针对云中键值存储的众包数据完整性验证

Grisha Weintraub以色列开放大学 数学与计算机科学系

Ehud Gudes 以色列内盖夫贝尔谢巴本古里安大学 计算机科学系

摘要 ： 由于其高可用性，可扩展性和可用性，云数据库已成为主要的云服务之一。但是，由于云用户在物理上不拥有其数据，因此数据完整性可能存在风险。在本文中，我们提出了一种新的协议，利用众包在键值云数据库中提供实用的数据完整性保证。我们的协议相对于以前的工作的主要优点是其高适用性 ，与现有方法相反，我们的方案不需要在云端进行任何系统更改，因此可以直接应用于任何现有系统。我们通过原型实现及其评估证明了我们的方案的可行性。

关键词：云计算， NoSQL，键值存储，数据完整性，安全存储;

一、介绍

关系数据库管理系统（RDBMS）不再是数据存储的唯一解决方案。近年来已经开发了大量非关系数据库，以便管理不断增长的数据量。这些系统的主要特征，通常称为NoSQL [1,2]，具有灵活的模式，水平扩展和宽松的一致性。它们通常跨数据中心在分布式系统中存储和复制数据，从而实现可扩展性和高可用性。 NoSQL数据库通常根据其数据模型分为三组：

•键值存储：数据存储为键值对，使得键是唯一标识符，值是任意条目。

•基于文档的存储：数据存储在类似文档的结构中，例如JSON或XML。

•面向列的存储：数据按表组织，包括行键和列键。列键被分组为称为列族的集合。

在本文中，我们将重点关注第一组——键值存储。

键值存储可用作内部数据库系统或云数据库-云服务，可为用户提供数据访问，而无需管理硬件或软件。但是，将数据存储在云中会引入一些安全问题。尤其是云用户没有物理上的数据，数据完整性可能

处于危险之中。云提供商（或某些恶意实体）可以更改用户的数据，从查询结果中省略一些数据或返回不是最新版本的数据。 其他文字，数据正确性，完整性和新鲜度可能妥协。外包关系数据库中的数据完整性问题已经被研究了几年[13-16,18-21]。 然而，现有的解决方案不适合用于键值存储。

原因如下：

•键值存储中的数据量预计比RDBMS中的数据量高得多，因此数据（通常）分布在许多不同的节点上。

•键值存储的查询模型比RDBMS简单得多。

RDBMS和键值存储之间的这些差异带来了挑战和机遇。一方面，由于其分布式架构和高数据量，键值系统中的数据完整性保证需要更复杂的解决方案。另一方面，它们极其简单的查询模型可以让我们设计更简单有效的数据完整性验证协议。我们特别感兴趣的是开发一种不需要服务器端修改的方法，使它可以无缝地应用于现有的真实世界系统。

本文的目的是证明无需服务器端修改和合理的性能开销即可验证云中键值存储的数据完整性。我们的主要贡献如下：

•开发一种新的概率方法，允许用户验证云密钥值存储及其分析中数据的完整性。

•通过原型实施及其实验评估证明我们方法的可行性。

论文的其余部分结构如下：第二节概述了系统和威胁模型。第三节介绍了我们的数据完整性验证方法。第四节介绍了我们方法的安全性分析。第五节介绍了我们的概念验证实现，并对其进行了实验评估。第六节回顾相关工作。第七章总结全文。

二、系统和威胁模型

A.系统模型和假设

数据库即服务范式（dbaas）在十多年前首次引入[3]，已成为当今云提供商的一种流行服务。Micorosoft的Azure SQL数据库，亚马逊的DynamoDB和Google的Cloud BigTable就是一些例子。

我们假设DBAS模型中有三个实体:

•  *数据所有者（DO）：* 将数据上传到云。

•  *云提供商（CP）：* 存储并提供对该服务的访问 数据。

•   *客户端：* 从云中检索数据。

在我们的模型中只有一个DO和CP实例，而客户端数量不受限制。 我们假设上传到云的数据存储在键值数据库中。 我们将键值数据库建模为一组元组 *D* = *{* （ *k，v* ） *}* ，其中 *k* 是键， *v* 是与 *k* 相关的值 。 DO通过发送将数据上传到云端到预定义大小 *B* 的云块 = *{* （ *k，v* ） *}* 。通过向 云 发送密钥 *k* ，客户端检索元组（ *k，v* ） *∈* *d。*

