On Securing Untrusted Clouds with Cryptography

ABSTRACT

In a recent interview, Whitfield Diffie argued that "the whole point of cloud computing is economy" and while it is possible in principle for "computation to be done on encrypted data, [...] *current techniques would more than undo the economy gained by the outsourcing and show little sign of becoming practical*". Here we explore whether this is truly the case and quantify just *how* expensive it is to secure computing in untrusted, potentially curious clouds.

1. INTRODUCTION

Commoditized outsourced computing has finally arrived, mainly due to the emergence of fast and cheap networking and efficient large scale computing. Amazon, Google, Microsoft and Sun are just a few of the providers starting to offer increasingly complex storage and computation outsourcing "cloud" services. CPU cycles have become consumer merchandise.

In [5] we explored the end-to-end cost of a CPU cycle in various environments and show that its cost lies between 0.45 picocents in efficient clouds and 27 picocents for small business deployment scenarios (1 picocent = \$1 $\times 10^{-14}$). In terms of pure CPU cycle costs, current clouds present seemingly cost-effective propositions for personal and small enterprise clients.

Nevertheless, cloud clients are concerned with the **privacy of their data and computation** – this is often the primary adoption obstacle, especially for medium and large corporations, who often fall under strict regulatory compliance requirements.

To address this, existing secure outsourcing research addressed several issues ranging from guaranteeing integrity, confidentiality and privacy of outsourced data to secure querying on outsourced encrypted database. Such assurances will likely require strong cryptography as part of elaborate intra- and client-cloud protocols. Yet, strong crypto is expensive. Thus, it is important to ask: how much cryptography can we afford in the cloud while maintaining the cost benefits of outsourcing?

Some believe the answer is simply *none*. In a recent interview [39] Whitfield Diffie argued that "**current techniques would more than undo the economy [of] outsourcing and show little sign of becoming practical**."

Here we set out to find out whether this holds and if so, by what margins. One way to look at this is in terms of CPU cycles. For each desired un-secured client CPU cycle, *how many additional cloud cycles can we spend on cryptography*, before its outsourcing becomes too expensive?

2. COST MODELS

In ongoing research [5] we explore the cost of computing (CPU cycles, networking, storage) in various environments as well as the boundary condition that defines when cloud computing becomes viable, i.e., when the CPU cycle cost savings are enough to offset the client-cloud distance. Here we summarize.

Computing Environments The cost of computing is a function of scale and environment of varying complexity: home (H), small (S), mid-size (M) and large size data centers (L).

Cost factors. A number of cost factors come into play across all of the above levels These can be divided into inter-dependent vectors, including: hardware (servers, networking gear), building (floor space leasing), energy (running hardware and cooling), service (administration, staffing, maintenance), and network service.

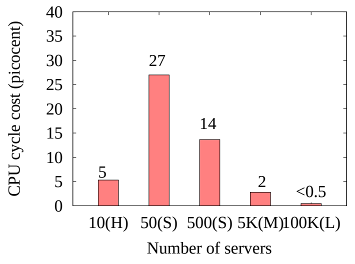

Figure 1: CPU cycle costs.

CPU Cycles The amortized cost of a CPU cycle in various environments were computed. This cost (Figure 1) ranges from 0.45 picocents/cycle for large and efficient enterprise/cloud settings, all the way up to (S), apparently the costliest environment, where a cycle costs up to 27 picocents (*1 US picocent* = 10^{-14} *USD*). These numbers were validated with the pricing points of main current cloud providers: Amazon [1], Google [18] and Microsoft . The prices lie surprisingly close to each other and to our estimates, ranging from 0.93 to 2.36 picocents/cycle. The difference in cost is due to the fact that these points include not only CPUs but also intra-cloud networking, and instance-specific disk storage.

Provider	**Picocents**
Amazon EC2	0.93 - 2.36
Google AppEngine	up to 2.31
Microsoft Azure	up to 1.96

Figure 2: Today's pricing.

