(AltitudeMode.ABSOLUTE.value()))
95     .withId(flight.icao24)
96     .withExtrude(false);
97
98     num_flights++;
99 }
100
101 public void writeToFile(File file) {
102     try {
103         kml.marshal(file);
104     } catch (FileNotFoundException e) {
105         System.err.println("Could not write to file '" + file.getAbsolutePath() +
" "+e.toString());
106     }
107 }
108
109 public int getNumberOfFlights() {
110     return num_flights;
111 }
112 }
113
114 public static void main(String[] args) {
115
116     // define command line options
117     Options opts = new Options();
118     opts.addOption("h", "help", false, "print this message" );
119     opts.addOption("0", "npos", false, "do not include flight without
positions" );
120     opts.addOption("i", "icao24", true, "filter by icao 24-bit address
(hex)" );
```

```
121     opts.addOption("s", "start", true, "only messages received after  
this time (unix timestamp)");  
122     opts.addOption("e", "end", true, "only messages received before  
this time (unix timestamp)");  
123     opts.addOption("n", "max-num", true, "max number of flights  
written to KML");  
124  
125     // parse command line options  
126     CommandLineParser parser = new DefaultParser();  
127     CommandLine cmd;  
128     File avro = null, kmlfile = null;  
129     String filter_icao24 = null;  
130     Long filter_max = null;  
131     Double filter_start = null, filter_end = null;  
132     String file = null, out = null;  
133     boolean option_nopos = true;  
134     try {  
135         cmd = parser.parse(opts, args);  
136  
137         // parse arguments  
138         try {  
139             if (cmd.hasOption("i")) filter_icao24 = cmd.getOptionValue("i");  
140             if (cmd.hasOption("s")) filter_start = Double.parseDouble(cmd.getOptionValue("s"));  
141             if (cmd.hasOption("e")) filter_end = Double.parseDouble(cmd.getOptionValue("e"));  
142             if (cmd.hasOption("n")) filter_max = Long.parseLong(cmd.getOptionValue("n"));  
143             if (cmd.hasOption("0")) option_nopos = false;  
144         } catch (NumberFormatException e) {  
145             throw new ParseException("Invalid arguments: "+e.getMessage());  
146         }  
147  
148         // get filename  
149         if (cmd.getArgList().size() != 2)
```

```
150     throw new ParseException("No avro file given or invalid arguments.");
151     file = cmd.getArgList().get(0);
152     out = cmd.getArgList().get(1);
153
154 } catch (ParseException e) {
155     // parsing failed
156     System.err.println(e.getMessage() + "\n");
157     printHelp(opts);
158     System.exit(1);
159 }
160
161 System.out.println("Opening avro file.");
162
163 // check if file exists
164 try {
165     avro = new File(file);
166     if(!avro.exists() || avro.isDirectory() || !avro.canRead())
{
167         throw new FileNotFoundException("Avro file not found or cannot
be read.");
168     }
169
170     kmlfile = new File(out);
171     if(kmlfile.exists() || kmlfile.isDirectory())
172         throw new java.io.IOException("KML is a directory or file
exists.");
173 } catch (FileNotFoundException e) {
174     // avro file not found
175     System.err.println("Error: " + e.getMessage() + "\n");
176     System.exit(1);
177 } catch (IOException e) {
178     // cannot write to KML
```

```
179     System.err.println("Error: "+e.getMessage()+"\n");
180     System.exit(1);
181 }
182
183 DatumReader<ModeSEncodedMessage> datumReader = new SpecificDatumReader<
184     long msgCount = 0, good_pos_cnt = 0, bad_pos_cnt = 0, flights_cnt
= 0, err_pos_cnt = 0;
185     try {
186         DataFileReader<ModeSEncodedMessage> fileReader = new DataFileReader<Mo
datumReader);
187
188         System.err.println("Options are:\n" +
189             "\tfile: "+file+"\n"+
190             "\ticao24: "+filter_icao24+"\n"+
191             "\tstart: "+filter_start+"\n"+
192             "\tend: "+filter_end+"\n"+
193             "\tmax: "+filter_max+"\n");
194
195         // stuff for handling flights
196         ModeSEncodedMessage record = new ModeSEncodedMessage();
197         HashMap<String, Flight> flights = new HashMap<String, Flight>();
198         Flight flight;
199         String icao24;
200
201         // message registers
202         ModeSReply msg;
203         AirbornePositionMsg airpos;
204         SurfacePositionMsg surfacepos;
205         IdentificationMsg ident;
206
207         // KML stuff
208         Avro2Kml a2k = new Avro2Kml();
```

```
209  OskyKml kml = a2k.new OskyKml();
210
211  mainloop:
212  while (fileReader.hasNext()) {
213      msgCount++;
214
215      // get next record from file
216      record = fileReader.next(record);
217
218      // time filters
219      if (filter_start != null && record.getTimeAtServer()<filter_start)
220          continue;
221
222      if (filter_end != null && record.getTimeAtServer()>filter_end)
223          continue;
224
225      // cleanup decoders every 1.000.000 messages to avoid excessive
memory usage
226      // therefore, remove decoders which have not been used for
more than one hour.
227      if (msgCount % 1000000 == 0) {
228          List<String> to_remove = new ArrayList<String>();
229          for (String key : flights.keySet()) {
230              if (flights.get(key).last<record.getTimeAtServer()−3600)
{
231                  to_remove.add(key);
232              }
233          }
234
235          for (String key : to_remove) {
236              // number of flights filter
237              if (filter_max != null && kml.getNumberOfflights()>=filter_max)
```

```
238         break mainloop;
239
240         if (option_nopos | flights.get(key).coords.size() > 0)
241             kml.addFlight(flights.get(key));
242             flights.remove(key);
243     }
244 }
245
246 try {
247     msg = Decoder.genericDecoder(record.getRawMessage().toString());
248 } catch (BadFormatException e) {
249     continue;
250 }
251
252 icao24 = tools.toHexString(msg.getIcao24());
253
254 // icao24 filter
255 if (filter_icao24 != null && !icao24.equals(filter_icao24))
256     continue;
257
258 // select current flight
259 if (flights.containsKey(icao24))
260     flight = flights.get(icao24);
261 else {
262     flight = a2k.new Flight();
263     flight.icao24 = icao24;
264     flight.first = record.getTimeAtServer();
265     flights.put(icao24, flight);
266     ++flights_cnt;
267 }
268
269 flight.last = record.getTimeAtServer();
```

```
270
271     if (!flight.serials.contains(record.getSensorSerialNumber()))
272         flight.serials.add(record.getSensorSerialNumber());
273
274     ////////////// Airborne Position Messages
275     if (msg.getType() == ModeSReply.subtype.ADSB_AIRBORN_POSITION)
{
276         airpos = (AirbornePositionMsg) msg;
277         Position rec = record.getSensorLatitude() != null ?
278             new Position(
279                 record.getSensorLongitude(),
280                 record.getSensorLatitude(),
281                 record.getSensorAltitude()) : null;
282
283         airpos.setNICS supplementA(flight.dec.getNICS supplementA());
284         Position pos = flight.dec.decodePosition(record.getTimeAtServer(),
rec, airpos);
285         if (pos == null)
286             ++err_pos_cnt;
287         else {
288             if (pos.isReasonable()) {
289                 Coordinate coord = new Coordinate(pos.getLongitude(), pos.getLatitude(),
290                     // set altitude to 0 if negative... looks nicer in google
291                     earth
292                     pos.getAltitude() != null && pos.getAltitude()>0 ? pos.getAltitude()
: 0);
293                     if (!flight.coords.contains(coord)) { // remove duplicates
294                         to safe memory
295                             flight.coords.add(coord);
296                             ++good_pos_cnt;
297                         }
298                     }
```

```
297     else {
298         flight.contains_unreasonable = true;
299         ++bad_pos_cnt;
300     }
301 }
302 }
303 ////////////// Surface Position Messages
304 else if (msg.getType() == ModeSReply.subtype.ADSB_SURFACE_POSITION)
{
305     surfacepos = (SurfacePositionMsg) msg;