  我们的系统模型是写入和读取密集型的。 我们假设DO只将新值附加到数据库中; 现有元组未更新。 虽然云中的隐私是一个备受关注的话题（例如参见[22-24]），但我们的模型侧重于数据完整性。 如上所述，我们对高度适用的解决方案感兴趣，因此我们假设不能在云端执行服务器更改。

*B.完整性和攻击模型*

我们假设DO和客户端都不信任CP，并且它可能以任何可能的方式恶意破坏数据完整性。 例如，CP可能会修改某些元组，添加或删除元组，或将部分（或空）结果返回给客户端的查询。

我们专注于以下两个方面的数据完整性保护：

•      *正确性：* 客户收到的数据最初是 由DO上传到云端，未在云端恶意或错误地修改。

•      *完整性：* CP返回所有客户端 与查询匹配的数据。 换句话说，结果中没有数据被省略。

新鲜度是数据完整性的另一个重要方面，这意味着客户端可以获得上载到云端的最新数据版本。 但是，由于在我们的系统模型中没有更新，因此新鲜度不是问题。

三、我们的方法

在我们的模型中，客户端通过“按键获取值”查询来查询云。CP返回的结果可能是空的，也可能不是空的。 如果它不是空的，客户端只需要检查其正确性，并且可以通过数据验证轻松实现。 我们工作的重点是确保空结果应该是空的。 换句话说，我们希望确保客户查询的结果是完整的 。

 下面我们将介绍我们用于正确性和完整性验证的保护技术。

*A.预赛*

*1）*  *散列函数：* 我们使用抗冲突散列函数，它具有一个属性，在计算上很难找到散列到同一输出的两个输入。 SHA-256和SHA-512 [4]就是这种功能的例子。 值 *x的* 哈希操作 由 *H* （ *x* ）表示 。

*2）*  *秘密密钥：* 我们假设DO和客户端 共享两个密钥 *{* *K* *e* *，K* *m* *}* ; *K* *e* 用于数据加密， *K* *m* 用于数据真实性。

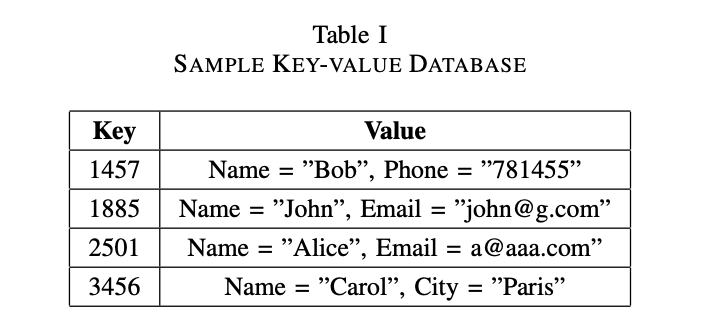
*3）*  *数据验证：* 验证我们的数据真实性 使用消息验证码（MAC）。 DO根据MAC方案（例如HMAC [5]）签署其数据，并将MAC值与签名数据一起存储在云中。 然后，基于MAC值和接收的数据，客户端可以验证数据的真实性。 签名和验证操作由 *Sign* （ *数据，密钥* ） 和V 序列化（ *数据，M AC，密钥* ）表示 。

*4）*  *数据加密：* 存储的敏感数据在云由DO加密，然后由客户端使用对称加密解密（例如AES [6]）。 我们将加密和解密操作表示为 *Enc* （ *明文，密钥* ） 和 *Dec* （ *密文，密钥* ） 。