Storage Cost Simply storing bits on disks has become truly cheap. Increased hardware reliability (with mean time between failures rated routinely above a million hours even for consumer markets) and economies of scale resulted in extreme drops in the costs of disks. In [5], we showed that in terms of amortized acquisition costs, the best price/hardware/MTBF ratio from our sample set is at 26.06 picocents/bit/year. The dominant factor is energy, 60-350 picocents/bit/year, at 60-90% of the total cost. The lowest total cost from our sample set is at about 100 picocents/bit/year.

Network Service Published network service cost numbers place network service costs for large data centers at around \$13/ Mbps/ mo and for mid-size setups at \$95/Mbps/mo [20] for *guaranteed* bandwidth. Home user and small enterprise pricing benefits from economies of scale, e.g., Optimum Online provides 15/5 Mbps internet connection for small business starting at \$44.9/ mo [34]. Yet we note that the quoted bandwidth is not guaranteed. Figure 3 summarizes network service cost in the four environments.

	H, S	M	L
monthly	\$44.90	\$95	\$13
bandwidth (d/u)	15/5 Mbps	per 1Mbps	per 1Mbps
dedicated	No	Yes	Yes
picocent/bit	115/345	3665	500

Figure 3: Summarized network service costs [5].

The end-to-end cost of network transfer includes the cost on both communicating parties and the CPU overheads of transfering a bit from one application layer to another. Moreover, for reliable networking (e.g., TCP/IP) we need to also factor in the additional traffic and spent CPU cycles (e.g., SYN, SYN/ACK, ACK, for connection establishment, ACKs for sent data, window management, routing, packet parsing, re-transmissions). In the $S \rightarrow L$ scenario, it costs more than 900 picocents to transfer one bit reliably.

3. CRYPTOGRAPHY

So far we know that a CPU cycle will set us back 0.45-27 picocents, transferring a bit costs at least 900 picocents, and storing it costs under 100 picocents/year. We now explore the costs of basic crypto and modular arithmetic. All values are in picocents. Note that CPU cycles needed in cryptographic operations often vary with optimization algorithms and types of hardware used (e.g., specialized secure CPUs and crypto accelerators with hardware RSA engines [2] are cheaper per cycle than general-purpose CPUs).

	AES-128	AES-192	AES-256
S	1.42E+03	1.48E+03	1.52E+03
L	2.37E+01	2.47E+01	2.53E+01

Figure 4: AES-128, AES-192, AES-256 costs (per byte) on 64-byte input.

Symmetric Key Crypto. We first evaluate the per-bit costs of AES-128, AES-192, AES-256 and illustrate in Figure 4. The evaluation is based on results from the ECRYPT Benchmarking of Cryptographic Systems (eBACS) [7].

RSA. Numerous algorithms aim to improve the speed of RSA, mainly by reducing the time to do modular multiplications. In Figure 5, we illustrate the costs of RSA encryption/decryption using benchmark results from [7].

	1024 bit		2048 bit	
	Encrypt	Decrypt	Encrypt	Decrypt
S	3.74E+06	1.03E+08	8.99E+06	6.44E+08
L	6.24E+04	1.72E+06	1.50E+05	1.07E+07

Figure 5: Cost of RSA on 59-byte messages. (picocents)

PK Signatures. We illustrate costs of DSA, and ECDSA signatures based on NIST elliptic curves [7] in Figures 6, 7.

	1024 bit		2048 bit	
	Sign	Verify	Sign	Verify
S	5.73E+07	6.94E+07	1.89E+08	2.30E+08
L	9.55E+05	1.16E+06	3.15E+06	3.84E+06

Figure 6: DSA on 59-byte messages. The 1024-bit DSA uses 148-byte secret key and 128-byte public key. The 2048-bit DSA uses 276-byte secret key and 256-byte public key.

	ECDSA-163		ECDSA-409		ECDSA-571	
	KG/SGN	Verify	KG/SGN	Verify	KG/SGN	Verify
S	1.36E+08	2.65E+08	9.60E+08	1.91E+09	2.09E+09	4.18E+09
L	2.27E+06	4.41E+06	1.60E+07	3.19E+07	3.48E+07	6.96E+07

Figure 7: ECDSA signatures on 59-byte messages (curve over a field of size 2^{163}, 2^{409}, 2^{571} respectively). (picocents)

Cryptographic Hashes We also show per byte cost of MD5 and SHA1 with varied input sizes.

