**基于可搜索加密的云存储方案**

雷云霖 何祎军 徐琳锴

**Abstract：** 随着云计算的高速发展，其廉价、按需服务、高可靠性的特点吸引了越来越多的用户将自己的数据存放在云端。但在实际应用场景中，数据脱离用户的控制后会带来安全隐患以及传统加密算法会给用户搜索等操作带来的不便。因此，上传到云服务器中的数据的安全问题引起我们的关注。为了解决这个问题研究者们提出了可搜索加密算法。但目前可搜索加密算法处于研究阶段，商业化以及实用案例非常少，没有适合当今功能复杂的云计算平台的部署解决方案。我们基于现有可搜索解密算法的改进，使得用户在使用尽量小的网络开销以及本地存储开销的情况下，对自己数据加密以及按关键词搜索。我们并且对不同用户数量、不同用户身份、不同云存储服务等多种实际应用场景提出了定制化的可行可搜索加密云存储部署解决方案。

**Index Terms：** 可搜索加密，云存储

**1．Introduction**

在信息时代，数据的产生源多元化以及数据量的迅速增长，这已经超出了一般传统软件的能力范畴。而部署大型服务器集群成本高昂、运维复杂，中小型企业以及个人用户没有足够的资源与技术支撑部署。于是云计算在这种需求下迅速发展。云计算的廉价、按需服务、高可靠性、便利性等特点使得越来越多的用户、企业选择将应用部署在云端，并且将数据存储在云端，以此来节省本地数据管理开销和系统维护开支，这些数据可能包括了通讯录、网站账号、操作日志等敏感信息。但在实际应用中由于数据脱离了用户的物理控制，云端服务器管理员和非法用户（如黑客）可以在未经用户授权的情况下访问数据，以及云服务提供商的安全措施不够，这都将造成数据信息和用户隐私的泄露。这些数据的泄露将会给用户带来极大的灾难。

在现实生活中云中数据泄露的案例并不少见。例如，2018年12月10日，Google公司宣布发现其社交平台Google+存在漏洞，这将可能导致超过5200万用户的数据泄露。2018年3月曝光的Facebook“数据门”事件，据报道数据分析企业剑桥分析（Cambridge Analytics）未经授权就获取了Facebook多达5000万名用户的信息，用以预测和影响了英国脱欧和美国大选中的选民投票行为。2017年6月，由于和美国共和党签约的Deep Root Analytics公司将内部重要文件保存在亚马逊的云服务器中，并且亚马逊云服务器的配置错误，导致了近2亿美国公民的政治数据泄露。这些数据中除了家庭地址、生日和电话号码等个人隐私数据外，还包含了政治团体采用先进分析机制预测出的大量数据资料，同时该事件还泄露了共和党超级政治行动委员会的绝密资料。云中数据泄露事件仍然在不断的发生，这些泄露事件不仅严重威胁了个人隐私，也对公司的企业形象产生了巨大的影响，甚至还会对国家的安全产生威胁。通过这些事件可以看出云中数据的安全越来越重要，吸引了越来越多研究者展开对云中数据安全的研究。

保证云中数据安全最直接的方案就是对数据进行加密存储，并且随着现代密码学的发展，当今主流加密算法的安全性很高。在这种方案下就算数据泄露了，由于缺乏密钥，未获得授权的用户不能获得明文数据，这有效的保证了数据的安全性。但是在当今主流加密算法（如DES,RSA算法）的方案下，用户搜索数据将变得非常困难，当用户想获得部分数据时只能下载所有数据并在本地解密后进行搜索，这将和存在本地没有什么区别。而且当今大量应用都是针对移动设备，移动设备的存储能力、计算能力都十分有限，极有可能其性能不足够支持这种操作。同时对于中小型企业，也没有足够的资源部署服务器或者私有云在本地处理、分析对自己用户的操作数据。这种方案并不能完全满足实际应用。

为了很好地解决这个问题，基于可加密搜索(searchable encryption，SE)的云存储是较好的选择。在可搜索加密机制下用户需要搜索某关键词时将该关键字的搜索凭证发给云服务器，云端将接收到的搜索凭证对数据进行试探匹配并返回结果。用户接收到搜索结果后，对结果进行解密即可。在这种机制下，云端服务器将无法获得除访问模式以外的任何明文数据，可以保证安全性。对于不同应用场景，研究者们提出了对称可搜索加密（Symmetric searchable encryption，SSE）和非对称可搜索加密（Asymmetric searchable encryption，ASE）。虽然可搜索加密算法较成熟了，但仍只是处于研究阶段，商业化程度低。实际部署案例非常少，并不能完全适应当今云计算提供的功能强大而又复杂的云存储服务。

我们基于阿里云平台提供的云服务器ECS服务，以及云数据库、NAS存储服务器、OSS对象存储服务器、分布式存储等云存储服务，提出了针对单用户、多用户、企业用户的记录数据、大量文件，以及在数据私有、数据共享等场景下，提出了定制化的可搜索加密云存储解决方案。在使用尽量小的网络开销以及本地存储开销的情况下，支持对敏感数据的关键字查询，并在算法上创新以提高通用性。

我们从搜索时间、搜索命中率、网络开销、数据安全性这几个指标进行测试。我们收集了几个涉及敏感数据的公开数据用于对解决方案实用性进行测试。同时使用阿里云提供的云监控服务对我们的云服务器运行情况进行监控。

这篇文章组织如下，第二部分将介绍可搜索加密的研究现状和相关理论基础。和我们方案的创新点。第三部分将介绍我们针对不同应用场景提出的方案。第四部分将介绍我们对方案的测试结果，并分析我们方案的有效性。

**2．Related Works**

**2.1 研究现状**

可搜索加密技术总体上可以分为对称可搜索加密和非对称可搜索加密两大类。

对称可搜索加密常用于数据私有的场景下，只有一个数据读/写者。第一个对称可搜索加密方案是Song等人[2]在2000年提出的，这也是首次提出可搜索加密的概念。在这个方案中，用户将明文文件划分为单词，并用伪随机函数和确定性加密算法生成关键字密文，在搜索时遍历所有密文来确认所搜关键字是否存在，其搜索开销和文件的大小呈线性关系，效率不够。为了提高效率，Goh等人[3]提出通过建立关键词索引并引入布隆过滤器（Bloom filter）来提高效率。我们基于这种思路，对数据按重要性进行分类并建立索引进一步提高效率，来达到较高效的可搜索加密。

非对称可搜索加密一般应用在多个数据读/写者的应用场景中，数据拥有者将自己的公钥公布，其他用户通过公钥加密数据并上传服务器，然后只有数据拥有者能通过自己的私钥解密数据。非对称可搜索加密主流算法PEKS是Boneh等人[4]在2004年针对邮件加密检索提出的。随后，Abdalla等人[5]对PEKS理论进行了完善，Rhee等人[6]基于前人的工作提出了一个安全性非常好的方案。我们基于这种方案，结合传统非对称加密算法RSA将文件数据加密存储并部署在云中，这样不需要对所有数据进行可搜索加密，只是对文件索引进行可搜索加密，提高了效率。