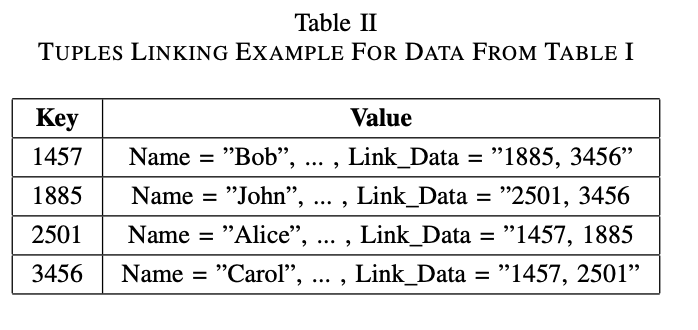
*B.完整性验证方案*

我们的目标是验证客户端查询的结果中是否没有遗漏任何数据。 在我们的方法中，我们使用两种技术——*元组链接* 和 下面描述的*众包验证* 。

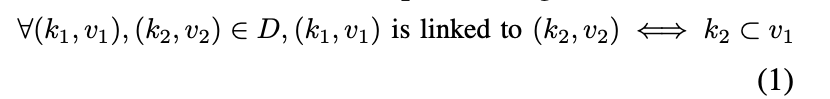
1. *元组链接：* *元组联* 背后的原理是，每一个元组知道其他的一些元组的存在。 例如，考虑以下键值数据库（表I），其中key是用户id，value是用户对象。



如果我们应用链接到表I中的样本数据的元组，结果可能如表II所示。



元组链接的正式定义如下：



在将数据上传到云时，DO负责元组之间的链接。之后，客户端可以依赖此链接数据来验证结果的完整性。例如，考虑对表II中的数据进行查询“获取具有ID 1457,1885的用户”以及仅包含具有id 1457的元组的结果。通过检查元组1457的链接数据，客户端知道元组1885应该是其中的一部分结果，从而检测到攻击

元组链接的实现基于这样的假设：DO通过批量加载将数据上传到云，因此，同一个批量内的元组可以彼此链接形成批量链接。但是，在同一批量的元组之间进行链接是不够的，因为如果CP删除了所有批量，则无法检测到它。我们建议DO在本地保留上一个批量的副本，当新批量到达时，将其元组链接到当前批量和前一个批量。链接元组的数量由参数p定义。批量链接可以正式定义如下：

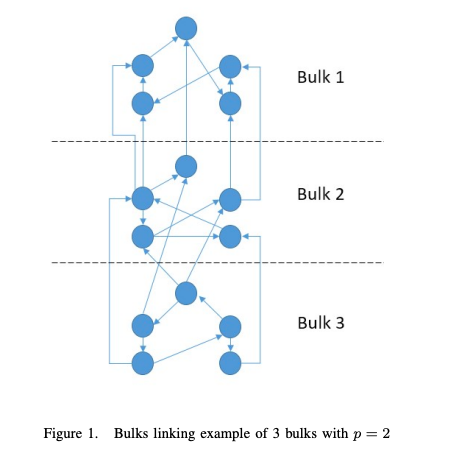


中有个元组，中有个元组，它们被链接到x。

图1显示了3个批量与5个密钥相关的批量连接图示：

•图形节点表示数据库元组。

•从节点A到节点B的边缘描绘了A链接到B（A知道B的存在）。



*2）众包验证：* 在众包（CS） 系统中，[7]用户通过合作实现一个有利于整个社区的目标。 协作可以是显式的（例如维基百科和Linux项目），也可以是隐式的，例如，在ESP游戏[8]中，用户将图像标记为玩游戏的副作用。在我们的方法中，我们构建了CS系统，用户通过隐式地协作以实现多目标数据库完整性保证。与之前关注查询完整性的工作（客户端仅验证自己的查询）不同，我们关注的是数据库完整性（客户端执行随机验证查询以验证CP的行为是否令人满意）。 这一战略背后的理由是：基于观察，数据库完整性比查询完整性更重要; 客户不愿意使用CP，因为CP会向某些查询返回可证明错误的结果（即使这些查询是假的并且是由其他用户发出的）。 我们在CS系统中依赖于这种观察，其中每个客户端贡献一点他的计算能力，以验证CP是否可信。 它的工作原理如下：

