定理 4. 如果 Hash 函数是抗碰撞的,则不存在一个 PPT 敌手能够以不可忽略的概率伪造出通过 TPA 校验的定位标签证明.

证明:被挑战服务器在每次挑战时都会收到 TPA 发来的重构 MHT 的参数.由于每次挑战用于定位的 MHT 根标签所使用的参数不同,使得服务器无法提前预知、计算并存储叶子节点,所以收到挑战的服务器都需要根据挑战中 MHT 参数集$\{\alpha_j\}$重新计算相应的 MHT 根值作为证明.在 MHT 中,叶子节点是参数与文件块连接后做 Hash 运算得到的,而根值是由所有叶子节点经过多层计算 Hash 值得到的,所以,若 Hash 函数是抗碰撞的,则要在不知道参数与文件块值的情况下伪造出可通过校验的根值的概率是可以忽略的.证毕. □

此外,要说明的是,为了实现错误定位,且使得用户端不额外存储多余的信息,有些信息,如用户数据块所存储的服务器和所有用户数据的定位索引表,TPA 都是知道的.而定位索引表中的元素是每个用户存储在不同服务器上的数据所构成的 MHT 树根,若 Hash 函数是抗原象攻击的,则 TPA 无法得知用户原始数据块值.

5.3 性能分析

在下面的分析中,n 表示所有云用户的文件块总数,c 表示 TPA 选中的被挑战块的数量,n_1 表示 c 个被挑战块所属用户的数量,n_2 表示 c 个被挑战块所在的服务器的数量,η表示所有云服务器的数量,γ表示所有云用户的数量,s 表示每个数据块的分区数,λ表示每个用户对其存放在一个服务器上的数据所构建的 MHT 数量.c_j 表示被挑战服务器 CS_j 上被挑战的块数,有 $1\leqslant c_j\leqslant c_{\max}=c-n_2+1$.假设云用户 DO_i 总共将 m_i 个文件块存储到多个服务器中,服务器 CS_j 上共存储了 m'_j 个文件块.

方案的通信轮数为 1 轮,在此一轮通信中既实现了批处理校验,又能实现错误数据的定位功能.

1. 关于通信复杂度

在初始阶段,云用户除了将文件块$\{M_{ijk}\}$和相应的数据标签$\{\sigma_{ijk}\}$发送给服务器以外,为了实现对其错误数据进行定位,还需要将定位索引表发送给 TPA,每个用户的定位索引表中包含$\lambda(\eta+1)$个 Z_q 中的元素.

在挑战阶段,TPA 发送总挑战 $chal=(I,K,\alpha)$,其中,I 中包括 $3c$ 个整数,K 中包含 c 个 Z_q 中的元素,α中也包含 c 个 Z_q 中的元素.综上,挑战阶段的通信复杂度为 $O(c)$.

在响应阶段,每个被挑战服务器返回的证明 $P_j=(S'_j,T'_j,\{F'_{ijl}\mid i\in O_j,l=1,...,s\},\{(i,j,R_{ijr})\mid i\in O_j\})$, 其中, S'_j, $T'\in G_1,\{F'_{ijl}\mid i\in O_j,l=1,...,s\}$中至多包含 n_1s 个由数据块分区计算得到的值,$\{(i,j,R_{ijr})|i\in O_j\}$中至多包含 $2n_1$ 个整数和 n_1 个 Z_q 中的元素.因此,每个服务器返回的证明中至多包含 $2+n_1s+2n_1+n_1$ 个元素,所有被挑战服务器共返回至多$(2+n_1s+3n_1)n_2$ 个元素.综上,响应阶段的通信复杂度为 $O(n_1n_2s)$.

2. 关于存储复杂度

服务器存储着用户的数据块和相应的数据标签,与其他方案相似.TPA 需要存储所有用户发来的数据块定位标签,共有γ张定位索引表,包含$\gamma\lambda(\eta+1)$个 Z_q 中的元素,存储复杂度为 $O(\gamma\lambda\eta)$.