可搜索加密在云中的应用也引起了研究者们的注意，2010年Seny Kamara等人[1]就提出了可搜索加密在云中的应用场景，并给出了理论模型。但是没有给出具体部署方案，并且随着云计算的发展，云服务提供商开始提供更丰富的云存储服务，如OSS对象存储、云数据库、NAS等。并没有证明

**2.2 对称可搜索加密（Symmetric searchable encryption，SSE）**

定义在字符文件上的对称可搜索加密算法可以描述成如下形式：

其中

。

4)Search():云服务器进行如下计算

这个算法的安全性在[7]中给出了。

**2.3 非对称可搜索加密（Asymmetric searchable encryption，ASE）**

**2.3.1 双线性对**

设表示三个乘法循环群，它们的阶均为大素数q，群的生成元分别为。一个双线性映射应该具备以下的性质：

1) 双线性性：对于任意的成立。

2) 非退化性：存在,使得。

3) 可计算性：对于任意的,存在高效的算法计算。

如果，我们称该双线性对是对称的。同时，也可以是加法循环群。

**2.3.2 基于PEKS的ASE**

PEKS下的ASE可以描述如下：

其中

此算法安全性证明在[8]中给出。

**2.3.2 PBC库**

PBC库(Pairing-Based Cryptography Library)是斯坦福大学研究人员开发的一个免费可移植C语言库，同时维护python版本。这个库提供了密码体制底层的数学运算，例如椭圆曲线生成，椭圆曲线算法和双线性配对计算等程序。我们在实现PEKS中双线性对配对算法直接调用此库中的函数进行运算。

**3．Implementation**

通过对现实生活场景的抽像，我们找到了五个需要应用可搜索加密云存储的场景。根据用户身份以及用户数量分类有：

1、单用户存放数据，并且数据私有。

2、单用户文件系统，并且数据私有。

3、单用户文件系统，并且数据与多用户共享。

4、中小型企业存放应用在多用户使用下产生的数据，数据私有。

以上这四个模型是我们利用云存储存储数据的几个基本应用场景，同时我们还可以根据云存储服务的不同分类，有：

1、直接存储在服务器磁盘中

2、存储在云NAS存储服务器中

3、存储在云0SS对象存储服务中

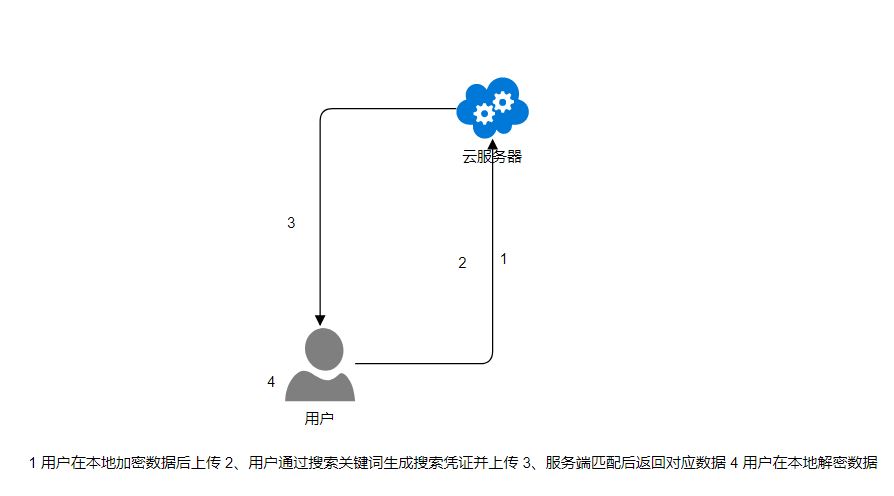
4、存储在云数据库中

下面我们将分别详细介绍如何在这些场景中部署可搜索加密云存储。

**3.1 场景1：单用户存放数据，并且数据私有**

在这个场景下，用户只是单纯地使用云存储来存储自己的数据，以节省自己本地的存储开销，并且这些数据都应该视为隐私数据，不能对外公布。

在这个场景下用户在自己本地生成SSE加密密钥。然后对需要加密的文件进行单词划分，并建立索引。然而对于一般数据我们并不需要对所有数据进行单词划分，对常用SSE进行改进。我们可以选取具有索引性质的数据进行划分，如通讯录中人名、邮件中发件人邮箱。如果没有这样具有索引性质的数据，可以根据在数据中添加这样一个编号数据，将编号和数据建立映射关系，这样就可以对编号进行单词划分。将单词划分的结果用密钥进行加密，之后将加密数据上传到云服务器中。当用户想要获得部分数据时，输入自己需要搜索的关键词，在本地计算机通过SSE密钥和关键词生成搜索凭证，发送到云服务器中。云中的服务器程序收到请求后和自己密文通过SSE进行匹配，将匹配成功的结果返回给用户。用户收到服务器返回的结果在本地通过密钥解密获得数据。

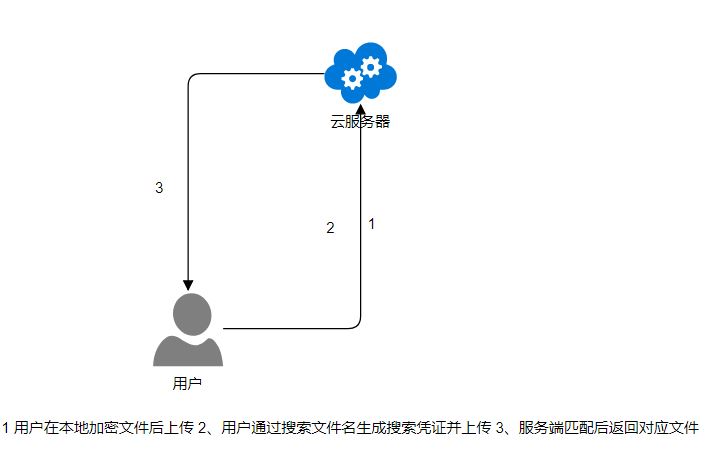


*场景1*

**3.2 场景2：但用户存储文件系统，并且数据私有**

考虑更一般的应用场景，用户将自己大量文件备份在云中，并考虑这些数据不进行公开。然后用户有下载某个文件下需求。

在这种场景下，为了提高效率，不应该对所有数据进行可搜索加密。而是用户将自己的文件名提取出来建立索引表，考虑到目录结构，文件名应该使用绝对路径，对于文件夹同样提取文件名。对这个索引表在本地进行SSE加密，并重命名加密文件的文件名，将并索引表上传云服务器，对文件内部数据不使用SSE加密，而是使用传统对称加密算法，如DES算法。加密完成后将文件打包上传到服务器中。当用户需要下载某个文件时，将要下载的文件名和SSE密钥通过计算生成搜索凭证，并发送到云服务器。云服务器收到请求后在索引表中进行匹配，匹配成功后将对应文件发送给用户。用户收到文件后在本地对这个文件进行解密。