1）     客户端向CP发送查询。

2）     CP将查询结果连同链接数据一起发送回客户端。

3）     客户端根据收到的链接数据构建验证查询，并将它们发送给CP。

4）     CP将验证查询的结果发送回客户端。

5）     客户端验证验证查询的结果是否与链接数据匹配。

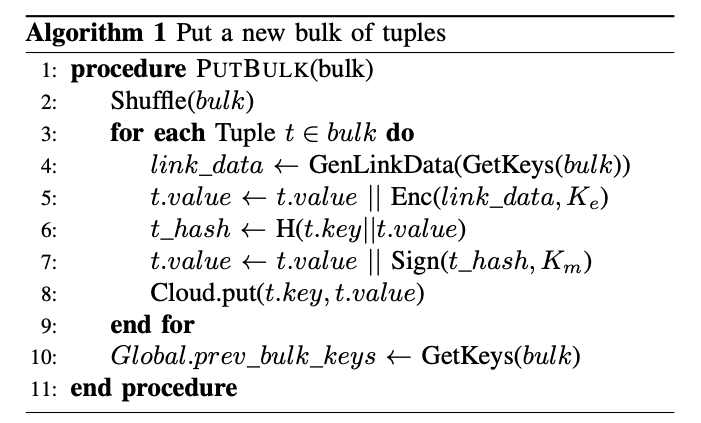
构建验证查询（步骤3），使得CP不能区分它们和常规客户端查询（步骤1）。 由于这一点，CP的客户端查询的恶意行为将不可避免地导致验证查询的恶意行为，从而被检测到。 例如，考虑对表II中的数据进行查询“获取id为1885的用户”。此客户端查询的验证查询将是“获取ID为2501的用户”和“获取ID为3456的用户”。 如果至少其中一个查询的结果为空，则检测到攻击。

注意，客户端查询（步骤1）和验证查询（步骤3）之间没有依赖性，因此步骤3-5可以异步执行（即不会损害读取延迟）。

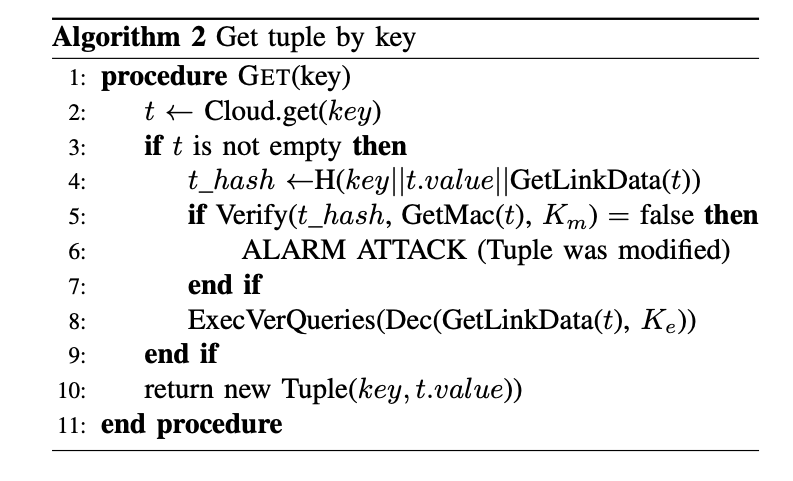
*C.伪代码*

下面给出了在我们的模型中提供正确性和完整性验证的算法的伪代码。

算法1将大量元组放入云数据库。 它遍历批量中的所有元组（第3-9行）并计算每个元组的链接数据。 然后链接数据被加密并存储为元组值的一部分。 为了能够将下一个批量链接到当前，我们将当前批量密钥存储在全局变量中（第10行）。