306     Position rec = record.getSensorLatitude() != null ?
307         new Position(
308             record.getSensorLongitude(),
309             record.getSensorLatitude(),
310             record.getSensorAltitude()) : null;
311
312     Position pos = flight.dec.decodePosition(record.getTimeAtServer(),
313         rec, surfacepos);
314     if (pos == null)
315         ++err_pos_cnt;
316     else {
317         if (pos.isReasonable()) {
318             Coordinate coord = new Coordinate(pos.getLongitude(), pos.getLatitude());
319             if (!flight.coords.contains(coord)) { // remove duplicates
320                 to safe memory
321                     flight.coords.add(coord);
322                     ++good_pos_cnt;
323             }
324         else {
```

```
325         flight.contains_unreasonable = true;
326         ++bad_pos_cnt;
327     }
328 }
329 }
330 ////////////// Identification Messages
331 else if (msg.getType() == ModeSReply.subtype.ADSB_IDENTIFICATION)
{
332     ident = (IdentificationMsg) msg;
333     flight.callsign = ident.getIdentity();
334 }
335 }
336
337 // write residual flights to KML
338 for (String key : flights.keySet()) {
339     // number of flights filter
340     if (filter_max != null && kml.getNumberOffFlights()>=filter_max)
341         break;
342     if (option_nopos | flights.get(key).coords.size() > 0)
343         kml.addFlight(flights.get(key));
344 }
345
346 fileReader.close();
347 kml.writeToFile(kmlfile);
348
349 } catch (IOException e) {
350     // error while trying to read file
351     System.err.println("IO Error: "+e.getMessage());
352     System.exit(1);
353 } catch (Exception e) {
354     // something went wrong
355     System.err.println("Something went wrong: "+e.getMessage());
```

```
356     e.printStackTrace();
357     System.exit(1);
358 }
359
360 System.err.println("Read "+msgCount+" messages.");
361 System.err.println("Good positions: "+good_pos_cnt);
362 System.err.println("Bad positions: "+bad_pos_cnt);
363 System.err.println("Erroneous positions: "+err_pos_cnt);
364 System.err.println("Flights: "+flights_cnt);
365 }
366 }
```

3.5 本章小结

本章首先介绍了本系统的建立的基础，即三个主要依据的模型，分别是系统模型、网络模型和ADS-B模型，这些模型为系统假设了一个理想的设计应用环境模型。主要对本系统的加解密模块、解码模块以及可视化模块的设计原理以及具体编程算法进行详细的介绍。

第4章 系统测试

4.1 测试环境

本项目的所有调试以及测试均在个人笔记本下完成。

硬件：

联想ThinkPadT440P笔记本配置如下：

- (a) Intel(R) Core(TM) i3-4000M CPU @2.40GHz 2.39GHz
- (b) 4.00G内存
- (c) 500G硬盘

软件环境：

- (a) Windows 10专业版操作系统(64位)
- (b) Python 2.7
- (c) java 1.8
- (d) jython 2.5.2

4.2 测试结果

4.2.1 加密

4.2.1.1 功能实现

通过结果截图可以看到，Google Earth上显示的飞行器的轨迹与未经过与原始的轨迹一样，即即使对ADS-B消息进行加密处理也满足不改变飞行器轨迹的要求，只对唯一标识ICAO24实现了格式保护加密。加密后的匿名ICAO24与原ICAO24不同，长度不变，字符空间不变，依旧在{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f}字符集内，故加密结果符合格式加密的功能要求。

4.2.1.2 稳定性

如图所示，对不同的单条轨迹、多条轨迹的ADS-B消息进行FFX算法的格式保护加密，结果均符合功能要求，故此ADS-B隐私保护系统的稳定性较强。

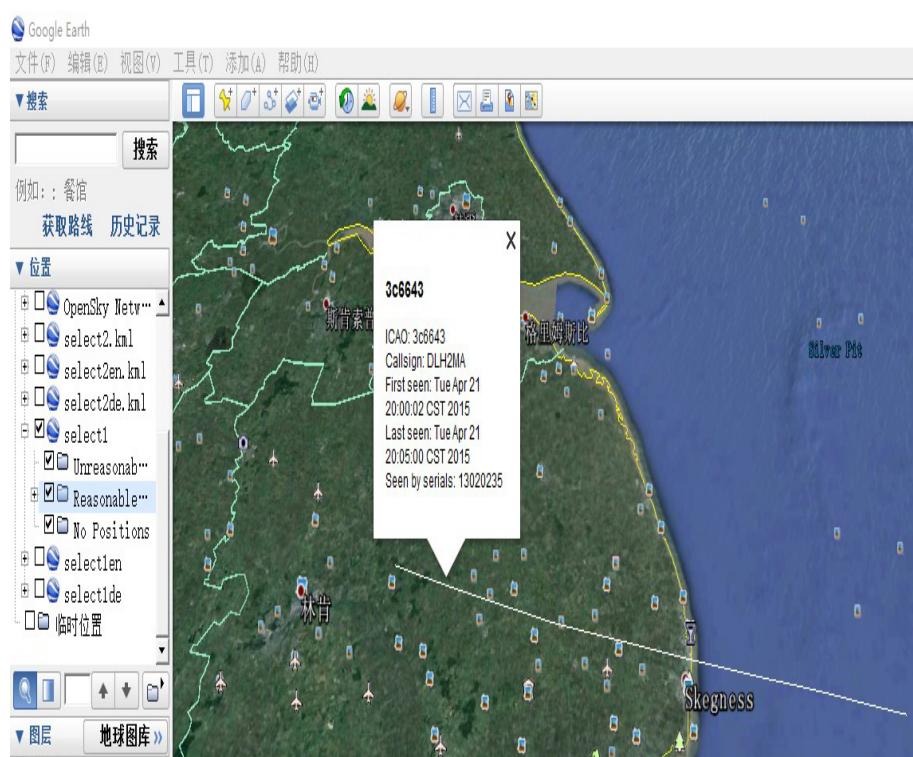


图 4-1 单条未经加密的原轨迹和原飞行标志ICAO24

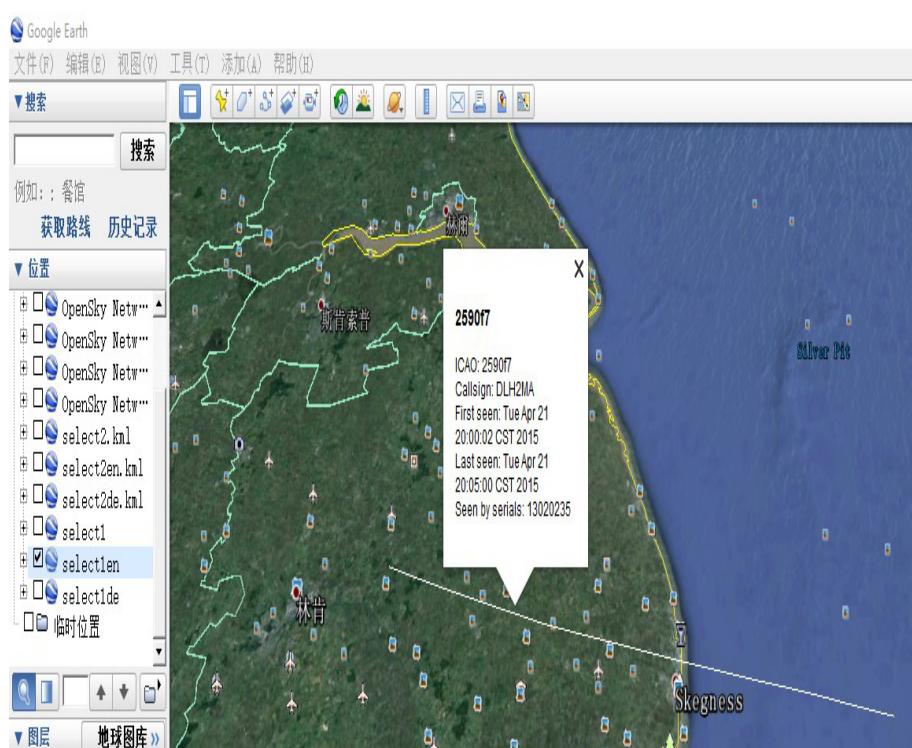


图 4-2 单条经过加密的轨迹和飞行标志ICAO24

4.2.1.3 时效性

如图所示，每条ADS-B消息的加密时间介于0.00ms-1.00ms之间，故将FFX加解密算法应用与ADS-B的隐私保护中对ADS-B消息的发送和接收的时效性无明显影响。

4.2.2 解密

4.2.2.1 功能实现

通过结果截图看到，经过解密飞行轨迹还是与原来一样，ICAO24也与未经过加密的ICAO24一样，即对ICAO24进行了解密还原，其他均不变，因此实现了解密的功能要求。

4.2.2.2 稳定性

如图所示，对不同的单条轨迹、多条轨迹的ADS-B消息进行FFX算法的格式保护解密，结果均符合功能要求，故此ADS-B隐私保护系统的稳定性较强。

4.2.2.3 时效性

如图15所示，每条ADS-B消息的解密时间与该消息的加密时间相同且时间介于0.00ms-1.00ms之间，故将FFX加解密算法应用与ADS-B的隐私保护中对ADS-B消息的发送和接收的时效性无明显影响。

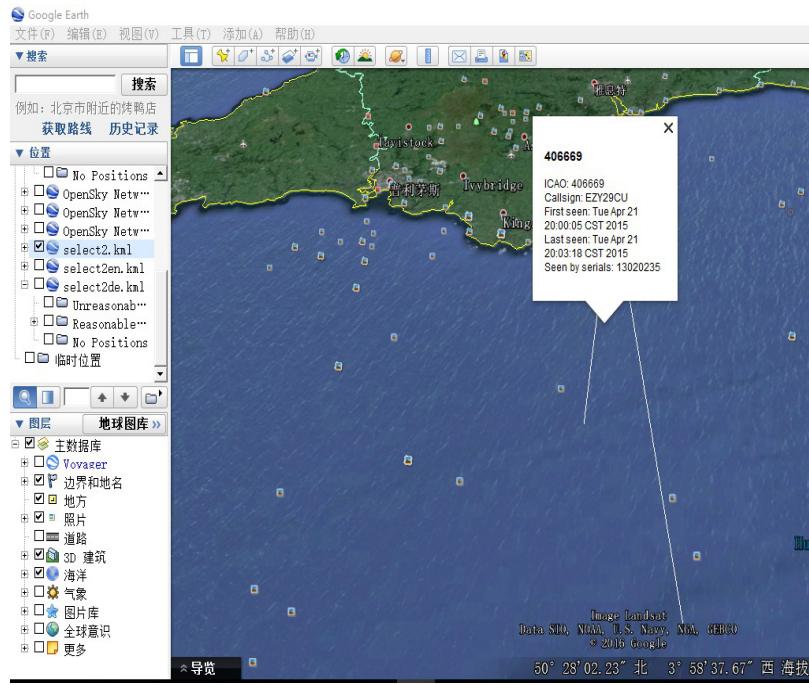
4.2.3 解码及解码结果的可视化

4.2.3.1 功能实现

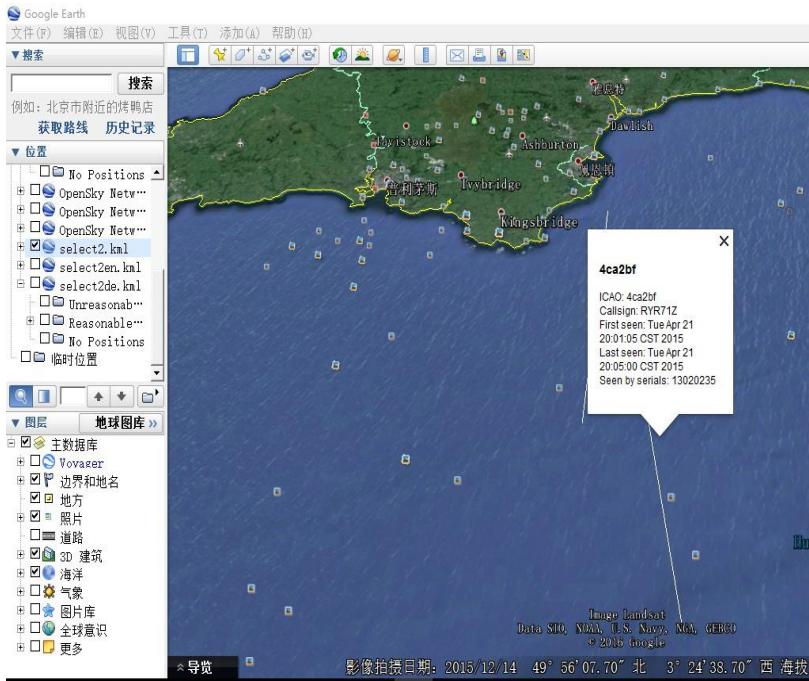
如上图所示，输入的.avro文件包含ADS-B消息和接收时间，接收地坐标等信息，输出的.kml文件在Google Earth上显示出来的是一条条轨迹和每条轨迹上代表飞行器的信息如ICAO24，因此实现了把ADS-B原始消息解码成具体的经纬度坐标或ICAO24飞行器标识信息并把经纬度信息和飞行器标识信息在地图软件上进行展示，因此也实现了结果可视化功能。

4.3 总体分析

本系统实现了对ADS-B数据链的隐私保护的基本课题要求。通过使用FFX格式保护算法对ADS-B消息的ICAO24部分进行选择性加密从而实现了让飞行器对无关者匿名的效果，从而保证了ADS-B数据链的隐私需求。同时，为了对加解密结果进行检验和效果进行展示，本系统还实现了对ADS-B消息进行解码和可视化处理的功能，通过解码和可视化处理，把ADS-B消息所包含的信息转化成地图软件上的一条条飞行轨迹和每条轨迹上代表的飞行器的信息如ICAO24直观的展示出来，从而证明了使用FFX格式保护算法对ADS-B消息的ICAO24部分进行加密不会影响飞行器的飞行轨迹信息，并且使用格式不糊算法这种加密方式使加密过后的ICAO24的字符数和字符域均不变，从而使加密后的ICAO24值也符合ADS-B消息格式对ICAO24格式的长度要求和字符域要求，使加密后的ICAO24也能成功解码，且解码后的ICAO24也符合ICAO24的命名标准，从而真正实现匿名的效果。由FFX格式保护算法的加密特性可得当输入的明文和密文一致时，对该明文进行多次加密的结果是相同的，故FFX格式保护算法是静态加密算法。因为FFX算法是静态加密算法，故长期使用同一密钥对同一飞行器的ADS-B消息进行加密会随着时间的增大而增大匿名ICAO24与真实ICAO24的对应破解的可能性，然而，对密钥的更换会产生一定的传输代价，因此需要对密钥更换周期做出合理的分析，使在这个周期频率内对密钥进行更新即能把对ICAO24进行对应破解的可能性降到一个可以接收的范围内有能把对密钥更新的成本控制在一定范围内，即寻找安全和经济的最佳平衡点。还有一点是ADS-B系统对消息收发的时效性是有很高的要求的，如果某种加密算法的加解密时效性不好，从而影响到整个ADS-B系统的时效性能的话则这种加密算法是不可取的，因为飞行器及乘客的人身安全保障是永远排在第一位的。由上面的数据可得，在个人笔记本的硬件配置下，使用的FFX加解密算法的加解密时间相同且都少于1ms，可以推测在ADS-B OUT的硬件配置下运行该加解密运算的耗时会更小，因此把该加解密算法嵌入ADS-B系统中对ADS-B的效率不会产生明显影响，因此满足实际应用的要求。