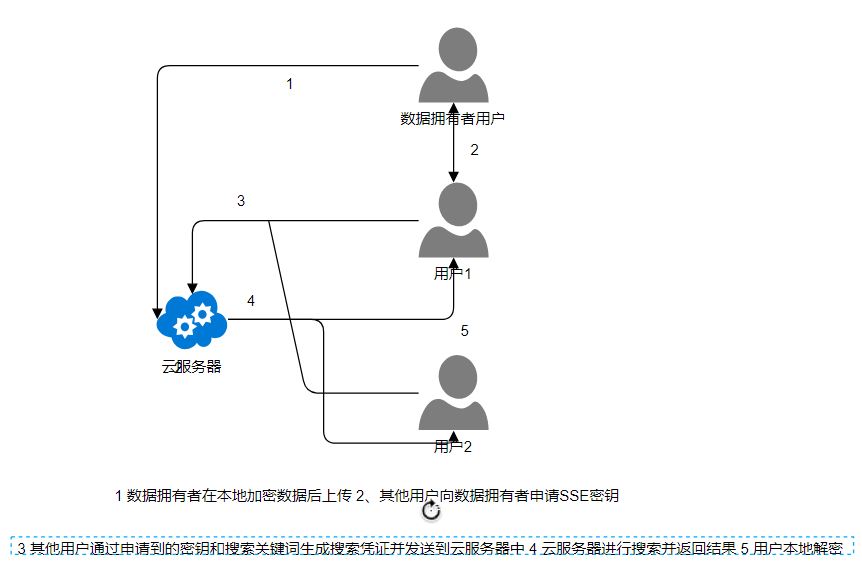


*场景2*

**3.3 场景3：单用户存储文件系统，并且数据与多用户共享**

针对场景2的扩展，用户将自己的大量文件加密存储到云端，同时当其他用户需要时可以供他人下载。

这种场景下我们应该使用非对称可搜索加密ASE，首先用户通过输入安全系数获得ASE的密钥对。和场景2的操作方法一样，建立文件的索引表，并用自己的私钥对这个表进行加密。然后对文件数据，使用传统的非对称加密算法进行加密，如RSA算法。加密完成后，用户将加密后的文件和索引表上传到云服务器中。当用户想要自己下载文件时，操作同场景2，获得云服务器返回结果就用自己的ASE公钥和RSA私钥分别对文件名和文件内容进行解密。当其他用户想要下载此用户文件时，此用户分享自己的ASE公钥和RSA公钥，其他用户进行相同操作获取到云服务器中的文件并解密。



*场景3*

**3.4 场景4：中小型企业存放应用在多用户使用下产生的数据，并且数据私有**

考虑中小型企业将应用部署到云中的应用场景。中小型企业利用公有云低成本、按需服务、弹性计算等便利处将自己开发的应用部署在云服务器中，但其用户产生的数据，如日志文件，账号表单等，涉及到商业机密或者具有很大的商业价值，为了数据安全，不能明文存储在云中。

为了在这个场景中部署，我们也应该使用ASE进行部署。为了保证企业服务的安全性、公正性和规范性，我们应该引入证书颁发机构，这个机构致力于给企业颁发数字证书，这个数字证书中包含了ASE和RSA这两个算法的两个密钥对。企业先向证书颁发机构申请数字证书。企业获得证书后，将自己的ASE公钥和RSA公钥公布。并将应用中向服务器发送数据时，将文件名用ASE公钥加密，文件内容用RSA公钥加密。当企业将要获得部分数据时，通过自己的私钥和想要下载的文件的文件名生成搜索凭证，将搜索凭证发送到云服务器，服务器收到后进行匹配，并将匹配成功的文件发至企业用户本地。企业拿到数据后用自己的ASE私钥对文件名进行解密，用RSA私钥对文件内容进行解密。

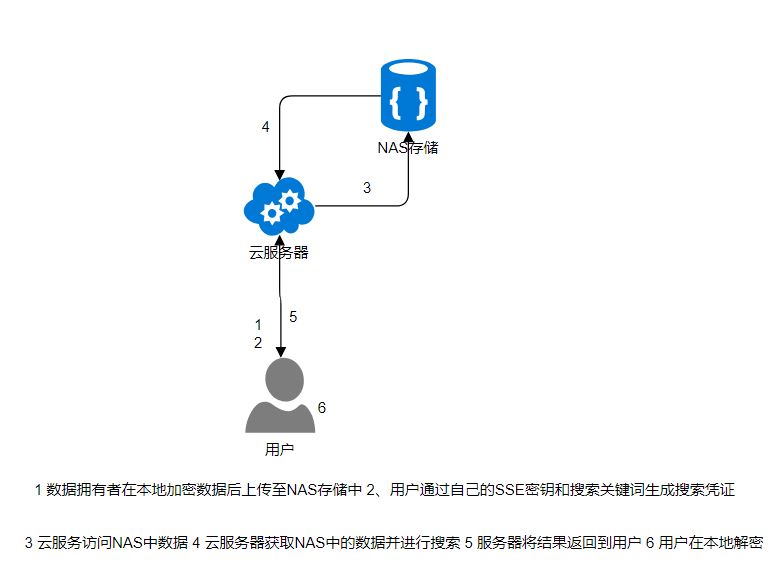


*场景4*

**3.5 场景5：数据存储在云NAS文件服务器中**

NAS(网络附加存储，Network Attached Storage)是一种专门的数据存储技术的名称，它可以直接连接在计算机网络上面，对不同网络的用户提供了集中式数据访问服务。和传统的文件存储服务不同，NAS服务器上的值提供了数据存储、数据访问、以及相关的管理功能。云NAS具有成本低、可靠性高、易用性高的优势。所以很多用户选择将大数据存放在云NAS服务中。

由于云NAS服务的遍历性，提供了非常方便的接口和操作。我们只需要将云NAS服务挂载在云服务器上，然后将数据存放在NAS中即可。场景1、2、3的应用场景都可以将数据部署在NAS中，数据加/解密、搜索等过程不变。我们针对场景1，给出示意图。

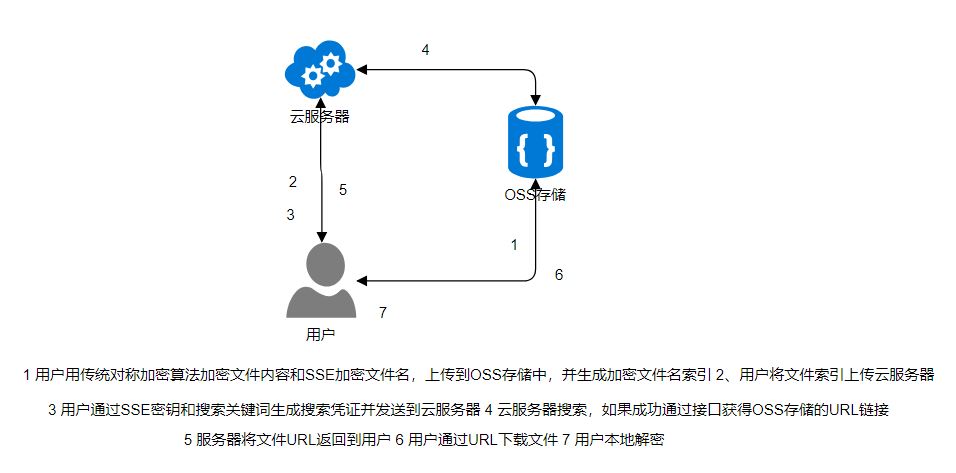


*场景5*

**3.6 场景6：数据存储在OSS对象存储中**

云对象存储服务（Object Storage Service，简称 OSS），是具有海量、安全、低成本、高可靠特点的云存储服务。它具有与平台无关的RESTful API接口，可以在任何应用、任何时间、任何地点存储和访问任意类型的数据。云OSS服务一般作为移动应用、大型网站、图片分享或热点音视频的存储方式。