3. 关于计算复杂度

云用户:除了为每个数据块计算相应的数据标签以外,为了实现错误定位,额外地,云用户需要对其存储在不同服务器上的数据块分别构建λ棵 MHT.含有 N_{ij} 个叶子节点的 MHT 最多需要做 $2^{\lceil\log_2 N_{ij}\rceil+1}-1$ 次 Hash 运算.拥有 m_i 个数据块的云用户 DO_i 最多计算 $\lambda\cdot(2^{\lceil\log_2 m_i\rceil+1}-1)$ 次 Hash,因此,DO_i 计算定位标签的时间复杂度为 $O(\lambda m_i)$,这项工作是一次性的.

被挑战服务器:计算的证明包含两部分:(1) 用于批处理校验的部分,需要做 c_j 次伪随机函数运算、$2c_j$ 次指数运算、$2(c_j-1)$次群中乘法运算、$s\cdot c_j$ 次普通乘法运算,最多 $s\cdot(c_j-1)$次普通加法运算;(2) 用于定位错误的部分,最多需要进行 $2^{\lceil\log_2 m'_j\rceil+1}-1$ 次 Hash 运算.

TPA:批处理校验包括 c 次伪随机函数运算、n_1 次指数运算和 3 次双线性对运算,最多 $n_1(s-1)\cdot(n_2-1)+c$ 次普通加法运算、最多 $s\cdot\min(n_1n_2,c)$次普通乘法运算、$2(n_2-1)+(n_1-1)$次群中乘法运算.若批处理校验不通过,则再对服务器返回的树根一一进行对比校验,判断某一用户存放在某一服务器上的数据是否遭到损毁,只需一次比较操作.

下面将我们的方案与其他多用户多服务器环境下支持批处理校验的方案进行比较.从效率和是否支持错误定位方面,我们的方案与 He 等人[18]、Shin 等人[19]和 Zhou 等人[17]的方案进行对比的结果见表 3.

Table 3 Comparison of efficiency and location function

表 3 效率及定位功能比较

方案	我们的方案		He 等人的方案[18]	Shin 等人的方案[19]	Zhou 等人的方案[17]
审计阶段通信复杂度	与数据标签相关的 $2n_2+4c+n_1n_2s$	与定位标签相关的 $3n_1n_2+c$	$2c+n_1n_2+2$	$4n_2+2c+n_1n_2$	$n_1n_2s+2n_2+4c$
用户端计算复杂度	$3m_iC_{exp}+(m_i)C_{mG}+sm_iC_{mZ}+(s+1)m_iC_{aZ}+2C_H$	$2\lambda m_iC_H$	$2m_iC_{exp}+m_iC_{mG}$	$2m_iC_{exp}+m_iC_{mG}+m_iC_H+C_{sig}$	$3m_iC_{exp}+(m_i)C_{mG}+sm_iC_{mZ}+(s+1)m_iC_{aZ}+2C_H$
服务器端计算复杂度	$c_jC_{per}+2c_jC_{exp}+(2c_j-2)C_{mG}+sc_jC_{mZ}+s(c_j-1)C_{aZ}$	$2m'_jC_H$	$n_1c'C_{exp}+n_1c'C_{mG}+n_1c'C_{mZ}+n_1c'C_{aZ}$	$(n_1+c_j)C_{exp}+3n_1C_{par}+(2n_1+c_j)C_{mG}+c_jC_{mZ}+c_jC_{aZ}+2n_1C_H$	$c_jC_{pre}+2c_jC_{exp}+(2c_j-2)C_{mG}+sc_jC_{mZ}+s(c_j-1)C_{aZ}$
TPA 端批处理校验计算复杂度	$cC_{per}+n_1C_{exp}+3C_{par}+(n_1+2n_2-3)C_{mG}+s\cdot\min(n_1n_2,c)C_{mZ}+[n_1\cdot(s-1)(n_2-1)+c]C_{aZ}$		$n_1C_{exp}+3C_{par}+2n_1C_{mG}$	$cC_{exp}+n_1n_2C_{par}+(4n_1+c)C_{mG}+C_{dG}+cC_H$	$cC_{per}+n_1C_{exp}+3C_{par}+(n_1+2n_2-3)C_{mG}+s\cdot\min(n_1n_2,c)C_{mZ}+[n_1\cdot(s-1)(n_2-1)+c]C_{aZ}$
TPA 端存储复杂度	$\gamma\lambda(\eta+1)$		无	无	无
是否支持定位错误数据所在服务器	是		否	是	否
是否支持定位错误数据所属拥有者	是		是	否	否
TPA 定位特定错误数据的时间或计算复杂度	T_{cpr}		$(c'+1)C_{exp}+(c'-1)C_{mG}+3C_{par}+T_{cpr}$	$n_2T_{cpr}+\left(\frac{3}{2}n_2^2+\frac{5}{2}n_2-1\right)C_{mG}$	–