(a)



(b)

图 4-3 多条未经加密的原轨迹和原飞行标志ICAO24

(a)轨迹1; (b)轨迹2

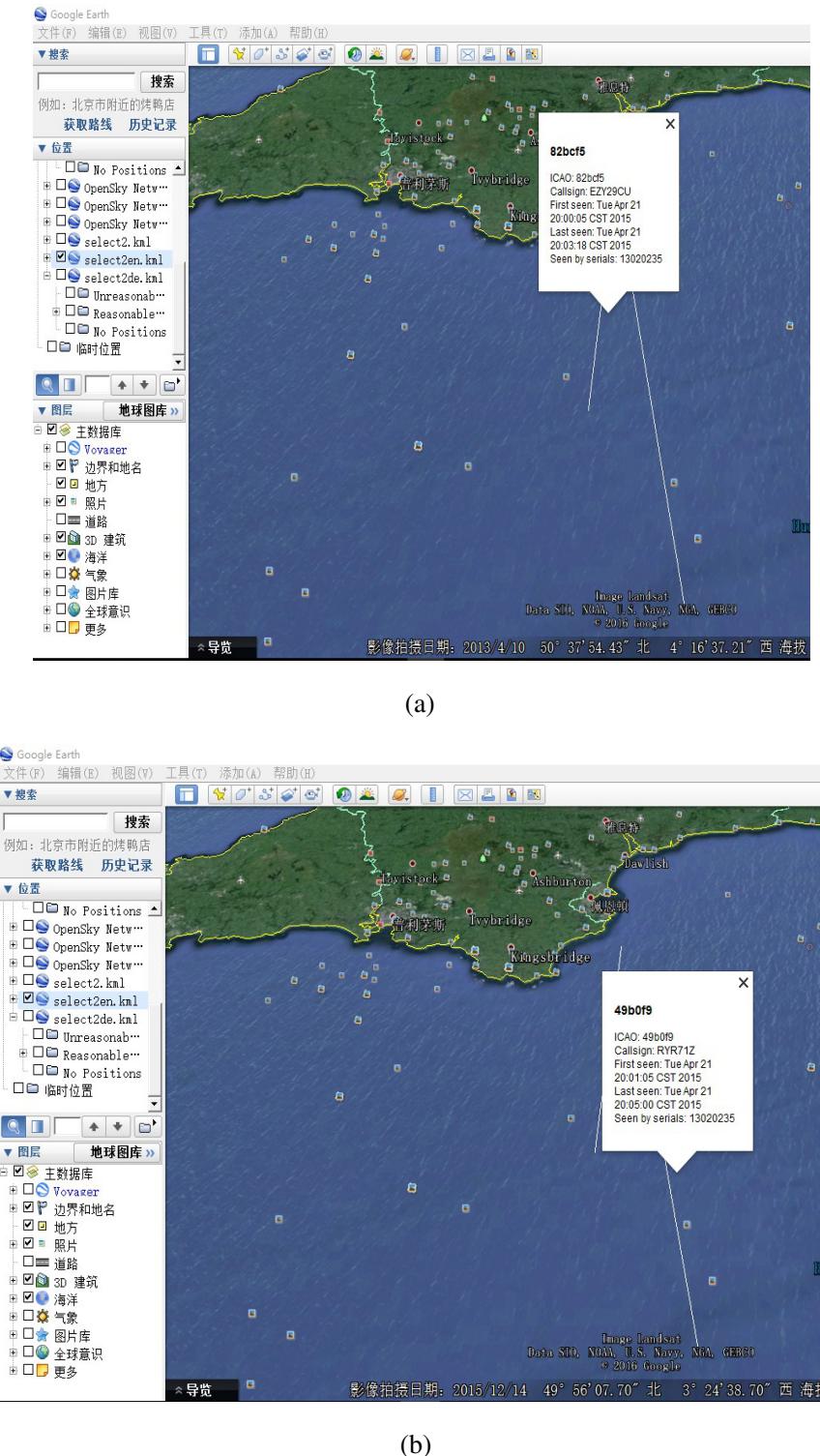


图 4-4 多条经过加密的轨迹和飞行标志ICAO24

(a)轨迹1; (b)轨迹2

```
C:\Python27\python.exe "C:/Users/Serlina/PycharmProjects/FinalyearProject/MDS-B out/libffx/benchmark.py"
RADIX=2, TWEAKSIZE=6, MESSAGESIZE=32, KEY=0xfbd4e36db74223c494ea846d2203c251
test #1 SUCCESS: (encrypt_cost=2.0ms, decrypt_cost=0.0ms, tweak=1011101, plaintext=10011000101010001100010000001111, ciphertext=100011100011011001010100000010100)
test #2 SUCCESS: (encrypt_cost=1.0ms, decrypt_cost=0.0ms, tweak=10101101, plaintext=1111001010101000110010011101101, ciphertext=0011110001111000110100110001110)
test #3 SUCCESS: (encrypt_cost=0.0ms, decrypt_cost=1.0ms, tweak=00110001, plaintext=0110110010100010001100110011101, ciphertext=00101110111100001001010011111)
test #4 SUCCESS: (encrypt_cost=0.0ms, decrypt_cost=1.0ms, tweak=1000011, plaintext=11000011100101110100101111010110, ciphertext=110111000011110111110110101110)
test #5 SUCCESS: (encrypt_cost=0.0ms, decrypt_cost=0.0ms, tweak=00100000, plaintext=1111001010111100110001111001001, ciphertext=1101001010111111010000001010000)
test #6 SUCCESS: (encrypt_cost=1.0ms, decrypt_cost=0.0ms, tweak=00010111, plaintext=11110111100011001101101111100, ciphertext=0111100000010111000000001011110)
test #7 SUCCESS: (encrypt_cost=1.0ms, decrypt_cost=0.0ms, tweak=01100110, plaintext=001101100011000000001111010001, ciphertext=001011110010101110101011111001)
test #8 SUCCESS: (encrypt_cost=0.0ms, decrypt_cost=1.0ms, tweak=11001101, plaintext=1101101010101000000000101001101, ciphertext=1000111010101011001110010000000)
test #9 SUCCESS: (encrypt_cost=0.0ms, decrypt_cost=1.0ms, tweak=10111000, plaintext=1011111011111100011010100110110, ciphertext=00100001001101010111110010101)
```

图 4-5 加解密一条消息的用时

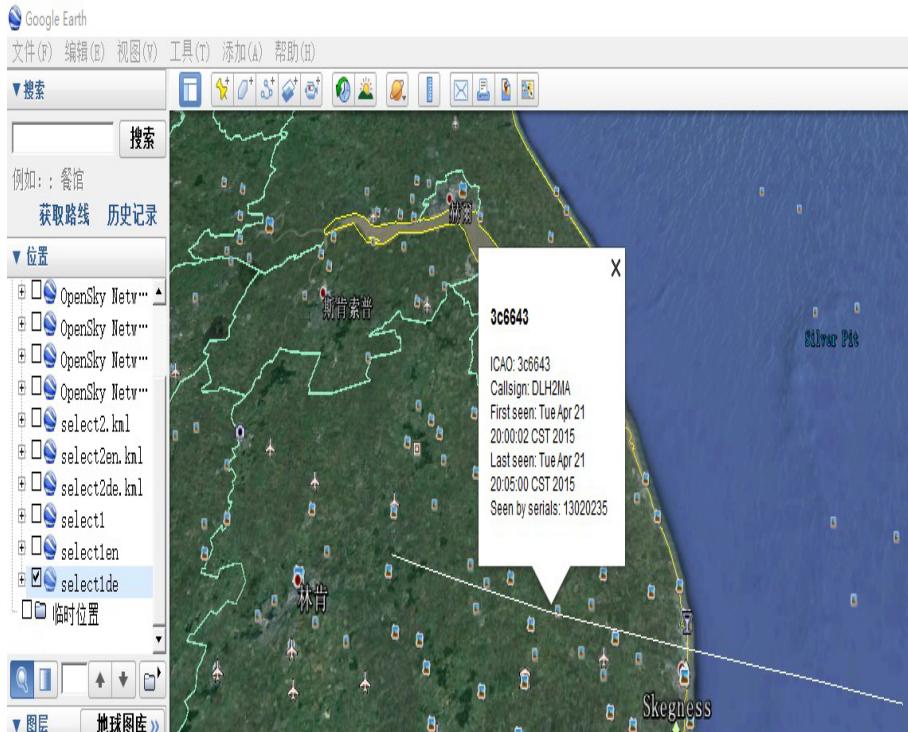
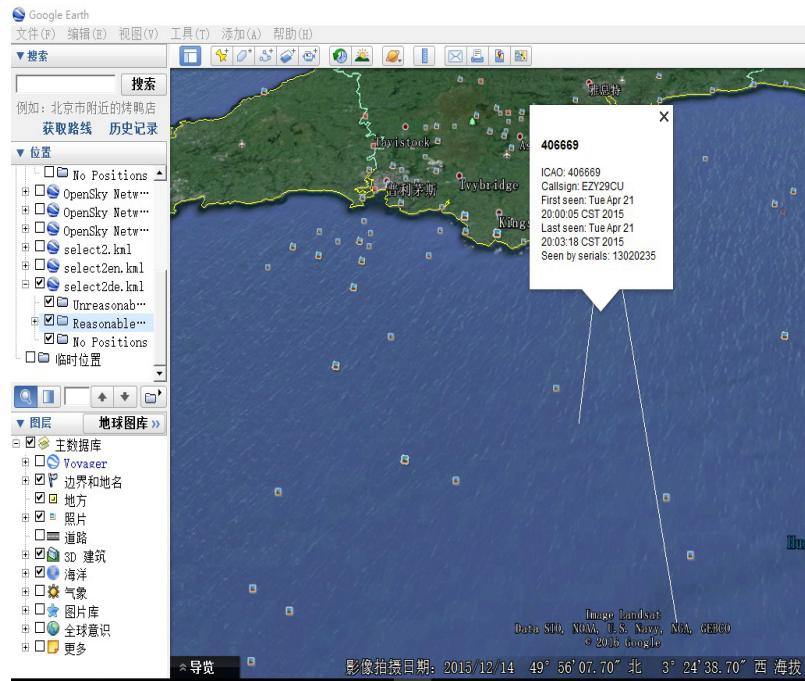
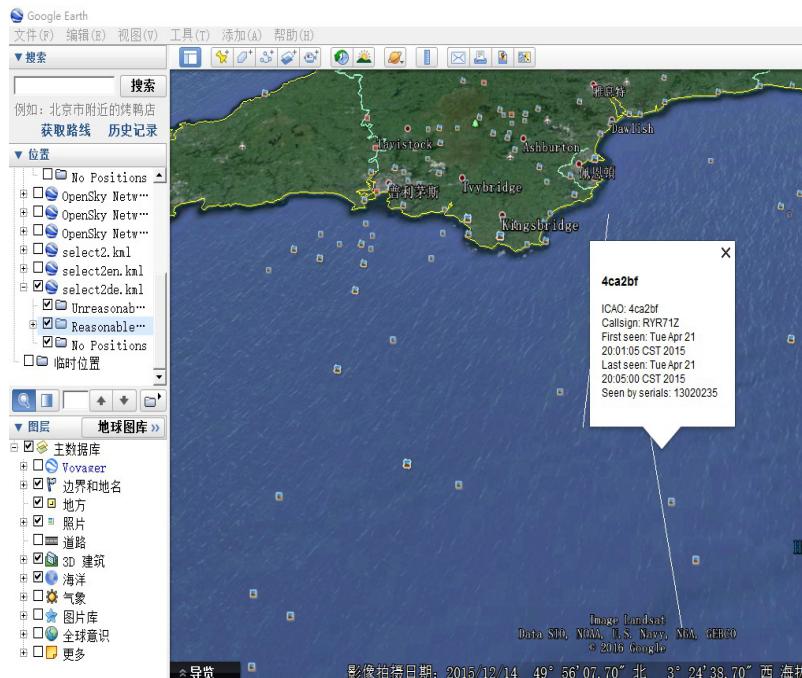


图 4-6 单条经过解密的轨迹和飞行标志ICAO24



(a)



(b)

图 4-7 多条经过解密的轨迹和飞行标志ICAO24

(a)轨迹1; (b)轨迹2

第5章 结束语

5.1 全文总结

本文中阐述了 1090ES 广播式自动相关监视系统的报文接收与解析过程，主要完成了以下工作：

1. 阐述本文的研究背景，分析了技术发展的国内外现状。

通过介绍航空监视系统的进化历程分析了新一代航空监视系统较上一代的航空监视系统的优势及其更替的必然性以及课题研究的意义。在分析和比较后，了解到我国的民用航空业目前正处于成长期，飞机的数量、飞行的规模和空域的范围都在不断扩大，需要引进并吸收 ADS-B 技术，以丰富和改进我们自己的空中交通管制系统和体系，并实现与国际最新技术的接轨。国内外在ADS-B隐私的保护上都有一定程度上的研究，针对ADS-B消息加密这种隐私保护方式需要更多考虑的问题是如何让加密算法更适合ADS-B的实际应用而不是仅仅停留在理论研究上和加密算法的时效性对于ADS-B应用来说也是非常重要的因为消息发送或接受的延误会影响ADS-B的性能，从而降低该航空监视系统的安全性。 ADS-B 技术的吸收和应用中，需要不断地研究其基本原理，掌握其技术核心，促进实现国际先进技术与中国本地情况的不断融合。

2. 对本课题中涉及到的相关技术进行深入的研究。

研究了系统中用到的一些基础技术，包括 ADS-B 技术、加密技术以及国外开源ADS-B消息获取网站OpenSky。在 ADS-B 技术中，重点研究了 ADS-B 的工作原理和 ADS-B 数据链和ADS-B消息格式。在ADS-B数据链技术的介绍中分析和对比了三种不同的ADS-B数据链技术，分别为MODE S 1090 ES、UAT和VDL MODE4，本课题的系统模型是建立在MODE S 1090 ES数据链收发方式上的。本课题实现的ADS-B隐私保护系统所需要的原始数据包括ADS-B原始消息，消息的发送和接收时间，接收地点经纬度等一系列信息以及处理这些信息的一些辅助函数。

3. 实现了报文加解密、解码及可视化处理。

首先，介绍各个部分的代码实现算使用的语言，FFX加/解密算法是使用python2实现的，因为python下对数学编程的包较全面和简单易用故选择用python实现该算法。解码以及可视化处理均是使用java实现的，因为OpenSky网站上提供的原始

数据为.avro文件格式，该文件格式是使用Hadoop大数据平台工具进行操作的文件，这些数据操作工具以及函数包只能再Linux平台下使用而Java具有跨平台性好的特点而且OpenSky网站上提供的一些辅助函数均是使用Java实现的，所以本课题实现中的解码及可视化部分代码均采用Java实现。

第二，对.avro文件中提取出的ADS-B消息进行加/解密处理，采用之前分析过的适用于ADS-B应用实际的加解密算法FFX加/解密算法，在FFX算法中需要用户输入密钥参数。然后将经过加/解密的ADS-B消息输出到解码模块。该算法为静态算法，即密钥和其他参数一定的情况下对同一消息进行加/解密的结果是一样的。

第三，对经过加/解密的ADS-B消息进行解码处理。解码所依据的原理是ADS-B的消息格式，即ADS-B消息的各个域所代表的编码含义以及各个域为不同值是所代表的编码信息，根据ADS-B某些域的值的范围分别根据其编码规则进行合适的编码。然后把解码后得到的消息分别放到相应的类的具体对象中。

第四，对ADS-B解码后得到的信息进行可视化处理，即把它的轨迹、ICAO24和时间等信息展示在Google Map上。要让位置信息和其他标志信息在Google Earth软件中显示，要先创建.kml文件的框架描述然后建立.kml文件再把这些ADS-B解码后得到的具体信息放在.kml文件中。最后，描述了关键功能的代码实现，包括加/解密算法、位置解码和.kml文件转换。

5.2 后续工作展望

一、密钥更新：

由于两次密钥相同的加密会产生相同的结果，故此假名多次使用后会增大假名与真实飞机身份的相关性。故需找到最合适的更新周期从而为密钥制定合适的周期更新方案，从而减少飞机和假名之间的相关性，从而极大的增加攻击者通过ADS-B消息跟踪飞机的难度。

二、用随机静默期的方法解决ADS-B隐私保护问题：

ADS-B的隐患有很多，包括主动攻击和被动攻击，目前也提出了针对各种攻击的具体解决方案。关于ADS-B被动攻击中的ADS-B隐私保护的方法不仅有数据链加密这种形式，目前提出的解决方案中较可靠的还有一种叫随机静默期的方法，这种方法的主要工作原理是让ADS-B OUT每隔一段时间就短暂的不播报ADS-B消息，这段不播报ADS-B消息的间期称为随机静默期，通过在随机静默期不播报消息的方式达到一种把飞行器与周围的飞行器混淆的效果。故接下来可以尝试研究用随机