对于这种服务，数据下载将不是有与服务器发送给你，而是通过OSS服务提供的URL链接进行下载。我们设计将文件内容通过传统对称加密算法加密，将文件名通过SSE加密。然后利用SSE建立索引，并用SSE加密上传至云中。当用户想下载某个文件，将文件名和SSE密钥计算获得可搜素凭证并发送给云服务器。云服务器获得可搜索凭证后进行搜索，如果寻找到匹配的文件，通过OSS服务的API生成文件下载的URL并发送到用户，用户收到URL后进行下载并解密。

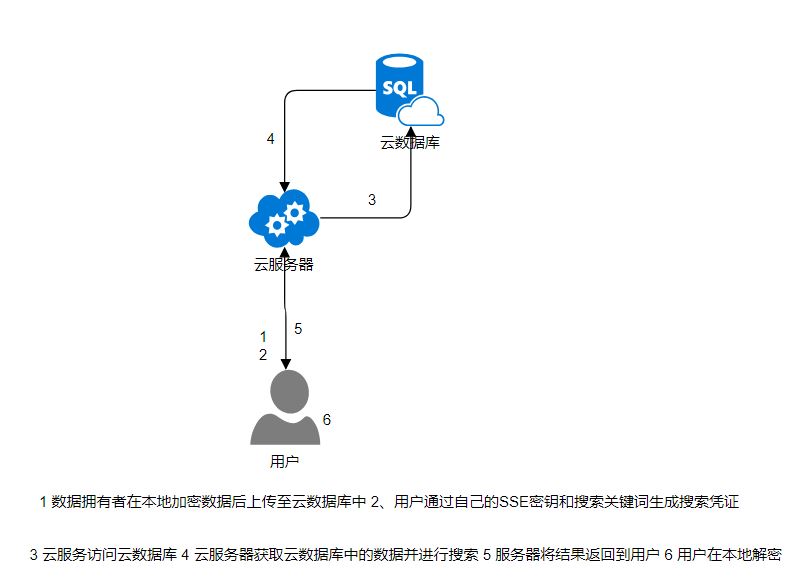


*场景6*

**3.7 场景7：数据存储在云数据库中**

数据库是现在最主要的存储数据方式。云服务提供商也提供云数据库，使数据库脱离云服务器的磁盘，可以获得更大的存储空间，并拥有按需服务、成本低的优势。

由于云数据库提供接口的完整性，对云数据库的访问和访问本机数据库一样的方便。所以可搜索加密的云存储部署很遍历，场景1、2、3中的应用场景都可以将数据存放在云数据库中。我们根据场景1提供示意图。