其中,$C_{\exp}$ 表示群 G_1 上单个指数运算的开销,C_H 表示单个 Hash 函数运算的开销,C_{par} 表示一个双线性对运算的开销,C_{per} 表示一个伪随机函数运算的开销,C_{mG} 表示群上单个乘法运算的开销,C_{dG} 表示群上单个除法运算的开销,C_{mZ} 表示单个普通乘法运算的开销,C_{aZ} 表示单个普通加法运算的开销,T_{cpr} 表示一次比较的时间开销.因为文献[18]中每个用户的数据都会被挑战且被挑战的块索引相同,令 c'表示一个被挑战用户被挑战的块数,即 $c=c'\cdot n_1$.

4 个方案中,审计阶段都仅需 1 轮通信,我们的方案在 1 轮通信中不仅能够实现批处理校验,还能在校验不通过情况下定位错误数据所属用户和所属服务器;而文献[18]只支持定位错误数据的拥有者;文献[19]只支持定位错误数据所在服务器,并且仅适用于只有 1 个服务器的数据被损毁的情景.我们的方案是基于查找的方式定位错误数据,判断特定用户存储在特定服务器上的数据是否出错仅需一次比较操作;而文献[18,19]都是利用计算的方式来实现错误数据定位,文献[18]判断特定用户的数据是否损毁需要 $O(c')$次指数运算,文献[19]定位唯一的数据被损毁服务器需要 $O(n_2^2)$ 次乘法运算.

在我们的方案中,为了计算定位标签,用户需要为其数据块构建 MHT,服务器在计算证明时需要重构 MHT,所以相对于其他方案,在用户端和服务器端分别增加了 $2\lambda m_i$ 和 $2m'_j$ 次 Hash 运算.由于 Hash 运算的速度很快,所以增加的计算对总体的计算开销影响并不显著.为了定位错误,相对于文献[17–19],TPA 需要额外存储 γ 张定位索引表,总共 $\gamma\lambda(\eta+1)$ 个 Z_q 中的元素,而 TPA 进行批处理校验的计算量相对于文献[17]来说并没有增加.

5.4 仿真实验

我们首先对云用户、服务器和 TPA 各自计算的时间开销进行了仿真实验,然后对两种定位错误方式的定位效率进行了对比.PC 硬件配置为 Intel Core2Duo 处理器、4G 内存,操作系统为 Ubuntu 16.04 LTS 32 位,利用 PBC 库[20]、GMP 库[21]和 Miracl 库[22],使用 gcc 编译执行.实验中,使用 PBC 库中 a.param 参数设置双线性对,构

造 MHT 时采用 SHA-256.

1. 用户计算数据标签 TagGen 和定位标签 PosTagGen 的计算开销

用户将文件分块后,对每个数据块计算相应的数据标签,然后对存储在不同服务器上的数据块计算定位标签.在实验中,我们设置每个数据块 4KB,每个分区 20B,通过设置不同的文件大小来观察 TagGen 和 PosTagGen 的计算开销.令用户文件大小为 5MB~40MB,相应的数据块数为 1 250~10 000,设置步长 5MB.TagGen 的计算开销与λ=64 和λ=128 时 PosTagGen 的计算开销实验结果如图 4 所示.