	MD5		SHA1	
	4096	64	4096	64
S	1.52E+02	3.75E+02	2.14E+02	6.44E+02
L	2.53E+00	6.25E+00	3.56E+00	1.07E+01

Figure 8: Per-byte cost of MD5 and SHA1 (varying inputs).

4. SECURE OUTSOURCING

Thus armed with an understanding of computation, storage, network and crypto costs, we now ask whether securing cloud computing against insiders is a viable endeavor.

We start by exploring what security means in this context. Naturally, the traditional usual suspects need to be handled in any outsourcing environment: (mutual) authentication, logic certification, inter-client isolation , network security as well as general physical security. Yet, all of these issues are addressed extensively in existing infrastructures and are not the subject of this work.

Similarly, for conciseness, within this scope, we will isolate the analysis from the additional costs of software patching, peak provisioning for reliability, network defenses etc.

4.1 Trust

We are concerned cloud clients being often reluctant to place sensitive data and logic onto remote servers without guarantees of compliance to their security policies [15, 23]. This is especially important in view of recent sub-poenas and other security incidents involving cloud-hosted data [11, 12, 29]. The viability of the cloud computing paradigm thus hinges directly on the issue of clients' trust and of major concern are cloud insiders. Yet how "trusted" are today's clouds from this perspective? We identify a set of scenarios.

Trusted clouds. In a *trusted* cloud, in the absence of unpredictable failures, clients are served correctly, in accordance to an agreed upon service contract and its (security) policies. No insiders act maliciously.

Untrusted clouds. For *untrusted* clouds, we distinguish several cases depending on the types of illicit incentives existing for the cloud and the client policies with which these will directly conflict. We call a cloud *data-curious* if insiders thereof have incentives to violate confidentiality policies (mainly) for (sensitive) client data. Similarly, in an *access-curious* cloud, insiders will aim to infer client access patterns to data or reverse-engineer and understand outsourced computation logic. A *malicious* cloud will focus mainly on (data and computation) integrity policies and alter data or perform incorrect computation.

Reasonable cloud insiders are likely to factor in the potential illicit gains (the incentives to violate the policy), the penalty for getting caught, as well as the probability of detection. Thus for most practical scenarios, insiders will engage in such behavior only if they can get away undetected with high probability, e.g., when no (cryptographic?) safeguards are in place to enable the detection.

4.2 Secure Outsourcing

Yet, millions of users embrace free web apps in **an untrusted provider model**. This shows that today's (mostly personal) cloud clients are willing to trade their privacy for (free) service. This is not necessarily a bad thing, especially at this critical-mass building stage, yet raises questions of clouds' viability for commercial, regulatory-compliant deployment, involving sensitive data and logic. And, from a bottom-line cost-perspective, is it worth even trying? This is what we aim to understand here.

In the following **we will assess whether clouds are economically tenable if their users do not trust them and therefore must employ cryptography and other mechanisms to protect their data.** A number of experimental systems and research efforts address the problem of outsourcing *data* to *untrusted service providers*, including issues ranging from searching in remote encrypted data to guaranteeing integrity and confidentiality to querying of outsourced data. In favor of cloud computing, we will set our analysis in the most favorable $S \rightarrow L$ scenario, which yields most CPU cycle savings.

4.2.1 The Case for Basic Outsourcing

Before we tackle cloud security, let us look at the simplest computation outsourcing scenario (where clients outsource data to the cloud, expect the cloud to process it, and send the results back). In existing work [5], we show that, to make (basic, unsecured) outsourcing cost effective, the cost savings (mainly from cheaper CPU cycles) need to outweigh the cloudâĂŹs distance from clients. In $S \rightarrow L$, outsourced tasks should perform at least 1,000 CPU cycles per every 32 bit data, otherwise it is not worth outsourcing them.

4.2.2 Encrypted Data Storage with Integrity

With an understanding of the basic boundary condition defining the viability of outsourcing we now turn our attention to one of the most basic outsourcing scenarios in which a single data client places data remotely for simple storage purposes. In the $S \rightarrow L$

scenario, the amortized cost of storing a bit reliably either locally *or remotely* is under 9 picocents/month (including power). Network transfer however, is of at least 900 picocents per accessed bit, a cost that is not amortized and two orders of magnitude higher.