算法2按键返回元组。首先，在第4-6行验证元组的正确性。完整性验证基于验证查询，验证查询是基于解密的链接数据构建的（第8行）。



四、安全分析

*A.正确性*

我们的正确性验证方案直接基于众所周知的原语（MAC方案和抗碰撞哈希函数），只要它们是安全的。

*B.完整性*

我们的完整性验证方法基于两种技术：元组链接和众包验证，如上所述。

假设删除的元组和查询范围的分布都是统一的，那么在Q查询之后，客户端检测到从数据库中删除了d个元组（或从结果中省略了d个元组）和个元组的概率是：

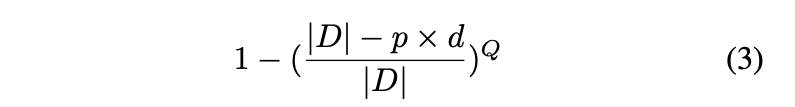
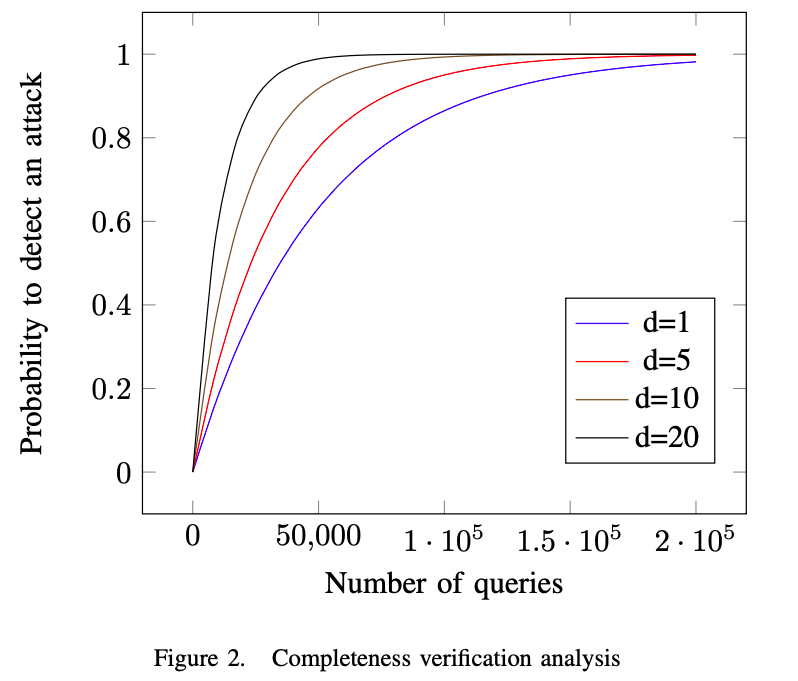


图2显示了检测攻击的概率，作为客户端使用 | 执行的查询数量的函数， p = 4 和 d ∈ {1,5,10,20} 。可以看出，即使 P很小，在相对少量的查询（生产系统每秒接收数万个查询[25]）和删除元组之后，CP逃脱被捕的可能性非常低。



五.实施和实验结果

对于实验评估，我们已经实现了我们的解决方案的原型。 作为云键值存储，我们使用Redis [9]。 为了评估我们的解决方案，我们使用Yahoo! 云服务基准（YCSB）框架[10]。 为了使用YCSB对特定数据库进行基准测试，唯一需要做的就是实现数据库接口层。 这将允许框架客户端执行诸如“读取元组”或“插入元组”之类的操作，而不必理解数据库的特定API。

我们以下列方式使用YCSB：

• YCSB框架已经有Redis客户端实现，名为RedisClient。

• 我们基于第III节中的算法实现了我们的Redis客户端版本（IRe-disClient）。 我们的实施可在线获取[11]。

• 我们配置了一个工作负载，其中对数据库执行预定义数量的可互换读取和插入操作。

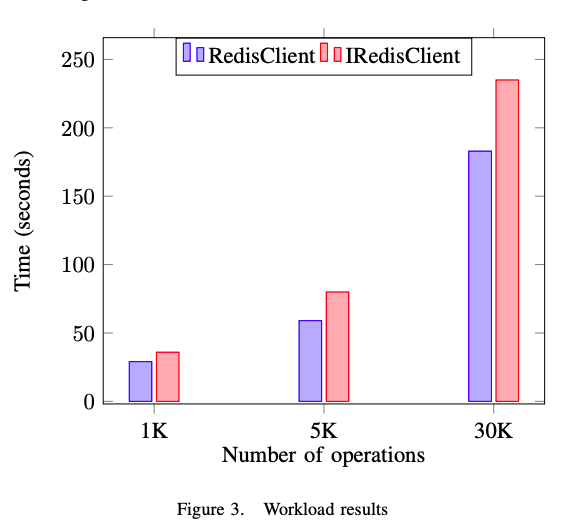
•      我们在两个客户端上执行了这个工作负载，并比较了它们的执行时间。 结果如下。