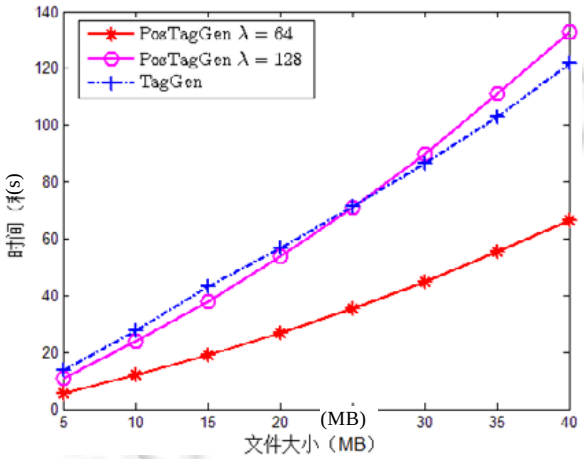


Fig.4 Computational cost of TagGen & PosTagGen for increased size of files

图 4 文件大小增加时 TagGen 和 PosTagGen 的计算开销

从实验结果可以看出,TagGen 的时间随着文件大小的增长(数据块数增长)呈线性增长,与性能分析结果一致.特别地,当文件大小为 5MB(40MB)时,TagGen 的时间为 13.6s(121.7s).PosTagGen 的计算开销与文件大小(数据块数量)和λ值是相关的:当λ值固定时,PosTagGen 的计算开销随着数据块数量增加而增大,基本呈线性增长,实验结果与性能分析相符合.进一步观察,当λ=64,文件大小为 5M(40M)时,PosTagGen 的时间为 5.5s(66.5s).当λ=128,文件大小为 5MB(40MB)时,PosTagGen 的时间为 11.0s(133.0s).TagGen 与 PosTagGen 的计算开销较大,耗费的时间显著高于其他操作耗费的时间,但是对于一个文件来说都仅需计算 1 次.表 4 为当λ=64 和λ=128 时,TPA 进行校验的不同时间间隔所能支持的错误定位有效期.

Table 4 Error locating period of validity for different verification time interval

表 4 不同校验时间间隔下,支持错误定位的有效期

校验时间间隔	支持定位的有效期	
	λ=64	λ=128
7 天	15 个月	30 个月
14 天	30 个月	60 个月
1 个月	64 个月	128 个月
3 个月	192 个月	384 个月

2. 服务器计算证明 Prove-DataTag、Prove-PositionTag 和 TPA 批量校验数据标签证明 Verify 的计算开销

在挑战响应阶段,收到挑战的服务器需要计算数据标签的证明 Prove-DataTag 和定位标签的证明 Prove-PositionTag,而 TPA 需对被挑战服务器返回的数据标签证明进行批量审计.

服务器 Prove-DataTag 的计算开销与 TPA 批量审计 Verify 的计算开销均与 TPA 选取的挑战块数量有关,所以我们在同样的条件下对这两个计算进行了仿真.实验中,我们设置了 10 个用户和 10 个服务器,每个用户拥有 10 000 个数据块,每个数据块 4KB,每个分区 20B,每个用户将其 10 000 个数据块均匀地存储到 10 个服务器

上,这样,每个服务器上存储着 10 个用户的 10 000 个数据块.若云服务器的数据块损毁率为 1%,则 TPA 挑战其上的 300 个(460 个)数据块,就能以 95%(99%)的概率检测出该服务器的损毁数据行为[23].因此,在 TPA 随机均匀选取 10 个服务器上的挑战块的情况下,我们对总挑战块数为 3 000~4 600(相应的每个服务器上被挑战块数为 300~460),以步长为 200,进行了实验,结果如图 5 所示.

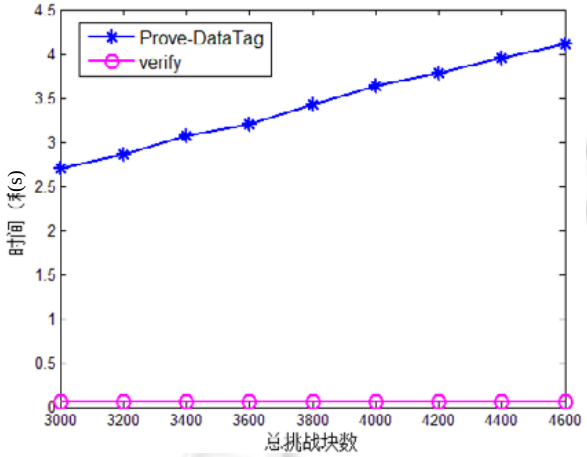


Fig.5 Computational cost of Prove-DataTag by single server & Verify by TPA for increased number of total challenged blocks

图 5 总挑战块数增加时,单个服务器 Prove-DataTag 和 TPA Verify 的计算开销

从实验结果可以看出,服务器 Prove-DataTag 的计算开销随着其上被挑战块数量的增加而增长.这主要是因为当其上被挑战块数增加时,服务器需要为增加的挑战块的数据标签做群上的指数运算.随着总挑战块数的增加,TPA 批量校验数据标签证明 Verify 的计算开销增长并不明显.这是因为增加的操作只是一些伪随机函数和普通加法运算,实验结果与性能分析结果一致.进一步观察,当总挑战块数为 3 000(4 600)个时,服务器 Prove-DataTag 的时间为 2.7s(4.1s),TPA Verify 的时间仅为 53ms(54ms).