*场景7*

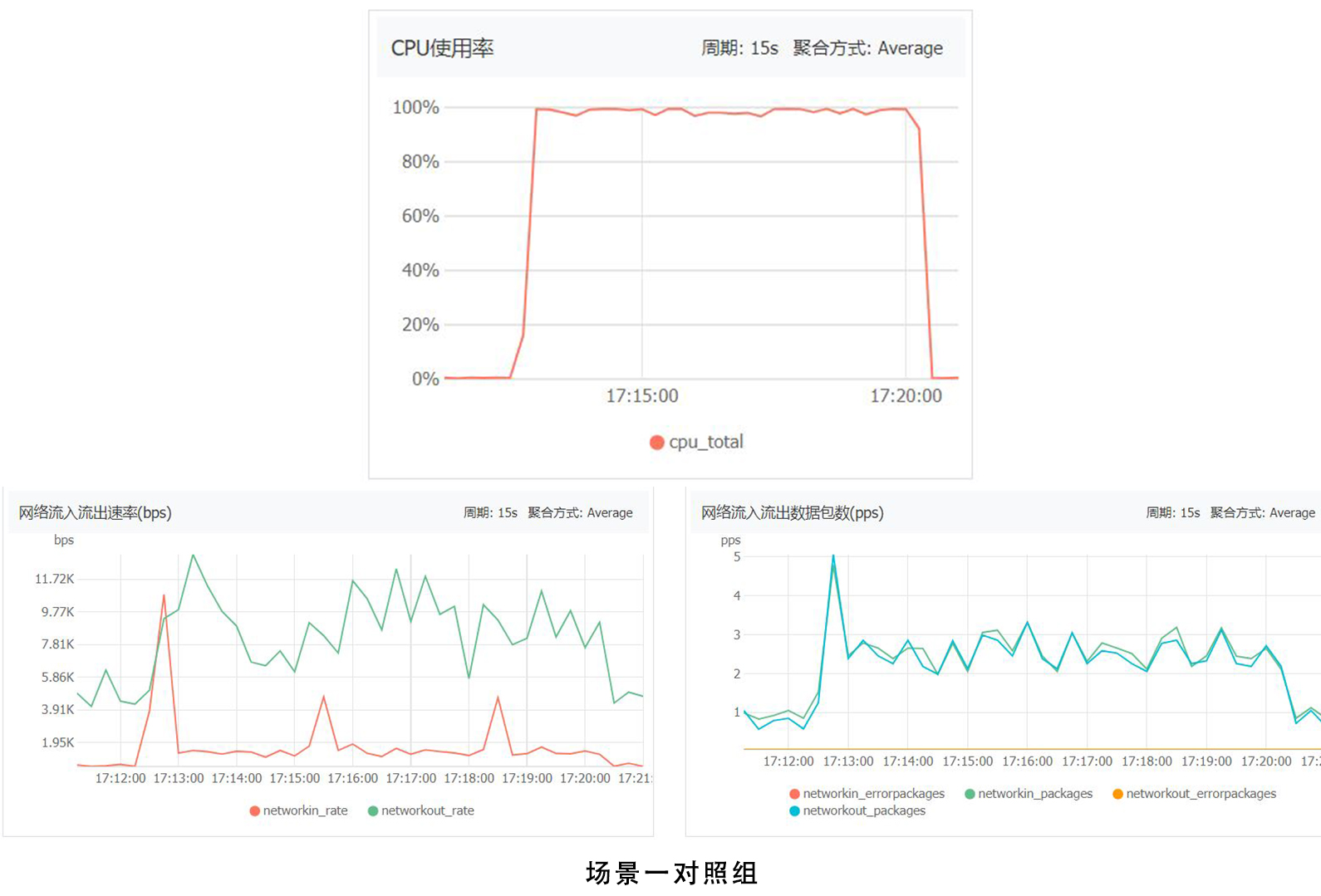
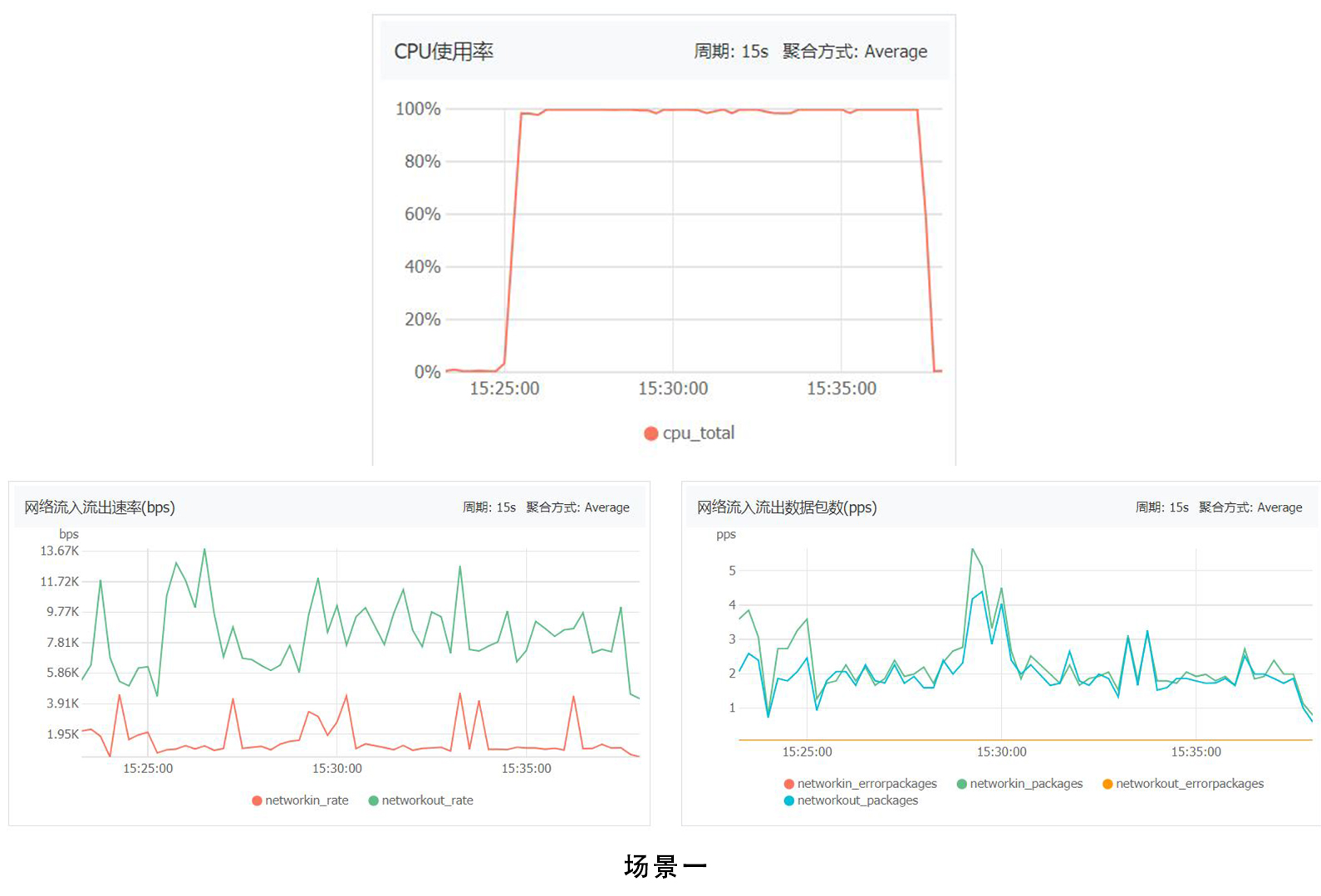
**4．Evaluation**

我们在阿里云上开通ECS服务器服务，服务器配置是1vCPU，2G内存，1M宽带。由于资金有限，服务器配置很低，只是用于验证我们的方案有效性，并不能进行商业化。对于数据，我们找到了Airbnb公布了其处理过的用户信息，包括时间、用户ID、姓名，我们将用这个模拟隐私数据的存储。同时我们构造了6个英文短篇小说的txt文件用于模拟文件存储。我们的客户端、服务器端、加/解密程序、证书颁发程序都是通过python语言开发，需要pypbc、Crypto两个python库。我们针对每个场景都设置了明文存储的对照组，通过阿里云提供的云监控服务确定方案的搜索时间、网络开销，我们自己实现了一个比较脚本来判断搜索命中率。

**4.1 场景1：单用户存放数据，并且数据私有**

因为服务器性能原因，我们测试了100次请求。对照组明文存储，搜索使用线性搜索。

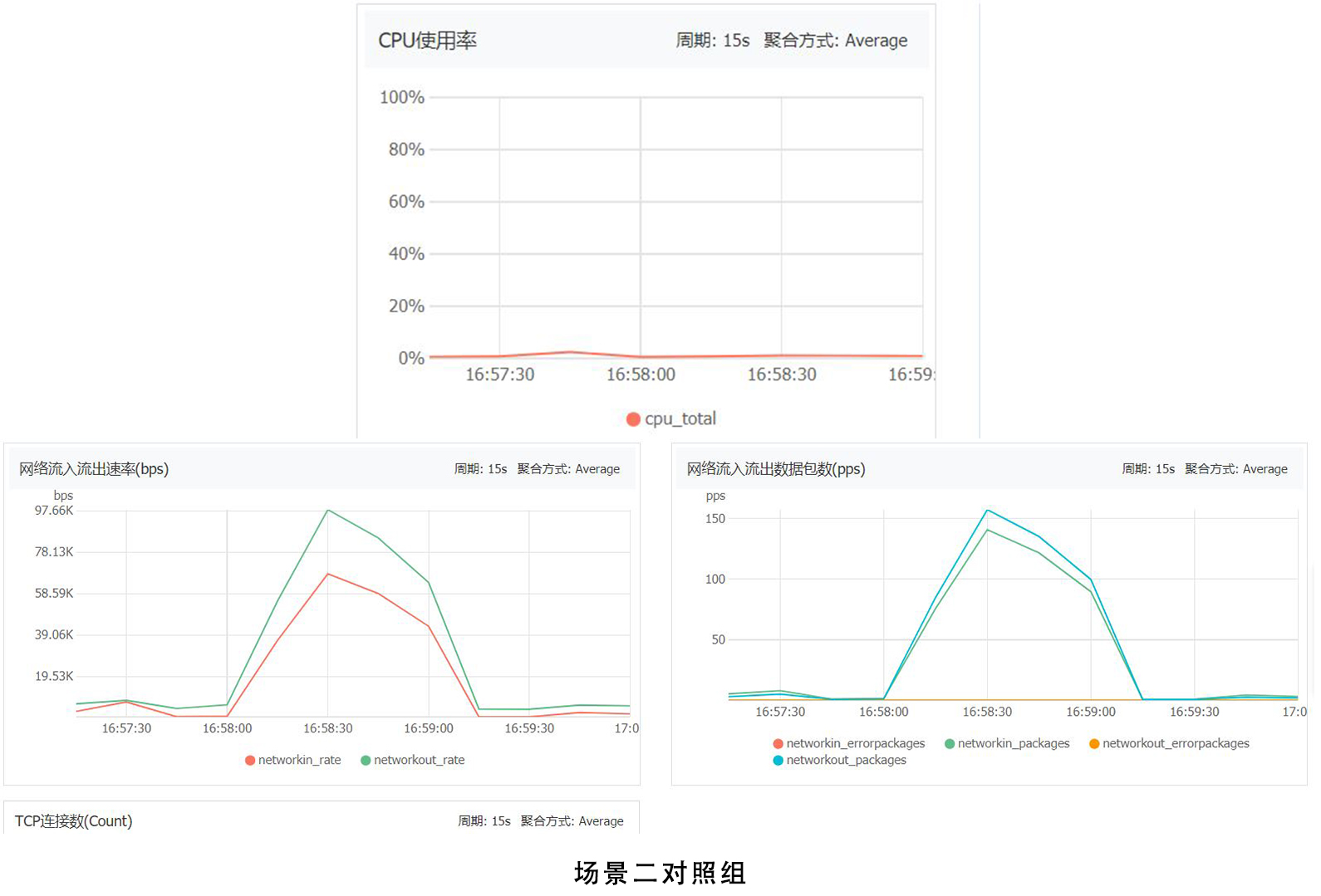
结果测试组程序运行了730s，对照组程序运行了451s。平均每次请求慢2s，原因预测是加解密运算涉及数学运算，单核通用型CPU在这上面并没优势，如果使用计算型CPU会极大提高速度。网络开销接近。说明在网络开销上此方案和明文存储一致，远好于传统加密。搜索命中率100%，保证数据的正确性。