1. *建立*

我们使用1KB大小的元组和随机值。 我们将批量大小定义为100，将参数 *p* （链接元组的数量）定义为4。工作负载在最初包含30000元组的数据库上执行。 我们为每个客户执行了三次工作量，分别为5000，10000和30000次操作。下面的结果代表了这三次执行的平均值。

*B.业绩分析*

IRedisClient插入操作的成本主要由两个加密和一个散列操作（MAC计算和链接数据的加密）决定。 IRedisClient读取操作的成本类似于插入成本（MAC计算和链接数据解密），另外还有 *p* 验证查询的成本 。 实验结果（图3）表明，这种开销平均增加了29％的执行时间。



六、相关工作

现有解决方案可以主要分为三种类型。 第一种类型基于Merkle哈希树（MHT），第二种类型基于数字签名（DS），第三种类型使用概率方法。

A.基于MHT的方法

MHT [12]是二叉树，其中每个叶子是数据块的散列，并且每个内部节点是其两个子节点的连接的散列。 Devanbu等人在[13]中介绍了一种使用MHT作为认证数据结构（ADS）的方法，以在DBaaS模型中提供数据完整性保证。一般的想法是为每个数据库表构建MHT，使得MHT的叶子是由搜索关键字排序的表记录的哈希。为了减少客户端和服务器端的I / O操作成本，可以使用更高扇出（MB-Trees）的树[14]，而不是使用二叉树。 在[15]和[16]中分别给出了为连接和聚合查询提供有效完整性保证的不同基于MHT的技术。基于MHT的方法不适用于我们的系统模型，因为它是为范围查询而设计的，不支持对任意键进行有效查询。 它还需要进行大量的服务器端更改，这在我们的系统模型中是不允许的。

B.基于DS的方法

在RDBMS中提供数据完整性的自然而直观的方法是使用数字签名方案（例如RSA [17]），方法如下：

• 包含由DO私钥签名的连续记录值的散列将被添加到每个表中。

• 客户端使用DO的公钥验证记录的完整性。

为了降低客户端和服务器之间的通信成本以及客户端的计算成本，可以使用签名ag-gregation技术[18]将多个记录签名组合成一个。 在我们的方法中，我们使用类似的技术来确保数据正确性（唯一的区别是我们使用MAC代替DS而不使用聚合）。 为了保证完整性，DO不签署单独的记录，而是签署连续的记录对[19]。为此，DO必须在本地拥有数据库副本，或者能够从云中获取连续记录。这两种选择在我们的模型中是不可能的。

C.概率方法

概率方法仅提供概率性完整性保证，但不需要DBMS修改，并且具有比基于MHT和DS方法更好的性能。 在此方法中，许多其他记录与原始记录一起上传到云中。传的附加记录越多，检测数据完整性攻击的可能性就越高。所有数据都在客户端加密，因此CP无法区分原始记录和附加记录。这些附加记录可能完全是伪造的，如[20]中提出的，或者是使用不同（次要的）密钥加密的原始记录，就像方案[21]中提出的那样。伪记录方法仅适用于范围查询，因此与我们的模型无关，而双加密方案可以应用于我们的模型，但要求整个数据库必须加密。在我们的方案中，每个元组中只有少量密钥（即链接数据）被加密。 与我们的方法相比，由于记录重复，双加密方案还需要在云端更多的额外存储。

七、结论

在本文中，我们提出了我们用于云密钥值存储中的数据完整性保证的新方法。 我们的方法依赖于众包范式————用户协作实现共同目标————数据库完整性保证。 我们的方法优于现有方法的主要优点是其高适用性————它可以应用于现有系统而无需修改服务器端。我们实施了协议的概念验证原型并进行了实验评估。结果表明我们的方案施加了合理的开销。为了将来的工作，我们计划在其他类型的云数据库（例如列存储）上扩展我们的方案。

参考

1. Leavitt, Neal. ”Will NoSQL databases live up to their promise?.” Computer 43.2 (2010): 12-14.
2. Cattell, R. (2011). Scalable SQL and NoSQL data stores. ACM SIGMOD Record, 39(4), 12-27.