静默期这种方式具体的解决ADS-B隐私的保护问题并对比这两种问题的长处和短处。

5.3 收获及体会

国际民航组织（ICAO）早就有意发展 ADS-B 系统，在欧洲和北美对该系统的试验已进行了多年，并已逐渐进入实际应用阶段。在美国阿拉斯加境内和东、西海岸部分地区，ADS-B 系统已进入了实际的应用阶段并取得了丰富的实践经验。我国对 ADS-B 技术的应用基本还是空白。目前除美国以外我国是第一家把UAT 模式的 ADS-B 技术应用于通用航空领域的国家。在我国低雷达覆盖的情况下，ADS-B 用于民航的空中交通管制是非常必要的，特别适用于没有二次雷达覆盖的地区。该系统能实现空中交通管制的自动监控、提高地面指挥的管理水平、工作效率和飞机飞行的安全性，并能节约建设二次雷达系统所需的大量资金，可在短期内使无雷达覆盖区域空中交通管理水平有质的飞跃和提高。ADS-B 技术的进一步推广，即使雷达不能覆盖，也能使民用航空更安全，为我国进一步开放空域，发展民航事业，提供了坚强的物质保证。在ADS-B应用的大前提下，保证ADS-B系统的安全性也变得尤其重要。在ADS-B隐私保护系统的实现中一个很重要的收获是任何针对实际应用的研究都要紧贴现实不能凭空想象，否则研究出来的东西就会缺乏实际应用价值，例如目前已经存在无数的加密算法，但是要针对ADS-B系统以及ADS-B消息的特点和格式进行加解密算法的选择，否则算出来的算法将没有实际应用意义。

参考文献

- [1] 张辰. ADS-B 信号解码板设计及报文处理[D]. 哈尔滨: 哈尔滨工程大学, 2013, 3--6
- [2] D. McCallie, J. Butts, R. Mills. Security analysis of the ADS-B implementation in the next generation air transportation system[J]. International Journal of Critical Infrastructure Protection, 2011, 4(2):78--87
- [3] 黄晋. 广播式自动相关监视(ADS-B)在中国民航飞行学院的应用研究[D][D]. 成都: 西南交通大学, 2008, 7--10
- [4] 戴超成. 广播式自动相关监视(ADS-B)关键技术及仿真研究[D]. 上海: 上海交通大学, 2011, 1--6
- [5] K. T. M. Krozel. Aircraft ADS-B intent verification based on a Kalman tracking filter[A][C]. AIAA Guidance Navigation and Control Conference Proceedings, Denver, 2000, 4067--4072
- [6] J. Krozel, D. Andrisani, M. A. Ayoubi, et al. Aircraft ADS-B data integrity check[C]. AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum, Chicago, 2004, 1--11
- [7] J. Krozel, D. Andrisani. Independent ADS-B verification and validation[C]. AIAA Aviation, Technology, Integration, and Operations Conference Proceedings, Arlington, 2005, 1--11
- [8] 刘畅. 1090ES 广播式自动相关监视系统 CPR 算法解析[J]. 电子世界, 2015, 7(13):180--181
- [9] 白松浩, 朱晓辉, 陈志杰. 广播式与合约式自动相关监视的信息转换[J][J]. 系统工程与电子技术, 2005, 27(9):1658--1660
- [10] 陈志杰, 朱晓辉. 基于现有机载设备信道构建 UAT 数据链路[J][J]. 系统工程与电子技术, 2008, 30(10):1840--1843
- [11] H. F. J. Y. Wei. Modeling and Simulation of an Aeronautical Subnetwork Based on Universal Access Transceiver[C]. Asia Simulation Conference, Chengdu, 2008, 541--544

- [12] K. Sampigethaya, R. Poovendran. Security and privacy of future aircraft wireless communications with offboard systems[C]. Communication Systems and Networks (COMSNETS) 2011 Third International Conference, Bellevue, 2011, 1--6
- [13] K. Sampigethaya, R. Poovendran, L. Bushnell. A framework for securing future e-enabled aircraft navigation and surveillance[C]. AIAA Proceedings, Seattle, 2009, 1--10
- [14] A. Costin, A. Francillon. Ghost in the Air (Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices[J]. Black Hat USA, 2012, 49(31):1--12
- [15] M. Schäfer, V. Lenders, I. Martinovic. Experimental analysis of attacks on next generation air traffic communication[C]. Applied Cryptography and Network Security, Banff, 2013, 253{271
- [16] B. Kovell, B. Mellish, T. Newman, et al. Comparative analysis of ADS-B verification techniques[J]. The University of Colorado, Boulder, 2012, 4(12):63--70
- [17] B. Nuseibeh, C. B. Haley, C. Foster. Securing the skies: In requirements we trust[J]. Computer, 2009, 4(9):64--72
- [18] J. Burbank, R. Nichols, S. Munjal, et al. Advanced communications networking concepts for the National Airspace System[C]. 2005 IEEE Aerospace Conference Proceedings, Big Sky, MT, 2005, 1905{1923
- [19] W. Li, P. Kamal. Integrated aviation security for defense-in-depth of next generation air transportation system[C]. Technologies for Homeland Security (HST) 2011 IEEE International Conference, Waltham, MA, 2011, 136{142
- [20] T.-C. Chen. An authenticated encryption scheme for automatic dependent surveillance-broadcast data link[C]. Cross Strait Quad-Regional Radio Science and Wireless Technology Conference, New Taipei City, 2012, 127--131
- [21] 何桂萍 . ADS-B 数据链的比较及特性研究[J] [J]. 中国民航飞行学院院报, 2010, 21(4):3--6

参考文献

- [22] 时宏伟. ADS-B 数据链应用技术的综合评述[J]. 空中交通管理, 2007, 17(6):13--16
- [23] 杨成 . ADS-B 数据链应用风险与对策研究[J]. 现代电子技术, 2014, 37(21):98--101
- [24] 杨荣盛何光勤. 基于低空空域适用性的 ADS-B 数据链研究[J] [J]. 苏州科技学院学报, 2011, 24(2):63--67
- [25] 李德胜. 基于 UAT 数据链的 ADS-B 机载系统的设计与实现[J] [J]. 航空维修与工程, 2012, 8(2):56--59
- [26] 吕小平. ADS—B 应用中 UAT 技术介绍[J] [J]. 空中交通管理, 2007, 12(8):19--21
- [27] 丁立平. 基于 VDL-4 技术的机场场面车辆管理系统研究[J]. 指挥信息系统与技术, 2010, 11(6):55--60
- [28] 邱伟杰. ADS-B广播式自动相关监视系统多数据链路融合方案探索[J] [J]. 科技风, 2011, 34(15):23--25