From a technological cost-centric point of view it is simply not effective to store data remotely: **outsourced storage costs can be upwards of 2+ orders of magnitude higher than local storage** for the $S \rightarrow L$ scenario *even in the absence of security assurances*.

Cost of Security. Yet, outsourced storage providers exist and thrive. This is likely due to factors outside of our scope, such as the convenience of being able to have access to the data from everywhere or collaborative application scenarios in which multiple data users share single data stores (multi-client settings). Notwithstanding the reason, since consumers have decided it is worth paying for outsourced storage, the next question we ask is, how much more would security cost in this context?

Several existing systems encrypt data before storing it on potentially data-curious servers [8, 10, 30]. File systems such as I^{3}FS [22], GFS [17], and Checksummed NCryptfs [37] perform online real-time integrity verification.

It can be seen that two main assurances are of concern here: integrity and confidentiality. The cheapest integrity constructs deployed in most of the above revolve around the use of hash-based MACs. As discussed above, SHA-1 based keyed MAC constructs with 4096-byte blocks would cost around 4 picocent/byte on the server and 200 picocents/byte on the client side, leading to a total cost of about 25 picocents/bit. This is at least 4 times lower than the cost of storing the bit for a year and at least one order of magnitude lower than the costs incurred by transferring the same bit (at 900+ picocents/bit). Thus, **for outsourced storage, integrity assurance overheads are negligible.**

For publicly verifiable constructs, crypto-hash chains can help amortize their costs over multiple blocks. In the extreme case, a single signature could authenticate an entire file system, at the expense of increased I/O overheads for verification. Usually, a chain only includes a set of blocks.

For an average of twenty 4096 byte blocks[1] secured by a single hash-chain signed using 1024-bit RSA, would yield an amortized cost approximately 1M picocents per 4096-byte block (30+ picocents/bit) for client read verification and 180+ picocents/bit for write/signatures. This is up to **8 times more expensive than the MAC based case**.

4.2.3 Searches on Encrypted Data

Confidentiality alone can be achieved by encrypting the outsourced content before outsourcing to potentially access-curious servers. Once encrypted however, it cannot be easily processed by servers.

One of the first processing primitives that has been explored allows clients to search directly in remote encrypted data [4, 6, 13]. In these efforts, clients either linearly process the data using symmetric key encryption mechanisms, or, more often, outsource additional secure (meta)data mostly of size linear in the order of the original data set. This meta-data aids the server in searching through the encrypted data set while revealing as little as possible.

But is remote searching worth it vs. local storage? We concluded above that simply using a cloud as a remote file server is extremely non-profitable, up to several orders of magnitude. Could the searching application possibly make a difference? This would hold if either (i) the task of searching would be extremely CPU intensive allowing the cloud savings to kick in and offset the large losses due to network transfer, or (ii) the search is extremely selective and its results are a very small subset of the outsourced data set – thus amortizing the initial transfer cost over multiple searches.

We note that existing work does not support any complex search predicates outside of simple keyword matching search. Thus the only hope there is that the search-related CPU load (e.g., string comparison) will be enough cheaper in the cloud to offset the initial and result transfer costs.

Keyword searching can be done in asymptotically constant time, given enough storage or logarithmic if B-trees are used. While the client could maintain indexes and only deploy the cloud as a file server, we already discovered that this is not going to be profitable. Thus if we are to have any chance to benefit here, the index structures need to also be stored on the server.

In this case, the search cost includes the CPU cycle costs in reading the B-tree and performing binary searches within B-tree nodes. As an example, consider 32 bit search keys (e.g., as they can be read in one cycle from RAM), and a 1 TB database. 1-3 CPU cycles are needed to initiate the disk DMA per reading, and each comparison in the binary search requires another 1-3 cycles (for executing a comparison conditional jump operation). A B-tree with 16KB nodes will have approximately a 1000 fanout and a height of 4-5, so performing a search on this B-tree index requires about 100-300 CPU cycles. Thus in this simple remote search, $S \rightarrow L$ outsourcing would result in CPU-related savings of around 2,500-8,000 picocents per access. Transferring 32 bits from $S \rightarrow L$ costs upwards of 900 picocents. Outsourced searching becomes thus more expensive for any results upwards of 36 bytes per query. In reality this is much worse because of TCP overheads and potential need for a full 3 way handshake.