**4.2 场景2：单用户存储文件，并且数据私有**

我们测试了1000次请求。对照组明文存储，无搜索过程，直接返回对应文件。

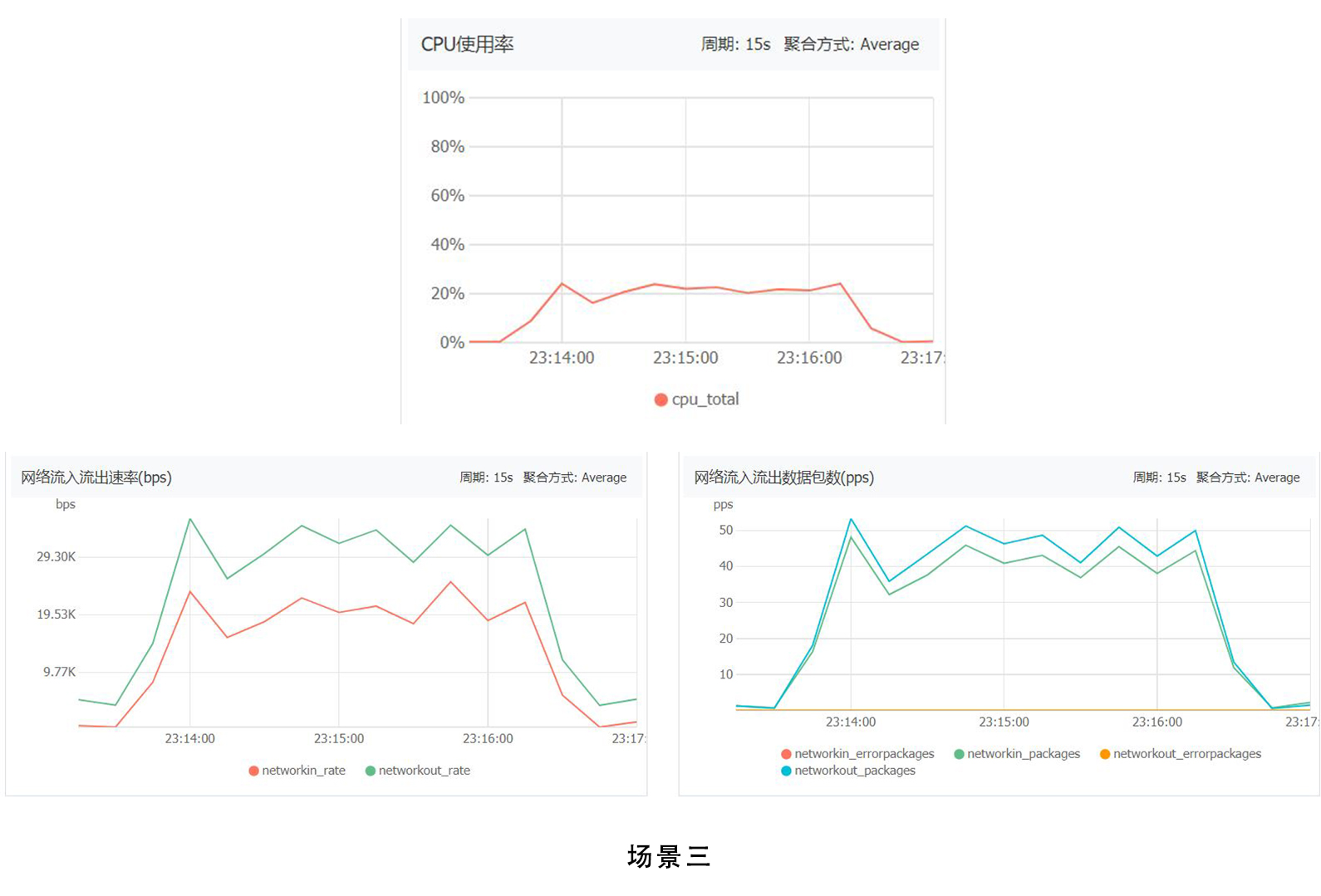
结果测试组程序运行了46s，对照组程序运行了49s。平均每次请求慢3ms，在运行速度上非常接近，说明我们的改进算法搜索效率非常高。网络开销接近。说明在网络开销上此方案和明文存储一致，远好于传统加密。搜索命中率100%，保证数据的正确性。