- [29] 苏杰, 王波. 三种 ADS-B 技术比较及基于 1090ES 的 ADS-B 在成都-九寨的应用[J] [J]. 空中交通管理, 2007, 11(7):55--57
- [30] 刘哲理贾春福. 保留格式加密技术研究[J]. 软件学报, 2012, 23(1):152--170
- [31] M. Bellare, P. Rogaway, T. Spies. The FFX mode of operation for format-preserving encryption[J]. NIST submission, 2010, 20(13)
- [32] M. Bellare, T. Ristenpart, P. Rogaway, et al. Format-preserving encryption[C]. Selected Areas in Cryptography, Calgary, Alberta, Canada, 2009, 295--312
- [33] J. Black, P. Rogaway. Ciphers with arbitrary finite domains[C]. Springer Topics in Cryptology—CT-RSA, CA, USA, 2002, 114--130

致 谢

在做毕业设计期间，首先衷心感谢我的毕业设计指导老师杨浩淼副教授，他在我做毕业设计期间给予了我很大的帮助，无论是在对课题的定位理解方向选择上还是在实际操作中遇到的很多具体的问题上他都给予了我很多耐心的指导和实用的建议。另外，董志斌等同学在我的方案选择上和在实际代码写作和调试上也给予了我不少灵感和很大的帮助。在此我对我的指导老师和同学、学长等人表示衷心的感谢。感谢这篇论文的引用中所涉及到的国内外学者，没有他们之前的研究我也很难完成毕业论文的写作。由于学术能力和表达水平的有限，该论文难免有不足之处，望各位老师和同学批评指正。

A parameter collection for enciphering strings of arbitrary radix and length

1.1 Introduction

A scheme for format-preserving encryption (FPE) is supposed to do that which a conventional (possibly tweakable) block cipher does—encipher messages within some message space χ , except that message space, instead of being something like $\chi = \{0,1\}^{128}$, is more general [1,3]. For example, the message space might be the set $\chi = \{0,1,\dots,9\}^{16}$, in which case each 16-digit plaintext $X \in \chi$ gets enciphered into a 16-digit cipher text $Y \in \chi$. In a string-based FPE scheme—the only type of FPE that we consider here—the message space is of the form $\chi = \{0,1,\dots,radix-1\}^n$ for some message length n and alphabet size $radix$.

One way to achieve FPE is with a Feistel-based mode of operation. The underlying round function can be based on a conventional block cipher, say AES. Two proposals along these lines were recently submitted to NIST. One is FFX, by the present authors [2]. Another is BPS, by Brier, Peyrin, and Stern [4], which was developed independently and submitted soon after FFX.

The FFX scheme is more open-ended than BPS; by itself, it is more a framework than a mode. To turn FFX into a concrete mode one has to fix a variety of parameters—some nine parameters in all. To make things concrete, the authors of FFX suggest two parameter collections, named A2 and A10. Scheme FFX-A2 encrypts binary strings of 8{128 bits, while FFX-A10 encrypts decimal strings of 4{36 digits. For sufficiently long strings, both schemes use 12 rounds of Feistel. But the number of rounds escalate as messages get shorter, going as high as 24 rounds (for FFX-A10) or 36 rounds (for FFX-A2) for the shortest permitted message lengths.

BPS brings to the table two new characteristics that a user might like. First is a more aggressive round count: exactly eight rounds are used, regardless of the input's length. Fewer rounds mean greater speed (and, of course, a less conservative design). Second, BPS includes a CBC-like chaining mode to allow for the encryption of very long strings.

We note that the CBC-like mode of BPS or, more generally, any CBC-like mode cannot possibly meet the security requirements specified in the literature for an FPE [1,3]: you will not get a pseudo random permutation (PRP). To begin with, the first bit of cipher text does not depend on the last bit of plain text. We feel that any scheme that calls itself a scheme for format-preserving encryption ought to achieve the PRP goal and preferably the strong PRP goal as well.

The FFX[radix] mode. This document provides new FFX parameter collections. First, we expand the allowed radix values; now, any reasonable radix may be used, not just 2 or 10. Second, we enlarge the allowed message lengths, permitting effectively arbitrary strings to be enciphered. Finally, we select more aggressive round counts than we did for FFX-A2 and FFX-A10.

To accomplish the above, we define the FFX scheme we call FFX[radix]. The bracketed value compactly names an FFX parameter collection. The value of radix is a number between 2 and 2^{16} . Messages to be enciphered under FFX[radix] are regarded as strings of characters drawn from the alphabet $\text{Chars} = \{0, 1, 2, \dots, radix - 1\}$. Scheme FFX[radix] does its work using an AES-based balanced Feistel network. If the message length is odd, an alternating, maximally-balanced Feistel scheme is used instead.

