On the (Im)Practicality of Securing Untrusted Computing Clouds with Cryptography

Yao Chen and Radu Sion

Stony Brook Network Security and Applied Cryptography Lab

{yaochen,sion}@cs.stonybrook.edu

February 24, 2010

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*". In this paper we aim to understand whether this is truly the case and quantify just *how* expensive it is to secure data processing in untrusted, potentially curious clouds.

We start by looking at the economics of computing in general and clouds in particular. Specifically, we derive the end-to-end cost of a CPU cycle in various environments and show that its cost lies between 0.58 picocents in efficient clouds and 26.02 picocents for small business deployment scenarios (1 picocent = 1×10^{-14}), values *validated against current cloud pricing*. We then evaluate the cost of networking and show that, in order to offset the costs of networking, cloud computing makes economical sense only for compute intensive applications requiring at least 3800 compute cycles per each 32 bits of transferred input.

Finally, we explore the cost of common cryptography primitives as well as the viability of their deployment for cloud security purposes. We conclude that Diffie was correct. Securing outsourced data and computation against untrusted clouds is indeed costlier than the associated savings, with outsourcing mechanisms up to 5+ orders of magnitudes costlier than their non-outsourced locally run alternatives.

This is simply because today's cryptography does not allow for *efficient* oblivious processing of *complex enough* functions on encrypted data. And outsourcing simple operations – such as existing research in querying encrypted data, keyword searches, selections, projections, and simple aggregates – is simply not profitable (too few compute cycles / input word to offset the client's distance from the cloud). Thus, while traditional security mechanisms allow the elegant handling of inter-client and outside adversaries, today it is still too costly to secure against cloud insiders with cryptography.

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.

The outsourcing concept incorporates a wide range of flavors. On the one hand, global-wide giants such as Google offer mostly managed "cloud" facilities as part of infrastructures composed of tens of thousands of nodes. At the other side of the spectrum we find a plethora of small and medium sized startups or more established companies such as RackSpace, Mosso, InetU, hosting.com, Verio, FastServers, and tens of others.

These companies offer services ranging from simple, raw networked hardware hosting client-provided operating system images and/or applications to more complex infrastructure setups with specifically deployed and managed application servers ready to support clients' workloads.

Current clouds seem to be well suited and cost-effective for personal and small enterprise clients that increasingly outsource data-driven web-based retail and end-user interfaces and minimize their in-house computing management footprints. Yet clouds have been somewhat less successful in attracting medium to large corporations. Such clients often fall under strict regulatory compliance requirements for manipulating information or simply are reluctant to place sensitive data and computation logic under the control of a remote, third-party provider, without practical assurances of privacy and confidentiality in which the provider is untrusted [72, 40].

Existing secure outsourcing research addressed several issues ranging from searching to guaranteeing integrity and confidentiality of querying of outsourced encrypted data. To protect sensitive data from untrusted servers, confidentiality alone can be achieved by encryption. Once encrypted however, data cannot be easily processed by the server. This limits the applicability of outsourcing, as the type of processing primitives available are reduced dramatically.

However, so far, the end-to-end viability of outsourcing has mostly not been explored or questioned. Is a remotely hosted computing cycle in the cloud indeed cheaper than performing it locally *when considering the end-to-end bottom-line*? Yet, at least outside security considerations, it seems the markets have spoken and the increasing number of service providers can be viewed as testimony that this indeed is the case.

Yet, regarding the security aspects of outsourced computing, impracticality results [110] hint at the fact that the answer is probably not clear cut. Different applications' computing and data models will fare differently and their end-to-end costs may or may not favor cloud computing. More importantly, given the integrity, confidentiality and privacy constraints discussed above, security needs to become a first-class citizen in clouds, especially when sensitive data should not be disclosed. This begs the question: *is secure processing on behalf of clients possible in untrusted clouds*?

Other security dimensions such as inter-application isolation and in-transit confidentiality against eavesdropping have been naturally demanded already and are (at least partially [105]) provided by virtualization and appropriate network security technology. Authentication and physical security are also important and have been studied extensively [36, 96, 12]. Similarly, for conciseness, we will not consider here the additional costs of software patching, peak-provisioning for reliability, network defenses etc. It can be easily shown that adding these in would only strengthen the arguments.