被挑战服务器计算定位标签证明 Prove-PositionTag 的计算开销与该服务器上被挑战用户的所有数据块数有关.我们对其中最坏的情况进行了模拟,即数据存放于该服务器上的所有用户均有数据块被 TPA 挑战,所以被挑战服务器需对其上存储的所有数据块进行重构 MHT 的操作.我们令被挑战服务器存储的数据块数为 5 000~12 000,步长为 1 000,对单个被挑战服务器计算定位标签证明 Prove-PositionTag(即重构 MHT)的计算开销进行测试,结果如图 6 所示.

从实验结果可以看出,服务器重构 MHT 的计算开销随着该服务器上存储的数据块数增加而增长,且增长趋势基本呈线性,与性能分析结果一致.由于 Prove-PositionTag 的计算是重构 MHT 树,所做的都是 Hash 操作,因此即使是在最坏情况下,重构 MHT 的时间也是较快的.当服务器上存有 5 000(12 000)个数据块,且所有其上用户都有数据块被挑战时,服务器 Prove-PositionTag 的时间为 0.42s(1.34s).

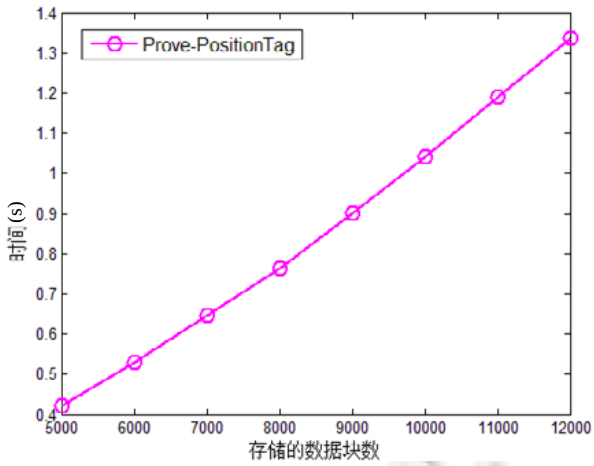


Fig.6 Computational cost of Prove-PositionTag by single server for its increased number of stored blocks

图 6 存储的数据块数增加时,单个服务器 Prove-PositionTag 的计算开销

3. TPA 定位错误时间比较

在本节中,我们对两种定位错误方式的定位效率进行对比:逐一校验方式和利用定位标签辅助定位的方式.为了使对比结果更合理,我们同样是在 Zhou 等人方案[17]的基础上实现逐一校验定位错误,并设置相同的环境参数.但需要说明的是,逐一校验方式使得 Zhou 等人的方案只能定位错误数据所在服务器.

在实验中,我们设置 10 个用户和 100 个服务器,每个用户拥有 100 000 个数据块,每个数据块 4KB,每个分区 20B,每个用户将其 100 000 个数据块均匀存储到 100 个服务器上,这样,每个服务器上就存储着 10 个用户的 10 000 个数据块.在 TPA 随机均匀选取每个被挑战服务器上 10 个用户的 300 个挑战块的情况下,令被挑战服务器个数为 10~100,模拟两种定位错误方法的计算开销,实验结果如图 7 所示.