Mode FFX[radix] effectively unifies and extends FFX-A2 and FFX-A10; we suggest its use in lieu of the later modes. The radix is more general, messages can be longer, and the number of rounds is made constant, rather than depending on the message length n . We have not tried to maintain upward compatibility; mode FFX[2] and FFX[10] do not coincide with with FFX-A2 and FFX-A10, and they would not coincide even if the same number of rounds had been employed (although, in that case, there would be no cryptographically significant difference).

1.2 The Scheme FFX[radix]

We assume the notation from the FFX spec [2] but, for the user's convenience, we recall the most relevant definitions here. Plaintexts and ciphertexts are regarded as strings over the alphabet $Chars = \{0, 1, \dots, radix - 1\}$. By $|X|$ we denote the length of string X , the number of characters in it. Let Byte denote $\{0, 1\}^8$, the set of 8-bit bytes. By $|T|_8$ we denote the length, in bytes, of the byte string $T \in \text{Byte}^*$. With radix understood, the blockwise addition function \boxplus is defined by $a_1 \cdots a_n \boxplus b_1 \cdots b_n = c_1 \cdots c_n$ where $c_1 \cdots c_n$ is the unique string such that $\sum c_i radix^{(n-i)} = (\sum a_i radix^{(n-i)} + \sum b_i radix^{(n-i)}) \bmod radix^n$. By $X \boxplus Y$ we mean the unique string Z such that $Y \boxplus Z = X$. By $[s]^i$ we mean the i -byte string that encodes the number $s \in [0..2^{8i} - 1]$. The function $NUM_{radix}(X)$ takes a nonempty string $X \in \{0, \dots, radix - 1\}^*$ and converts it to the corresponding number, where the number is interpreted in the given radix, most-significant character first. The function $STR_{radix}^m(x)$ takes a number $x \in [0 \cdots radix^m - 1]$ and returns the m -character string that represents it in the given radix, most significant character first. When $K \in \{0, 1\}^{128}$ and $X \in \{0, 1\}^*$ and $|X|$ is divisible by 128, algorithm $CBC-MACK_K(X)$ is defined as follows. First, let $X_1 \cdots X_m \leftarrow X$ where $|X_i| = 128$ and let $Y \leftarrow 0^{128}$. Then, for $j \leftarrow 1 \sim m$, set $Y \leftarrow AES_K(Y \oplus X_i)$. Finally, return Y .

The mode enciphers strings over $Chars = \{0, 1, \dots, radix - 1\}$. It does so using a maximally-balanced Feistel network and a round function based on the AES CBC-MAC. We do not repeat further material from the FFX spec [2] except, for the reader's convenience, we do give the definition of FFX itself, as specialized to alternating Feistel (method=2). This makes our specification terse but self-contained. Specification. In Figure 1 we specify FFX and the parameter collection [radix]. When FFX is instantiated with this parameter collection one obtains the scheme FFX[radix]. In Figure 2 we graphically illustrate the latter mode.

The left-hand side depicts the behavior for an odd-length input (in which case alternating, almost-balanced Feistel is used); the right-hand

<pre> 10 algorithm FFX.Encrypt(K, T, X) 11 if $K \notin \text{Keys}$ or $T \notin \text{Tweaks}$ or 12 $X \notin \text{Chars}^*$ or $X \notin \text{Lengths}$ 13 then return \perp 14 $n \leftarrow X$; $\ell \leftarrow \text{split}(n)$; $r \leftarrow \text{rnds}(n)$ 15 $A \leftarrow X[1.. \ell]$; $B \leftarrow X[\ell + 1.. n]$ 16 for $i \leftarrow 0$ to $r - 1$ do 17 $C \leftarrow A \boxplus F_K(n, T, i, B)$ 18 $A \leftarrow B$; $B \leftarrow C$ 19 return $A \parallel B$ </pre>	<pre> 20 algorithm FFX.Decrypt(K, T, Y) 21 if $K \notin \text{Keys}$ or $T \notin \text{Tweaks}$ or 22 $Y \notin \text{Chars}^*$ or $Y \notin \text{Lengths}$ 23 then return \perp 24 $n \leftarrow Y$; $\ell \leftarrow \text{split}(n)$; $r \leftarrow \text{rnds}(n)$ 25 $A \leftarrow Y[1.. \ell]$; $B \leftarrow Y[\ell + 1.. n]$ 26 for $i \leftarrow r - 1$ downto 0 do 27 $C \leftarrow B$; $B \leftarrow A$ 28 $A \leftarrow C \boxminus F_K(n, T, i, B)$ 29 return $A \parallel B$ </pre>
<code>radix</code>	a number $\text{radix} \in [2.. 2^{16}]$
<code>Lengths</code>	$[\text{minlen}.. \text{ maxlen}]$ where $\text{minlen} = 2$ if $\text{radix} \geq 10$ and $\text{minlen} = 8$ otherwise; and $\text{ maxlen} = 2^{32} - 1$.
<code>Keys</code>	$\{0, 1\}^{128}$
<code>Tweaks</code>	$\text{BYTE}^{\leq \text{ maxlen}}$ where $\text{ maxlen} = 2^{32} - 1$
<code>addition</code>	1
<code>method</code>	2
<code>split(n)</code>	$\lfloor n/2 \rfloor$
<code>rnds(n)</code>	10
<code>F</code>	given below
	alphabet is $\text{Chars} = \{0, 1, \dots, \text{radix} - 1\}$
	permitted message lengths
	128-bit AES keys
	tweaks are arbitrary byte strings
	blockwise addition
	alternating Feistel
	maximally balanced Feistel
	number of rounds
	AES-based round function
30 algorithm $F_K(n, T, i, B)$	
31 $\text{vers} \leftarrow 1$; $t \leftarrow T _8$; $\beta \leftarrow \lceil n/2 \rceil$; $b \leftarrow \lceil \lceil \beta \log_2(\text{radix}) \rceil / 8 \rceil$; $d \leftarrow 4 \lceil b/4 \rceil$	
32 if EVEN(i) then $m \leftarrow \lfloor n/2 \rfloor$ else $m \leftarrow \lceil n/2 \rceil$	
33 $P \leftarrow [\text{vers}]^1 \parallel [\text{method}]^1 \parallel [\text{addition}]^1 \parallel [\text{radix}]^3 \parallel [\text{rnds}(n)]^1 \parallel [\text{split}(n)]^1 \parallel [n]^4 \parallel [t]^4$	
34 $Q \leftarrow T \parallel [0]^{(-t-b-1) \bmod 16} \parallel [i]^1 \parallel [\text{NUM}_{\text{radix}}(B)]^b$	
35 $Y \leftarrow \text{CBC-MAC}_K(P \parallel Q)$	
36 $Y \leftarrow \text{first } d + 4 \text{ bytes of } (Y \parallel \text{AES}_K(Y \oplus [1]^{16}) \parallel \text{AES}_K(Y \oplus [2]^{16}) \parallel \text{AES}_K(Y \oplus [3]^{16}) \dots)$	
37 $y \leftarrow \text{NUM}_2(Y)$	
38 $z \leftarrow y \bmod \text{radix}^m$	
39 return $\text{STR}_{\text{radix}}^m(z)$	

图 A-1 Figure 1: Definition of FFX[radix]

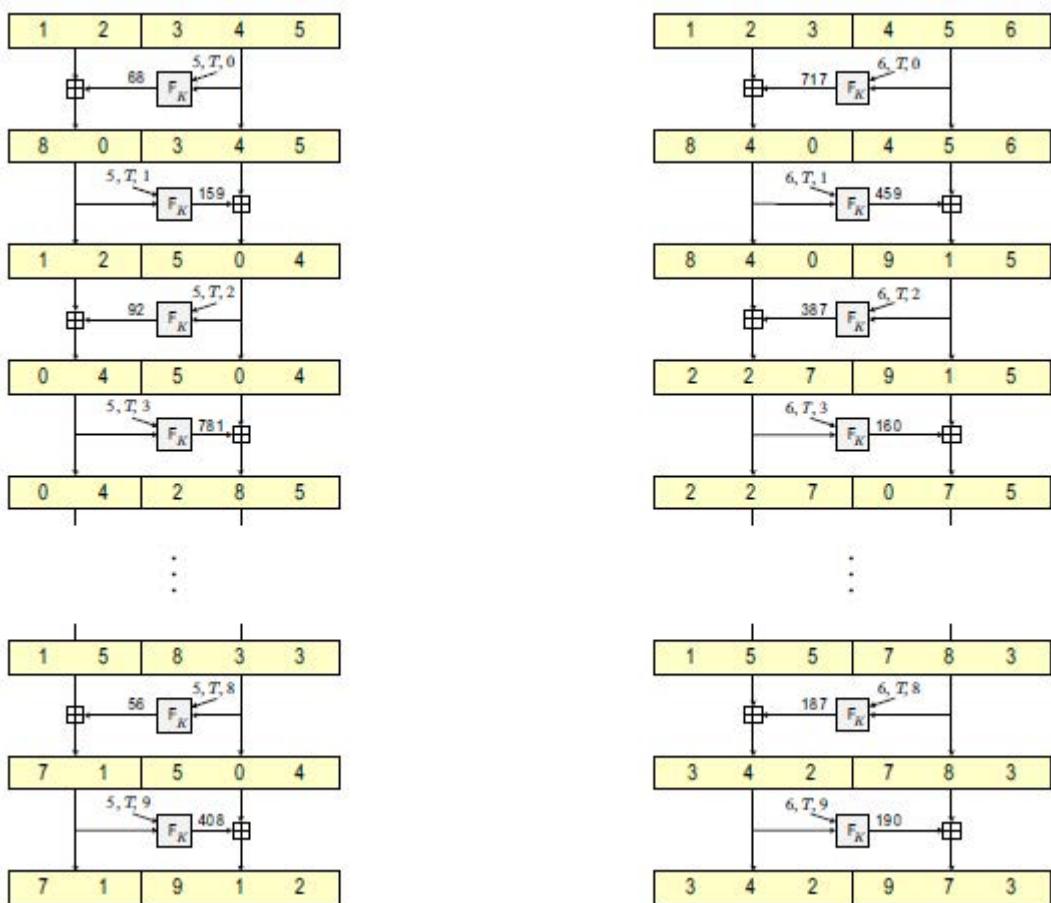


图 A-2 Figure 2: Illustration of FFX[10]

side shows an even-length input (in which case balanced Feistel is employed). The indicated round-function outputs are fictitious; the actual values depend on the key K and tweak T, neither of which has been specified.