4.2.4 Insights into Secure Query Processing

By now we start to suspect that similar insights hold also for outsourced query processing. This is because we now know that (i) the tasks to be outsourced should be CPU-intensive enough to offset the network overhead – in other words, outsourcing peanut counting will never be profitable, and (ii) existing confidentiality (e.g., homomorphisms) and integrity (e.g., hash trees, aggregated signatures, hash chains) mechanisms can "secure" only very simple basic arithmetic (addition, multiplication) or data retrieval (selection, projection) which would cost under a few of cycles per word if done in an unsecured manner. In other words, *we do not know yet how to secure anything more complex than peanut counting*. And outsourcing of peanut counting is counter productive in the first place. Ergo our suspicion.

We start by surveying existing mechanisms. Hacigumus et al. [19] propose a method to execute SQL queries over partly obfuscated outsourced data to protect data **confidentiality** against a data-curious server. The main functionality relies on (i) partly obfuscating the outsourced data by dividing it into a set of partitions, (ii) query rewriting of original queries into querying referencing partitions instead of individual tuples, and (iii) client-side pruning of (necessarily coarse grained) results. The information leaked to the server is balancing a trade-off between client-side and server-side processing, as a function of the data segment size. [21] explores optimal bucket sizes for certain range queries.

Ge et al. [38] discuss executing aggregation queries with confidentiality on an untrusted server. Unfortunately, due to the use of extremely expensive homomorphisms this scheme leads to large processing times for any reasonably security parameter settings (e.g., for 1024 bit fields, 12+ days *per query* are required).

Other researchers have explored the issue of **correctness** in settings with potentially malicious servers. In a publisher-subscriber model, Devanbu et al. deployed Merkle trees to authenticate data published at a third party's site [14], and then explored a general model for authenticating data structures [26, 27]. In [31, 32] as well

[1]Douceur et al. [16], show that file sizes can be modeled using a log-normal distribution. E.g., for $\mu^e = 8.46$, $\sigma^e = 2.4$ and 20,000 files, the median file size would be 4KB, mean 80KB, along with a small number of files with sizes exceeding 1GB [3, 16].

as in [25], mechanisms for efficient integrity and origin authentication for selection predicate query results are introduced. Different signature schemes (DSA, RSA, Merkle trees [28] and BGLS [9]) are explored as potential alternatives for data authentication primitives. In [24, 36] *verification objects* VO are deployed to authenticate data retrieval in "edge computing" In [33, 35] Merkle tree and cryptographic hashing constructs are deployed to authenticate range query results.

To summarize, existing secure outsourced query mechanisms deploy (i) partitioning-based schemes and symmetric key encryption for ("statistical" only) confidentiality, (ii) homomorphisms for oblivious aggregation (SUM, COUNT) queries (simply too slow to be practical), (iii) hash trees/chains and (iv) signature chaining and aggregation to ensure correctness of selection/range queries and projection operators. SUM, COUNT, and projection usually behave linearly in the database size. Selection and range queries may be performed in constant time, logarithmic time or linear time depending on the queried attribute (e.g., whether it is a primary key) and the type of index used.

For illustration purposes, w.l.o.g., consider a scenario most favorable to outsourcing, i.e., assuming the operations behave linearly and are extremely selective, only incurring two 32-bit data transfers between the client and the cloud (one for the instruction and one for the result). Informally, to offset the network cost of $900 \times 32 \times 2 = 57,600$ picocents, only traversing a database of size at least 10^5 will generate enough CPU cycle cost savings. Thus it seems that with very selective queries (returning very little data) over large enough databases, outsourcing can break even.

Cost of Security. In the absence of security constructs, we were able to build a scenario for which outsourcing is viable. But what about a general scenario? What are the overheads of security there? It is important to understand whether the cost savings will be enough to offset them. While detailing individual secure query protocols is out of scope here, it is possible to reason generally and gain an insight into the associated order of magnitudes.