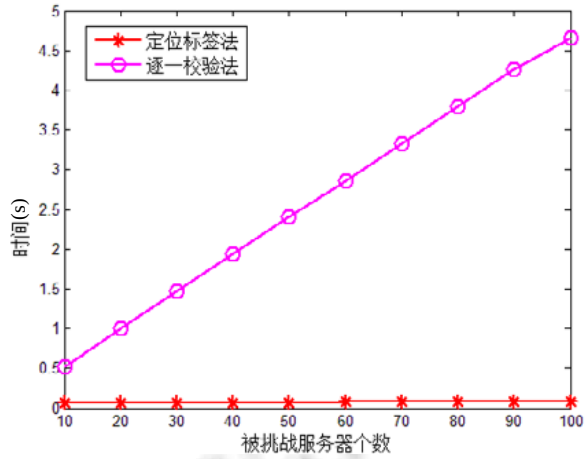


Fig.7 Time comparison of location errors by TPA

图 7 TPA 定位错误时间比较

从实验结果可以看出,随着被挑战服务器数量的增加,逐一校验方式定位错误服务器的时间耗费呈线性增长趋势,而利用定位标签定位错误数据所属用户和所在服务器的计算开销增长并不明显.这是因为逐一校验方式中,TPA 针对每个被挑战服务器返回的证明单独校验,每次校验都需要做多个指数运算和双线性对运算;而利用定位标签的方式只涉及比较操作.在被挑战服务器个数为 10(100)时,定位标签方式仅需 0.059s(0.087s),而逐

一校验方式则需 0.523s(4.652s).显然,随着被挑战服务器个数的增加,使用定位标签辅助错误定位的效率显著优于逐一校验的方式.

6 总 结

本文主要研究了批处理 PDP 方案在批量审计不通过情况下的错误数据定位问题.提出了利用定位标签辅助第三方审计员快速定位错误的方法,并在多用户多服务器环境下给出了一个具体实现,在批处理校验不通过的情况下,仅通过比较操作就能同时定位错误数据所属用户和所在服务器.我们对方案的正确性和安全性进行了证明,并对方案的性能进行了理论分析和仿真实验.性能分析结果表明,我们的方案在定位错误数据的能力和效率方面均高于其他具有单一定位功能的方案.实验结果也表明,利用定位标签辅助错误定位的效率显著优于逐一校验的方式,且实现错误定位功能的额外计算开销是可接受的.

在本文方案中,预先设定的 MHT 数量使得本方案不适合进行无限次的错误定位.为了缓解次数限制问题,有两种可行的解决办法.

(1) 不要求每次校验都返回树的根值.仅当批处理校验发生错误后,校验者给服务器再次发送挑战,要求服务器提供相应树的根值.

(2) 对服务器建立分级制度,校验者在挑战时可设立一个状态标识,若状态为“1”,则要求返回根值;若为“0”则不要求.对信誉好的服务器,可以适当减少其返回根值的次数.

References:

[1] Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D. Provable data possession at untrusted stores. In: Proc. of the 14th ACM Conf. on Computer and Communications Security. ACM Press, 2007. 598−609. [doi: 10.1145/1315245.1315318]

[2] Ateniese G, Pietro RD, Mancini LV, Tsudik G. Scalable and efficient provable data possession. In: Proc. of the 4th Int’l Conf. on Security and Privacy in Communication Networks. ACM Press, 2008. 1−10. [doi: 10.1145/1460877.1460889]

[3] Wang Q, Wang C, Li J, Ren K, Lou WJ. Enabling public verifiability and data dynamics for storage security in cloud computing. In: Backes M, Ning P, eds. Proc. of the Computer Security (ESORICS 2009). LNCS 5789, Berlin: Springer-Verlag, 2009. 355−370. [doi: 10.1007/978-3-642-04444-1_22]

[4] Wang C, Wang Q, Ren K, Lou WJ. Privacy-preserving public auditing for data storage security in cloud computing. In: Proc. of the 2010 IEEE INFOCOM. IEEE, 2010. 1−9. [doi: 10.1109/INFCOM.2010.5462173]

[5] Hao Z, Zhong S, Yu NH. A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. IEEE Trans. on Knowledge & Data Engineering, 2011,23(9):1432−1437. [doi: 10.1109/TKDE.2011.62]

[6] Yu Y, Au MH, Mu Y, Tang SH, Ren J, Susilo W, Dong LJ. Enhanced privacy of a remote data integrity-checking protocol for secure cloud storage. Int’l Journal of Information Security, 2015,14(4):307−318. [doi: 10.1007/s10207-014-0263-8]

[7] Yu Y, Zhang YF, Ni JB, Au MH, Chen LX, Liu HY. Remote data possession checking with enhanced security for cloud storage. Future Generation Computer Systems, 2015,52:77−85. [doi: 10.1016/j.future.2014.10.006]

[8] Yu Y, Ni JB, Au MH, Mu Y, Wang BY, Li H. Comments on a public auditing mechanism for shared cloud data service. IEEE Trans. on Services Computing, 2015,8(6):998−999. [doi: 10.1109/TSC.2014.2355201]