**4.3 场景3：单用户存储文件系统，并且数据与多用户共享**

我们测试了1000次请求。对照组明文存储，无搜索过程，直接返回对应文件。

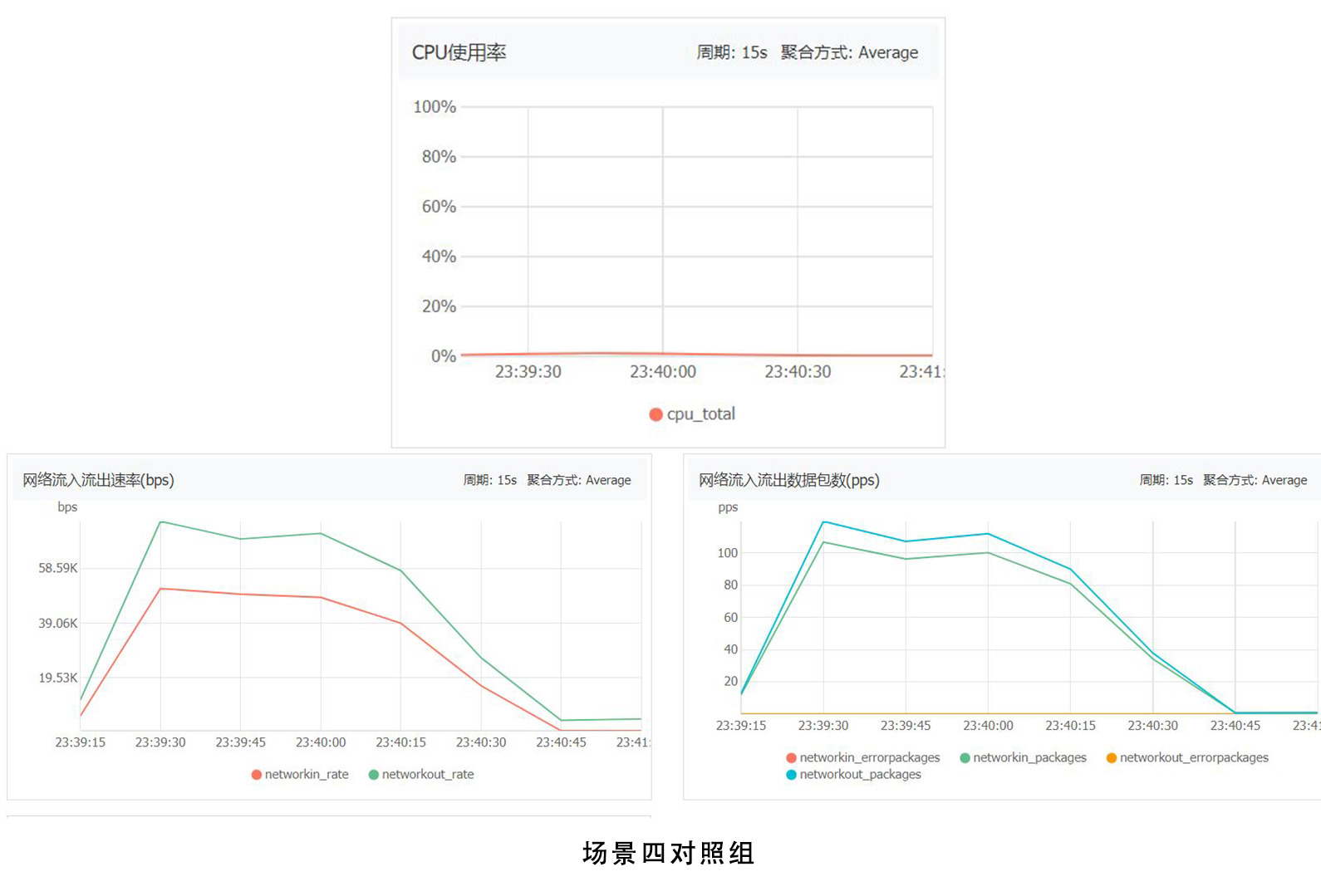
结果测试组程序运行了159s，对照组程序运行了105s。平均每次请求慢50ms，在运行速度上非常接近，说明我们的改进算法搜索效率非常高。和场景2比较因为多了一次请求，所以所需时间变长。网络开销接近。说明在网络开销上此方案和明文存储一致，远好于传统加密。搜索命中率100%，保证数据的正确性。



**4.4 场景4：中小型企业存放应用在多用户使用下产生的数据，并且数据私有**

我们测试了1000次请求。对照组明文存储，无搜索过程，直接返回对应文件。

结果测试组程序运行了128s，对照组程序运行了105s。平均每次请求慢20ms，在运行速度上接近，结合场景2、3可以发现，我们算法在存储文件时效率很好，有商业潜力。网络开销接近。说明在网络开销上此方案和明文存储一致，远好于传统加密。搜索命中率100%，保证数据的正确性。



**4.5 场景5：数据存储在云NAS文件服务器中**

我们测试了100次请求。对照组明文存储，搜索算法是线性搜索。

结果测试组程序运行了719s，对照组程序运行了442s。平均每次请求慢2.7s，。说明在网络开销上略高于明文存储，因为需要访问NAS服务，但远好于传统加密。搜索命中率100%，保证数据的正确性。