Existing integrity mechanisms deploy hash trees, hash chains and signatures to secure simple selection, projection or range queries. Security overheads would then include *at least* the (client-side) hash tree proof re-construction ($O(\log n)$ crypto-hashes) and subsequent signature verification of the tree's root. The hash tree proofs are often used to authenticate range boundaries. The returned element set is then authenticated often through either a hash chain (in the case of range joins, at least 30 picocents per byte) or aggregated signature constructs (e.g., roughly 60,000 picocents each, for selects or projections). This involves either modular arithmetic or crypto-hashing of the order of the result data set. For illustration purposes, we will again favor the case for outsourcing, and assume only crypto-hashing and a linear operation are applied.

Consider a database of $n = 10^9$ tuples of 64 bits each. In that case (binary) hash tree nodes need to be at least 240 bits (80 + 160 bits = 2 pointers + hash value) long. If we assume 3 CPU cycles are needed per data item, the boundary condition results in selectivity $s \leq 0.00037$ before outsourcing starts to make economical sense. In a more typical scenario of $s = 0.001$ (queries are returning 0.1% of the tuples), a per-query loss of over 0.3 US cents will be incurred.

The above holds only for the $S \rightarrow L$ scenario in which hash trees are deployed. In the case of signature aggregation [32, 33], the break-even selectivity would be even lower due to the higher computation overheads.

5. TO CONCLUDE

In this paper we explored whether cryptography can be deployed to secure cloud computing against insiders. We estimated common cryptography costs (AES, MD5, SHA-1, RSA, DSA, and ECDSA) and finally explored outsourcing of data and computation to untrusted clouds. We showed that deploying the cloud as a simple remote encrypted file system is extremely unfeasible if considering only core technology costs. We also concluded that existing secure outsourced data query mechanisms are mostly cost-unfeasible because **today's cryptography simply lacks the expressive power to efficiently support outsourcing** to untrusted clouds. Hope is not lost however. We found borderline cases where outsourcing of simple range queries can break even when compared with local execution. These scenarios involve large amounts of outsourced data (e.g., 10^9 tuples) and extremely selective queries which return only an infinitesimal fraction of the original data (e.g., 0.00037%).

6. REFERENCES

[1] Amazon Elastic Compute Cloud. Online at `http://aws.amazon.com/ec2`.

[2] IBM 4764 PCI-X Cryptographic Coprocessor. Online at `http://www-03.ibm.com/security/cryptocards/pcixcc/overview.shtml`, 2007.

[3] Nitin Agrawal, William J. Bolosky, John R. Douceur, and Jacob R. Lorch. A five-year study of file-system metadata. In *Proceedings of the 5th USENIX conference on File and Storage Technologies (FAST 07)*, Berkeley, CA, USA, 2007. USENIX Association.

[4] Georgios Amanatidis, Alexandra Boldyreva, and Adam O'Neill. Provably-secure schemes for basic query support in outsourced databases. In Steve Barker and Gail-Joon Ahn, editors, *DBSec*, volume 4602 of *Lecture Notes in Computer Science*, pages 14–30. Springer, 2007.

[5] Anonymized. Anonymized for double-blind submission, 2009.

[6] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 535–552. Springer, 2007.

[7] Daniel J. Bernstein and Tanja Lange (editors). ebacs: Ecrypt benchmarking of cryptographic systems. Online at `http://bench.cr.yp.to` accessed 30 Jan. 2009.

[8] M. Blaze. A Cryptographic File System for Unix. In *Proceedings of the first ACM Conference on Computer and Communications Security*, pages 9–16, Fairfax, VA, 1993. ACM.

[9] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EuroCrypt*, 2003.

[10] G. Cattaneo, L. Catuogno, A. Del Sorbo, and P. Persiano. The Design and Implementation of a Transparent Cryptographic Filesystem for UNIX. In *Proceedings of the Annual USENIX Technical Conference, FREENIX Track*, pages 245–252, Boston, MA, June 2001.

[11] CNN. Feds seek Google records in porn probe. Online at `http://www.cnn.com`, January 2006.

[12] CNN. YouTube ordered to reveal its viewers. Online at `http://www.cnn.com`, July 2008.

[13] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 79–88, New York, NY, USA, 2006. ACM.