1.3 Comments

A thorough discussion of FFX is given in the spec to which this document is an addendum [2]; we confine ourselves here to a few quick comments.

First, we would emphasize that, while we have permitted enciphering very long strings, mode FFX[radix] and indeed FFX in general is intended for enciphering relatively short strings. For binary strings whose length exceeds 128 bits, EME2 [5] may be a better choice than FFX[radix]. For long non-binary strings, one can, similarly, do “better” than FFX[radix], both in terms of speed and proven-security results. However, nobody has written a spec for such a mode, one reason we have chosen to allow use of FFX[radix] even for quite long strings.

Second, we would like to briefly comment on the number of rounds, $\text{rnds}(n) = 10$. Our earlier work, in parameter collections A2 and A10, had used a rather complicated function (we have come to call it 12^+) in which the number of rounds $\text{rnds}(n)$ was four more than the minimal number of rounds r such that $r \cdot \text{split}(n) \cdot \log_2(\text{radix}) \geq 128$; but at least 12 and always a multiple of 6. The rule was heuristically motivated and extremely conservative. For radix = 10 it translates to $\text{rnds}(n)=12$ when $n \geq 10$; $\text{rnds}(n)=18$ if $6 \leq n \leq 9$; and $\text{rnds}(n)=24$ if n is 4 or 5. The feedback we got—and our own sensibilities as well—was that a non-constant number of rounds was rather too complex. Also, it is unintuitive why one should increase the number of rounds as messages get shorter. While the explanation for these choices make sense (see the appendices in the FFX spec [2]), we have come to regard the conceptual cost as too high. In the meantime, there remains no real evidence that, once one overcomes the “meet-in-the-middle” attack, more rounds are actually needed for shorter strings. We have already noted

that BPS [4] selected a round count of 8, independent of message lengths. With all these considerations in mind, we have elected to simplify and cut back the round count to 10, speeding up implementations yet leaving some margin of safety. The choice is certainly not as (hyper-)conservative as a round count of 12⁺, but ultimately the designers of a mechanism must make a choice, based on their informed judgement and the cryptographic state-of-the-art.

Third, as discussed in the FFX spec [2], implementation choices will greatly affect performance. For example, the first AES computation implicit in line 35 need be done only once across all the rounds. When the radix is not a power of two, the mod at line 38 may require significant attention.

Finally, we re-emphasize that FFX[radix] is an instantiated version of FFX, and that other instantiations are compliant with the higher-level scheme.

1.4 References

- [1] M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-preserving encryption. Selected Areas in Cryptography (SAC 2009), LNCS 5867, Springer, 2009. Also ePrint report 2009/251.
- [2] M. Bellare, P. Rogaway, and T. Spies. The FFX mode of operation for format-preserving encryption. Draft 1.1. February 20, 2010. Manuscript, available on the NIST website at URL <http://csrc.nist.gov/groups/ST/toolkit/BCM/modes-development.html>.
- [3] J. Black and P. Rogaway. Ciphers with arbitrary finite domains. RSA Data Security Conference, Cryptographer's Track (RSA CT '02). LNCS vol. 2271, pp. 114{130, Springer, 2002.1
- [4] E. Brier, T. Peyrin, and J. Stern. BPS: a format-preserving encryption proposal. Available at URL <http://csrc.nist.gov/groups/ST/toolkit/BCM/modes-development.html>. Undated manuscript, published on the above website in April 2010.

- [5] IEEE P1619.2. Draft standard architecture for wide-block encryption for shared storage media. 2008. Available from <https://siswg.net>.

A parameter collection for enciphering strings of arbitrary radix and length

1.1 Introduction

1.1.1 背景

一种对于格式保护加密(FPE)的方案理应做到传统块加密能做到的，比如加密具有空格的消息空间而不加密空格，不只是像 $\chi = \{0,1\}^{128}$ 这样的消息空间，而是更加普遍[1,3]。举个例子，消息空间可能是集合 $\chi = \{0,1,\dots,9\}^{16}$ ，在这种情况下每加密一条16-digit的明文 $X \in \chi$ 则得到一条16-digit的密文 $Y \in \chi$ 。在一个以字符串为基础的FPE方案里，我们考虑的唯一一种对于一些消息长度n和字母表大小radix的FPE类型的消息空间是 $\chi = \{0,1,\dots,radix - 1\}^n$ 这样的形式。实现FPE的一种方式使使用以Feistel为基础的操作模型。作为基础的轮功能可以是基于传统的块加密算法，比如AES。两个关于这个问题的提议目前已经提交给NIST了。其中一种就是FFX，由这篇文章的作者提供[2]。另一种是BPS，由Brier, Peyrin, 和 Stern [4]独立完成并在FFX不久之后提交。

FFX方案相对于BPS方案更加的开放。就它本身而言，它是一种框架而不是一种模式。如果要把FFX变成一种具体的模式的话需要为其填充一系列的参数，它一共有9个参数。为了让它看起来更加具体，FFX的作者建议要填充两个参数，叫做A2和A10。方案FFX-A2加密8{128 bits二进制比特串，而FFX-A10加密4{36 digits的十进制字符串。对于高效率的长字符串，两种方案都采用的是12轮的Feistel。但是轮数会随着消息长度的缩短而快速增长，对于所允许的最短消息长度，FFX-A10的轮数最高能到到24轮，FFX-A2最高能达到36轮。

BPS带来了两种用户可能会喜欢的心特性。第一个是更加有挑战性的轮数：确定使用8轮，与用户的输入长度无关。更小的轮数意味着更快的速度（当然，也意味着更不保守的设计）。第二个是BPS包括一个像CBC一样的链状模式，因此可以加密非常长的字符串。

我们需要注意的是像BPS这种CBC模式的加密算法，或者更一般性的说，任何像CBC一样的模式都可能无法达到FPE规定中所要求的安全性需求[1,3]：我们将无

法得到一个伪随机置换(PPR)。首先，密文的第一个比特不依赖于明文的最后一个比特。我们认为任何一种自称为格式保护加密的方案都应该达到(PPR)的目标，当然强PPR目标更好。

1.1.2 FFX[radix]的模式

这篇文章提供了新的FFX参数集合。首先，我们拓展了允许的radix值；现在任何合理的radix值都可以被使用，不仅仅是2或者10。第二，我们也拓展了允许的消息长度，允许更有效的任意长度字符串进行加密。最后，相对于FFX-A2 和 FFX-A10我们选择了更有挑战性的轮数。为了完成上面提到的拓展，我们把FFX方案定义为FFX[radix]。括号里的值可以被称为FFX参数。radix的取值范围介于2到 2^{16} 之间。要在FFX[radix]下进行加密的消息被视为从字母表 $\text{Chars} = \{0, 1, 2, \dots, radix - 1\}$ 中取出的字符串。FFX[radix]方案使用基于AES的平衡Feistel网络结构来完成它的工作。如果消息长度是奇数，则使用一种交替的，最大平衡的Feistel方案。

FFX[radix]模式有效的统一和拓展了FFX-A2 和 FFX-A10；因此我们建议用它代替之前提出的模式。radix的取值变得更普通化，消息长度可以变得更长，还有轮数可以变成常量，而不是像之前一样需要依赖于消息的长度决定。我们并没有尝试去维持向上兼容性；FFX[2]和 FFX[10]与FFX-A2 和 FFX-A10不一致，即使给它们赋予相同的轮数它们也不会一致(尽管这样加密上的区别不大)。

1.2 The Scheme FFX[radix]