[9] Yu Y, Li YN, Ni JB, Yang GM, Mu Y, Susilo W. Comments on “public integrity auditing for dynamic data sharing with multiuser modification”. IEEE Trans. on Information Forensics & Security, 2016,11(3):658−659. [doi: 10.1109/TIFS.2015.2501728]

[10] Yu Y, Xue L, Au MH, Susilo W, Ni JB, Zhang YF, Vasilakos AV, Shen J. Cloud data integrity checking with an identity-based auditing mechanism from RSA. Future Generation Computer Systems, 2016,62:85−91. [doi: 10.1016/j.future.2016.02.003]

[11] Yu Y, Au MH, Ateniese G, Huang XY, Susilo W, Dai YS, Min GY. Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Trans. on Information Forensics & Security, 2017,12(4):767−778. [doi: 10.1109/TIFS.2016.2615853]

[12] Wang C, Chow SM, Wang Q, Ren K, Lou WJ. Privacy-preserving public auditing for secure cloud storage. IEEE Trans. on Computers, 2013,62(2):362−375. [doi: 10.1109/TC.2011.245]

[13] Ren ZW, Wang LN, Wu QH, Deng RY. Data dynamics enabled privacy-preserving public batch auditing in cloud storage. Chinese Journal of Electronics, 2014,23(2):297−301.

[14] Wang HQ. Identity-based distributed provable data possession in multicloud storage. IEEE Trans. on Services Computing, 2015, 8(2):328−340. [doi: 10.1109/TSC.2014.1]

[15] Mao J, Cui J, Zhang Y, Ma HJ, Zhang JH. Collaborative outsourced data integrity checking in multi-cloud environment. In: Yang Q, Yu W, Challal Y, eds. Proc. of the Wireless Algorithms, Systems, and Applications (WASA 2016). LNCS 9798, Berlin: Springer-Verlag, 2016. 511−523. [doi: 10.1007/978-3-319-42836-9_45]

[16] Liu X, Jiang YJ. Batch auditing for multi-client dynamic data in multi-cloud storage. Int'l Journal of Security & Its Applications, 2014,8(6):197−210. [doi: 10.14257/ijsia.2014.8.6.18]

[17] Zhou FC, Peng S, Xu J, Xu ZF. Identity-based batch provable data possession. In: Chen L, Han J, eds. Proc. of the Provable Security (ProvSec 2016). LNCS 10005, Berlin: Springer-Verlag, 2016. 112−129. [doi: 10.1007/978-3-319-47422-9_7]

[18] He K, Huang CH, Wang JH, Zhou H, Chen X, Lu YL, Zhang LZ, Wang B. An efficient public batch auditing protocol for data security in multi-cloud storage. In: Proc. of the 8th Chinagrid Conf. IEEE, 2013. 51−56. [doi: 10.1109/ChinaGrid.2013.13]

[19] Shin S, Kim S, Kwon T. Identification of corrupted cloud storage in batch auditing for multi-cloud environments. In: Khalil I, Neuhold E, Tjoa A, Xu L, You I, eds. Proc. of the Information and Communication Technology—EurAsia Conf. (ICT-EurAsia 2015). LNCS 9357, Berlin: Springer-Verlag, 2015. 221−225. [doi: 10.1007/978-3-319-24315-3_22]

[20] https://crypto.stanford.edu/pbc/

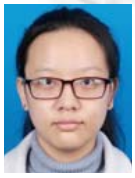
[21] https://gmplib.org/

[22] https://certivox.org/display/EXT/MIRACL

[23] Ateniese G, Burns R, Curtmola R, Herring J, Khan O, Kissner L, Peterson Z, Song D. Remote data checking using provable data possession. ACM Trans. on Information & System Security, 2011,14(1):No.12. [doi: 10.1145/1952982.1952994]

庞晓琼(1982－),女,山西太原人,博士,讲师,CCF 专业会员,主要研究领域为信息安全,密码学.

陈文俊(1980－),男,博士生,高级工程师,主要研究领域为密码学理论及其应用.



王田琪(1993－),女,硕士,主要研究领域为信息安全,密码学.

任孟琦(1994－),女,硕士生,CCF 学生会员,主要研究领域为信息安全,密码学.