[14] Premkumar T. Devanbu, Michael Gertz, Chip Martel, and Stuart G. Stubblebine. Authentic third-party data publication. In *IFIP Workshop on Database Security*, pages 101–112, 2000.

[15] Donna Bogatin. Google Apps data risks: Security vs. privacy. Online at `http://blogs.zdnet.com/micro-markets/?p=1021`, February 2007.

[16] John R. Douceur and William J. Bolosky. A large-scale study of file-system contents. In *Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 59–70. ACM New York, NY, USA, 1999.

[17] S. Ghemawat, H. Gobioff, and S. T. Leung. The Google File System. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, pages 29–43, Bolton Landing, NY, October 2003. ACM SIGOPS.

[18] Google Inc. Google App Engine. Online at `http://code.google.com/appengine/`.

[19] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 216–227. ACM Press, 2002.

[20] James Hamilton. Internet-scale service efficiency. Large Scale Distributed Systems & Middleware (LADIS 2008),, 2008.

[21] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *Proceedings of ACM SIGMOD*, 2004.

[22] A. Kashyap, S. Patil, G. Sivathanu, and E. Zadok. I3FS: An In-Kernel Integrity Checker and Intrusion Detection File System. In *Proceedings of the 18th USENIX Large Installation System Administration Conference (LISA 2004)*, pages 69–79, Atlanta, GA, November 2004. USENIX Association.

[23] Larry Dignan. Will you trust Google with your data? Online at `http://blogs.zdnet.com/BTL/?p=4544`, February 2007.

[24] M. Atallah and C. YounSun and A. Kundu. Efficient Data Authentication in an Environment of Untrusted Third-Party Distributors. In *24th International Conference on Data Engineering ICDE*, pages 696–704, 2008.

[25] Maithili Narasimha and Gene Tsudik. DSAC: integrity for outsourced databases with signature aggregation and chaining. Technical report, 2005.

[26] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. Stubblebine. A general model for authenticated data structures. Technical report, 2001.

[27] Charles Martel, Glen Nuckolls, Premkumar Devanbu, Michael Gertz, April Kwong, and Stuart G. Stubblebine. A general model for authenticated data structures. *Algorithmica*, 39(1):21–41, 2004.

[28] R. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Research in Security and Privacy*, 1980.

[29] Jeralyn Merritt. What google searches and data mining mean for you. Online at `http://www.talkleft.com/story/2006/01/25/692/74066`.

[30] Microsoft Research. Encrypting File System for Windows 2000. Technical report, Microsoft Corporation, July 1999. `www.microsoft.com/windows2000/techinfo/howitworks/security/encrypt.asp`.

[31] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. In *Proceedings of Network and Distributed System Security (NDSS)*, 2004.

[32] E. Mykletun, M. Narasimha, and G. Tsudik. Signature bouquets: Immutability for aggregated/condensed signatures. In *Computer Security - ESORICS 2004*, volume 3193 of *Lecture Notes in Computer Science*, pages 160–176. Springer, 2004.

[33] Maithili Narasimha and Gene Tsudik. Authentication of Outsourced Databases using Signature Aggregation and Chaining. In *Proceedings of DASFAA*, 2006.

[34] Optimum. Optimum online plans. Online at `http://www.buyoptimum.com`.

[35] HweeHwa Pang, Arpit Jain, Krithi Ramamritham, and Kian-Lee Tan. Verifying Completeness of Relational Query Results in Data Publishing. In *Proceedings of ACM SIGMOD*, 2005.

[36] HweeHwa Pang and Kian-Lee Tan. Authenticating query results in edge computing. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, page 560, Washington, DC, USA, 2004. IEEE Computer Society.

[37] G. Sivathanu, C. P. Wright, and E. Zadok. Enhancing File System Integrity Through Checksums. Technical Report FSL-04-04, Computer Science Department, Stony Brook University, May 2004. `www.fsl.cs.sunysb.edu/docs/nc-checksum-tr/nc-checksum.pdf`.

[38] Tingjian Ge and Stan Zdonik. Answering aggregation queries in a secure system model. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 519–530. VLDB Endowment, 2007.

[39] Whitfield Diffie. How Secure Is Cloud Computing? Online at `http://www.technologyreview.com/computing/23951/`, November 2009.