我们假设符号注释来自FFX spec[2]，但是为了读者的方便，我们在这里回忆一下相关性最高的定义。明文和密文被看成从字母表 $\text{Chars} = \{0, 1, \dots, radix - 1\}$ 中取出的字符串。我们定义 $|X|$ 为字符串X的长度，即字符串X里的字符个数。让Byte由 $\{0, 1\}^8$ ，这种8-bit字节(8位元组)来表示。通过 $|T|_8$ ，我们把字符串 $T \in \text{Byte}^*$ 的长度定义为用字节计算的长度。基于对radix含义的理解，块相加功能田被定义为 $a_1 \dots a_n \oplus b_1 \dots b_n = c_1 \dots c_n$ 当 $c_1 \dots c_n$ 是满足 $\sum c_i radix^{(n-i)} = (\sum a_i radix^{(n-i)} + \sum b_i radix^{(n-i)}) \bmod radix^n$ 要求的唯一字符串。通过 $X \Leftarrow Y$ 我们认为存在唯一的字符串Z，使 $Y \oplus Z = X$ 。我们认为 $[s]^i$ 是由 $s \in [0 \dots 2^{8i} - 1]$ 编码的长度为i-byte的字符串。功能 $NUM_{radix}(X)$ 需要一个非空字符串 $X \in \{0, \dots, radix - 1\}^*$ 把它转换为相应的数字，这些数字要在给定的radix的解释里，最重要的字节在前。功能 $STR_{radix}^m(x)$ 需

要从 $x \in [0 \dots radix^m - 1]$ 取出一个数字然后返回一个 m 字节的代表给定radix的字符串，最重要的字节在前。

当 $K \in \{0,1\}^{128}$ 和 $X \in \{0,1\}^*$ 以及 $|X|$ 能被128整除， $CBC-MACK_K(X)$ 算法的定义如下。首先让 $X_1 \dots X_m \leftarrow X$ 当 $|X_i| = 128$ 和让 $Y \leftarrow 0^{128}$. 然后，对于 $j \leftarrow 1 \sim m$, 置 $Y \leftarrow AES_K(Y \oplus X_i)$ 。最后，返回 Y 。

10 algorithm FFX.Encrypt(K, T, X)	20 algorithm FFX.Decrypt(K, T, Y)
11 if $K \notin \text{Keys}$ or $T \notin \text{Tweaks}$ or	21 if $K \notin \text{Keys}$ or $T \notin \text{Tweaks}$ or
12 $X \notin \text{Chars}^*$ or $ X \notin \text{Lengths}$	22 $Y \notin \text{Chars}^*$ or $ Y \notin \text{Lengths}$
13 then return \perp	23 then return \perp
14 $n \leftarrow X $; $\ell \leftarrow \text{split}(n)$; $r \leftarrow \text{rnds}(n)$	24 $n \leftarrow Y $; $\ell \leftarrow \text{split}(n)$; $r \leftarrow \text{rnds}(n)$
15 $A \leftarrow X[1.. \ell]$; $B \leftarrow X[\ell + 1.. n]$	25 $A \leftarrow Y[1.. \ell]$; $B \leftarrow Y[\ell + 1.. n]$
16 for $i \leftarrow 0$ to $r - 1$ do	26 for $i \leftarrow r - 1$ downto 0 do
17 $C \leftarrow A \boxplus F_K(n, T, i, B)$	27 $C \leftarrow B$; $B \leftarrow A$
18 $A \leftarrow B$; $B \leftarrow C$	28 $A \leftarrow C \boxminus F_K(n, T, i, B)$
19 return $A \parallel B$	29 return $A \parallel B$

radix	a number $\text{radix} \in [2.. 2^{16}]$	alphabet is $\text{Chars} = \{0, 1, \dots, \text{radix} - 1\}$
Lengths	[minlen .. maxlen] where minlen = 2 if radix ≥ 10 and minlen = 8 otherwise; and maxlen = $2^{32} - 1$.	permitted message lengths
Keys	$\{0, 1\}^{128}$	128-bit AES keys
Tweaks	$\text{BYTE}^{\leq \text{maxlen}}$ where maxlen = $2^{32} - 1$	tweaks are arbitrary byte strings
addition	1	blockwise addition
method	2	alternating Feistel
split(n)	$\lfloor n/2 \rfloor$	maximally balanced Feistel
rnds(n)	10	number of rounds
F	given below	AES-based round function

30 algorithm $F_K(n, T, i, B)$
31 vers $\leftarrow 1$; $t \leftarrow T _8$; $\beta \leftarrow \lceil n/2 \rceil$; $b \leftarrow \lceil \lceil \beta \log_2(\text{radix}) \rceil / 8 \rceil$; $d \leftarrow 4\lceil b/4 \rceil$
32 if EVEN(i) then $m \leftarrow \lfloor n/2 \rfloor$ else $m \leftarrow \lceil n/2 \rceil$
33 $P \leftarrow [\text{vers}]^1 \parallel [\text{method}]^1 \parallel [\text{addition}]^1 \parallel [\text{radix}]^3 \parallel [\text{rnds}(n)]^1 \parallel [\text{split}(n)]^1 \parallel [n]^4 \parallel [t]^4$
34 $Q \leftarrow T \parallel [0]^{(-t-b-1) \bmod 16} \parallel [i]^1 \parallel [\text{NUM}_{\text{radix}}(B)]^b$
35 $Y \leftarrow \text{CBC-MAC}_K(P \parallel Q)$
36 $Y \leftarrow \text{first } d + 4 \text{ bytes of } (Y \parallel \text{AES}_K(Y \oplus [1]^{16}) \parallel \text{AES}_K(Y \oplus [2]^{16}) \parallel \text{AES}_K(Y \oplus [3]^{16}) \dots)$
37 $y \leftarrow \text{NUM}_2(Y)$
38 $z \leftarrow y \bmod \text{radix}^m$
39 return $\text{STR}_{\text{radix}}^m(z)$

图 A-1 FFX[radix]的定义

由字符 $\text{Chars} = \{0, 1, \dots, \text{radix} - 1\}$ 组成的加密字符串。它们用最大的平衡Feistel网络和一个基于AES CBC-MAC的轮函数。我们不重复FFX spec [2]里的更多材料，为了躲着的方便，我们提供了FFX自身的定义，特定是改变Feistel (method =2)。这使得我们的着重点简洁和独立。在图一我们强调FFX和参数集合 [radix]。当FFX被这种参数集合实例化则包含FFX[radix]方案。在图2我们用图表解释了后面的模

式

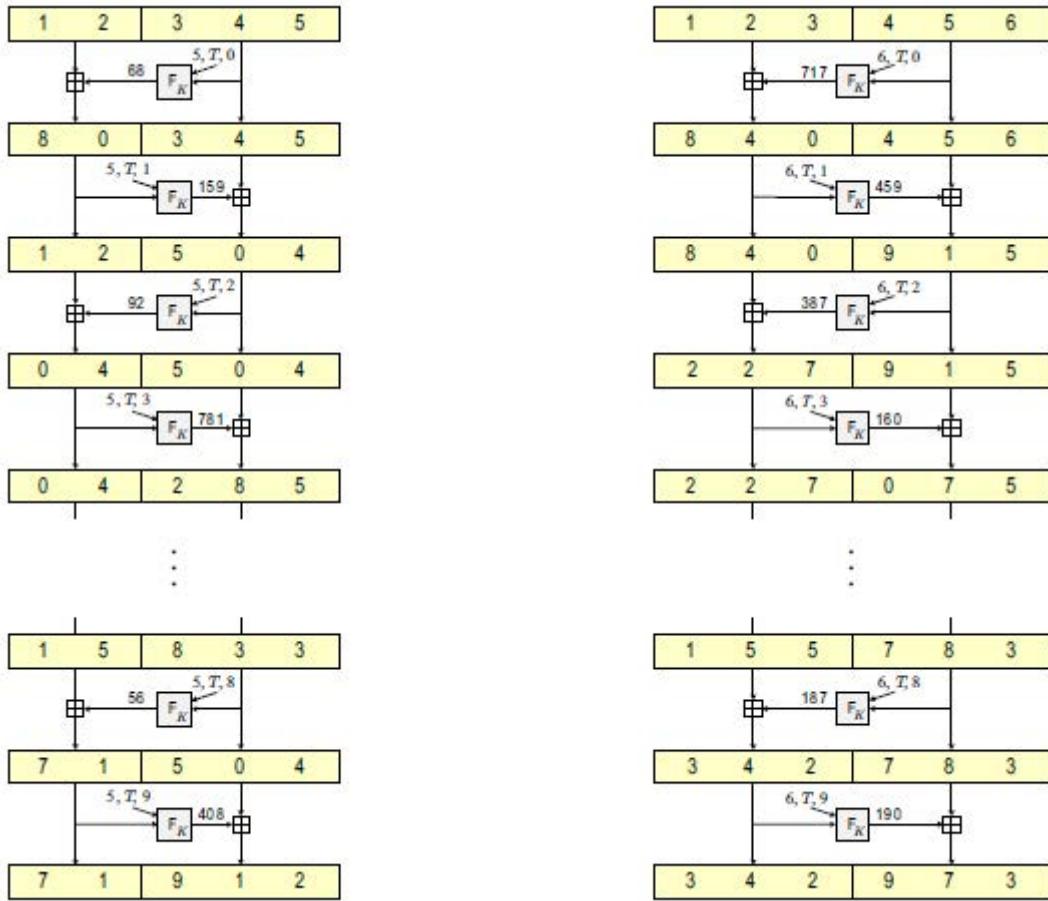


图 A-2 FFX[10]的解释

左边的描绘了奇数长度的输入行为(在这样的方案改动下，使用几乎平衡的Feistel)；右边展示了一个偶数长度的输入(在这样的方案下，使用平衡的Feistel)。理论上的轮功能输出是虚构的；实际的值取决于密钥K和微调T，两个中的任一个都未被指定。

1.3 Comments

关于FFX的一个全面的讨论在说明书中给出，因此这篇文章只是一个附加物[2]；我们在这里把自己限制为一个简短的评论。

首先，我们需要强调的是，当我们被允许去加密非常方的字符串，FFX[radix]模式和真正的FFX通常是打算加密相对较短的字符串的。对于长度超过128 bits的二进制字符串，EME2 [5]或许是比FFX[radix]更好的选择。对于较长的非二进制

字符串，一种模式可以，近似的，比FFX[radix]做得更好，不仅在速度上也在可证明安全性结果上。然而，没有人为这种模式写过一个说明，这是我们选择允许使FFX[radix]来加密即使相对来说较长的串的一个原因。

第二，我们想要简短的讨论一下轮数， $\text{rnds}(n) = 10$ 。我们之前的工作，在A2和A10参数的手机上，已经使用了一个相对复杂的功能(我们把它称为 12^+)在这方面上轮数是超过小轮数r的四倍，因此 $r \cdot \text{split}(n) \cdot \log_2(\text{radix}) \geq 128$;但是至少为12和总是6的倍数。规定是启发式的刺激和极端的保守。对于 $\text{radix} = 10$ 它当 $n \geq 10$ 转化为 $\text{rnds}(n)=12$; 如果 $6 \leq n \leq 9$, $\text{rnds}(n)=18$ 和如果n为4或者5, $\text{rnds}(n)=24$ 。我们获得的反馈以及我们自己的识别都是一个非固定的轮数当消息变短的时候。当为了使得对于这种选择的解释有意义(见FFX说明书[2]上的附件)，我们认为理论上的小号已经太高了。与此同时，没有任何真实的证据证明，一旦有人客服了“中间相遇”攻击，更短的字符串需要更多的轮数。我们早就注意到BPS [4]选择了一个为8的轮数，独立于消息长度。当把这些都纳入考虑范围，我们选出了简化和把轮数减少到10，加速了实现但是留下了一些安全的隐患。这个选择当然没有很保守因为轮数是 12^+ ，但是最终这个机制的设计者一定要做出一个选择，这个决定基于他们对已有知识的判断和加密的艺术。

第三，如同在FFX说明书[2]中讨论的一样，实施的选择将会很大程度的影响表现。比如，第一个AES计算暗示当要计算所有轮的时候第35行需要被执行。当 radix 不是2的幂，第38行的模需要着重的注意。

最后我们再次强调FFX[radix]是FFX的一个实例化版本，以及其他实例化都服从于更高层次的方案。

1.4 References

- [1] M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-preserving encryption. Selected Areas in Cryptography (SAC 2009), LNCS 5867, Springer, 2009. Also ePrint report 2009/251.
- [2] M. Bellare, P. Rogaway, and T. Spies. The FFX mode of operation for format-preserving encryption. Draft 1.1. February 20, 2010. Manuscript, available on the NIST website at URL <http://csrc.nist.gov/groups/ST/toolkit/BCM/mod-development.html>.

- [3] J. Black and P. Rogaway. Ciphers with arbitrary finite domains. RSA Data Security Conference, Cryptographer's Track (RSA CT '02). LNCS vol. 2271, pp. 114{130, Springer, 2002.
- [4] E. Brier, T. Peyrin, and J. Stern. BPS: a format-preserving encryption proposal. Available at URL <http://csrc.nist.gov/groups/ST/toolkit/BCM/modes development.html>. Undated manuscript, published on the above website in April 2010.
- [5] IEEE P1619.2. Draft standard architecture for wide-block encryption for shared storage media. 2008. Available from <https://siswg.net>.