**4.6场景6：数据存储在OSS对象存储中**

我们测试了1000次请求。对照组明文存储，无搜索过程，直接返回对应文件。

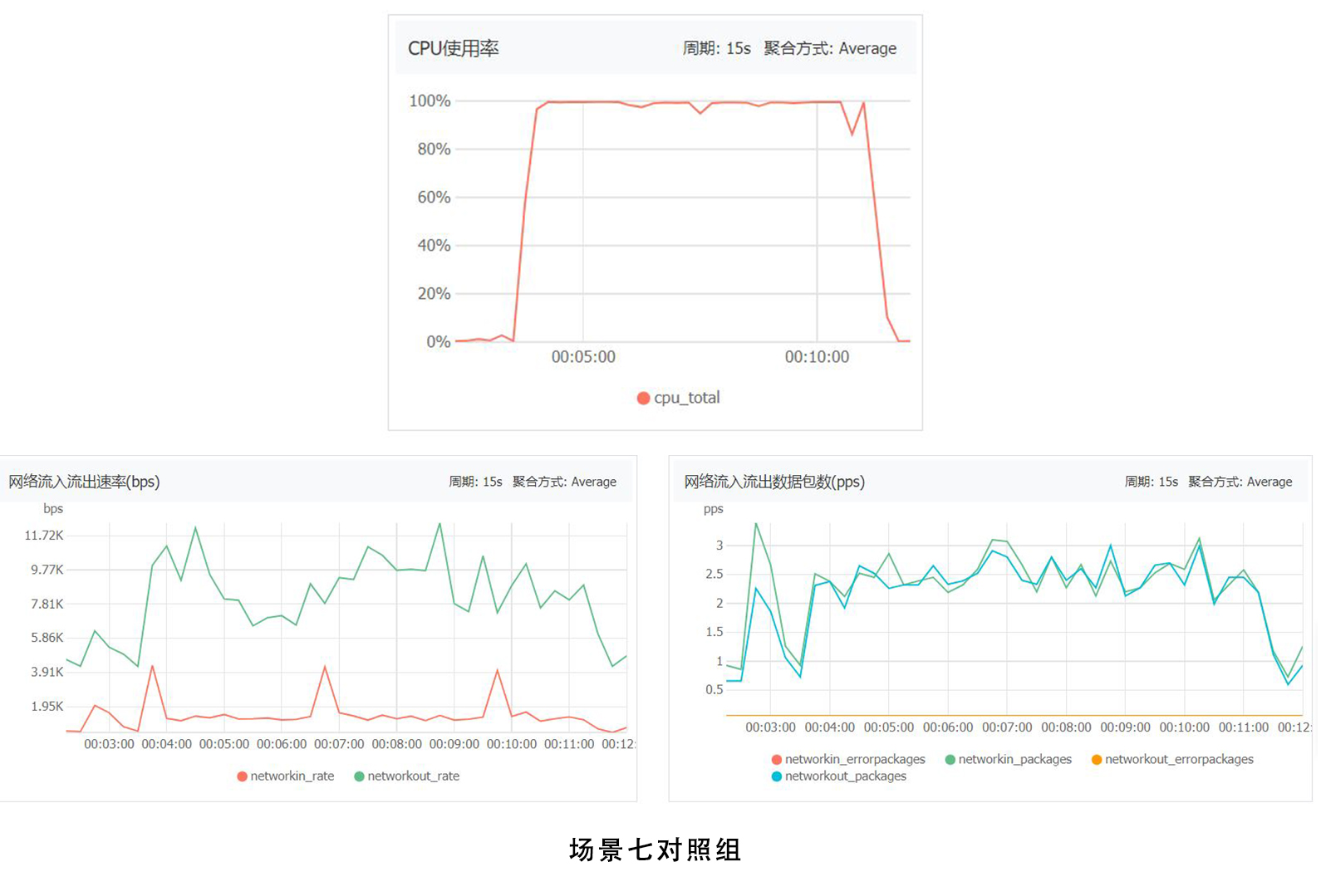
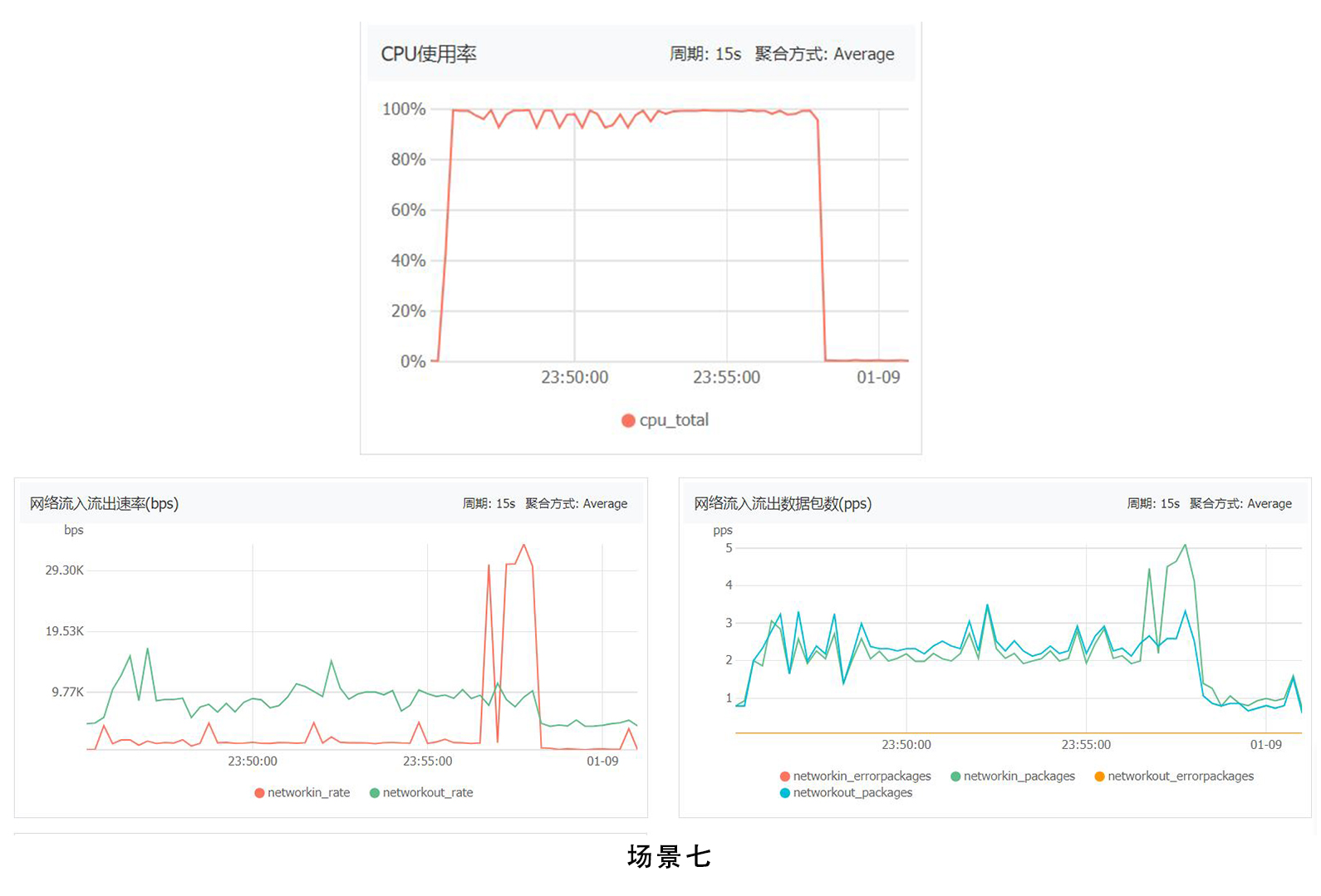
结果测试组程序运行了121s，对照组程序运行了69s。平均每次请求慢59ms，在运行速度上非常接近，但比直接存储在服务器中慢，因为用户还需要访问OSS服务器，但是OSS服务器可以用CDN加速来提高速度。不过仍慢于NAS存储，说明在访问速度上，NAS服务好于OSS。网络开销明文高于OSS，因为文件存放在OSS中，服务器不需要返回文件。搜索命中率100%，保证数据的正确性。



**4.7 场景7：数据存储在云数据库中**

我们测试了100次请求。对照组明文存储，使用线性搜索。

结果测试组程序运行了741s，对照组程序运行了461s。平均每次请求慢2.8ms。网络开销接近。说明在网络开销上此方案和明文存储一致，远好于传统加密。搜索命中率100%，保证数据的正确性。



**5．References**

[1] Kamara, S., & Lauter, K. (2010, January). Cryptographic cloud storage. In *International Conference on Financial Cryptography and Data Security* (pp. 136-149). Springer, Berlin, Heidelberg.

[2] Song, D. X., Wagner, D., & Perrig, A. (2000). Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on* (pp. 44-55). IEEE.

[3] Goh, E. J. (2003). Secure indexes. *IACR Cryptology ePrint Archive*, *2003*, 216.

[4] Boneh, D., Di Crescenzo, G., Ostrovsky, R., & Persiano, G. (2004, May). Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques* (pp. 506-522). Springer, Berlin, Heidelberg.

[5] Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., ... & Shi, H. (2005, August). Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *Annual International Cryptology Conference* (pp. 205-222). Springer, Berlin, Heidelberg.

[6] Rhee, H. S., Park, J. H., Susilo, W., & Lee, D. H. (2010). Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software*, *83*(5), 763-771.

[7] Curtmola, R., Garay, J., Kamara, S., & Ostrovsky, R. (2011). Searchable symmetric encryption: improved definitions and efficient constructions. *Journal of Computer Security*, *19*(5), 895-934.

[8] Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., ... & Shi, H. (2005, August). Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *Annual International Cryptology Conference* (pp. 205-222). Springer, Berlin, Heidelberg.