[3] Hacigms, H., Iyer, B., Mehrotra, S. (2002). Providing database as a service. In Data Engineering, 2002. Proceed-ings. 18th International Conference on (pp. 29-38). IEEE.

1. Standard, S. H. National Institute of Standards and Technol-ogy (NIST), FIPS Publication 180-2, Aug 2002.
2. Krawczyk, H., Canetti, R., Bellare, M. (1997). HMAC: Keyed-hashing for message authentication.
3. Pub, N. F. (2001). 197: Advanced encryption standard (AES). Federal Information Processing Standards Publication, 197, 441-0311.
4. Doan, A., Ramakrishnan, R., Halevy, A. Y. (2011). Crowd-sourcing systems on the world-wide web. Communications of the ACM, 54(4), 86-96.
5. Von Ahn, L., Dabbish, L. (2004, April). Labeling images with a computer game. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 319-326). ACM.
6. Redis. http://redis.io/
7. Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R. (2010, June). Benchmarking cloud serving systems with YCSB. In Proceedings of the 1st ACM symposium on Cloud computing (pp. 143-154). ACM.
8. IRedisClient. https://github.com/grishaw/ycsb-iredis-binding
9. Merkle, R. C. (1989, August). A certified digital signature. In Advances in CryptologyCRYPTO89 Proceedings (pp. 218-238). Springer New York.

[13] Devanbu, P., Gertz, M., Martel, C., Stubblebine, S. G. (2003). Authentic data publication over the internet. Journal of Computer Security, 11(3), 291-314.

1. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L. (2006, June). Dynamic authenticated index structures for outsourced databases. In Proceedings of the 2006 ACM SIGMOD inter-national conference on Management of data (pp. 121-132). ACM.
2. Yang, Yin, et al. ”Authenticated join processing in outsourced databases.” Proceedings of the 2009 ACM SIGMOD Interna-tional Conference on Management of data. ACM, 2009.
3. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L. (2010). Authenticated index structures for aggregation queries. ACM Transactions on Information and System Security (TISSEC), 13(4), 32.
4. Rivest, R. L., Shamir, A., Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120-126.
5. Mykletun, E., Narasimha, M., Tsudik, G. (2006). Authentica-tion and integrity in outsourced databases. ACM Transactions on Storage (TOS), 2(2), 107-138.
6. Narasimha, M., Tsudik, G. (2005, October). DSAC: in-tegrity for outsourced databases with signature aggregation and chaining. In Proceedings of the 14th ACM international conference on Information and knowledge management (pp. 235-236). ACM.
7. Xie, M., Wang, H., Yin, J., Meng, X. (2007, September). Integrity auditing of outsourced data. In Proceedings of the 33rd international conference on Very large data bases (pp. 782-793). VLDB Endowment.
8. Wang, H., Yin, J., Perng, C. S., Yu, P. S. (2008, October). Dual encryption for query integrity assurance. In Proceedings of the 17th ACM conference on Information and knowledge management (pp. 863-872). ACM.
9. Hacigm, H., Iyer, B., Li, C., Mehrotra, S. (2002, June). Executing SQL over encrypted data in the database-service-provider model. In Proceedings of the 2002 ACM SIGMOD international conference on Management of data (pp. 216-227). ACM.
10. Damiani, E., Vimercati, S. D. C. D., Jajodia, S., Paraboschi, S., Samarati, P. (2003, October). Balancing confidentiality and efficiency in untrusted relational DBMSs. In Proceedings of the 10th ACM conference on Computer and communica-tions security (pp. 93-102). ACM.
11. Ciriani, V., Vimercati, S. D. C. D., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P. (2010). Combining fragmentation and encryption to protect privacy in data storage. ACM Transactions on Information and System Security (TISSEC), 13(3), 22.
12. Atikoglu, B., Xu, Y., Frachtenberg, E., Jiang, S., Paleczny, M. (2012, June). Workload analysis of a large-scale key-value store. In ACM SIGMETRICS Performance Evaluation Review (Vol. 40, No. 1, pp. 53-64). ACM.