Yet, how about *data integrity, confidentiality or privacy for curious, untrusted clouds*? 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 [116] Whitfield Diffie, while being asked about encrypted search and encrypted computation in the cloud, said: "The whole point of cloud computing is economy [...]. It has been shown to be possible in principle for the 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**."

In this work 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?

To this end, we deploy the insights gained into the costs of computing primitives (CPU cycles, networking, storage) and the viability of outsourcing, which we derived elsewhere [29], to evaluate the costs of cryptography and its practicality in today's clouds.

Ultimately, we show that today's secure data outsourcing primitives are up to 4-5 orders of magnitude more expensive than local execution, mainly due to the fact that we do not know how to process complex

functions on encrypted data efficiently enough. And outsourcing simple operations – such as existing research in querying encrypted data, keyword searches, selections, projections, and simple aggregates – is simply not profitable. Thus, while traditional security mechanisms allow the elegant handling of inter-client and outside adversaries, today it is still too costly to secure against cloud insiders with cryptography.

2 Cost Models: CPU, Network, Storage

Elsewhere [29] we explored 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. For completeness here we outline some of the main results.

2.1 Computing Environments

To understand CPU cycles, we explore first the cost of computing in different environments of increasing complexity: home, small, mid-size and large size data centers [1].

Home Users (H). We include this scenario as a baseline for a simple home setup containing several computers. This could correspond to individuals with spare time to maintain a small set of computers, or a very small home-based enterprise with no staffing overheads. This niche is important as it features a set of peculiarities, including access to residential energy pricing, negligible cooling, rental and management costs (as we will not factor such individuals' time in).

Parameters	H	S	M	L
CPU utilization	5-8%	10-12%	15-20%	40-56%
server:admin ratio	N.A.	100-140	140-200	800-1000
Space (sqft/month)	N.A.	$0.5	$0.5	$0.25
PUE	N.A.	2-2.5	1.6-2	1.2-1.5

Figure 1: Sample key parameters.

Small Enterprises (S). We consider here any scenario involving an infrastructure of up to 1000 servers run in-house in a commercial enterprise. The cost structure will start to feature most of the usual suspects, including commercial energy pricing, cooling, space leases, staffing etc. Small enterprises however can not afford custom hardware, efficient power-distribution, and cooling or dedicated buildings among others. More importantly, in addition to power distribution inefficiencies, due to their nature, small enterprises cannot be run at high utilization as they would be usually under the incidence of business cycles and its associated peak loads. Traditional data center utilization is often below 20% [42, 15] and for small enterprises without good design and management, the number would be even lower (10-15%).

Mid-size Enterprises (M). We consider here setups of up to 10,000 servers, run by a corporation, often in its own dedicated data center(s). Mid-size enterprises might have some clout and access to better service deals as well as more efficient cooling and power distribution. They are usually not fully global, yet could feature several centers across one or two time zones, allowing increased independence from local load cycles as well as the ability to handle daily peaks better by shifting loads across timezones. All the above will result ultimately in increased utilization (20-25% est.) and overall efficiency.

Large Enterprises/Computing Clouds (L). Computing Clouds (such as Amazon and Google), and large enterprises run over 10,000 servers, cross multiple time-zones, often literally at a global level, with large data centers distributed across all continents and often in tens to hundreds of countries. For example Google has built a 30-acre site in Dallas, Oregon, next to a hydroelectric dam providing cheap power. The site is composed of 34,000 square feet buildings [63]. Especially in cloud setups, high speed networks allow global-wide distribution and integration of load from thousands of individual points of load. This in turn

---

[1]We note it is not the subject of our work to explore in-depth data center infrastructures. A plethora of online sources discuss issues related to data centers, often focusing on power and overall efficiency (most notably James Hamilton's blog [53]).

flattens the 24-hour overall load curve and allows for efficient peak handling and comparably high utilization factors (50-60% est. [55]). Cloud providers have the clout to ask vendors for custom designed hardware and power supply components [78, 55]. Moreover, these providers run the most efficient infrastructures, and often are at the forefront of innovation. In one notorious instance, Google for example asked Intel for chips tolerating more heat, to allow for a few degrees increase in data center operating temperatures - which in turn increases cooling efficiency by whole percentage points [86].

2.2 Factors

A number of cost factors come into play across all of the above levels (see Figure 2). These factors can be divided into a set of inter-dependent vectors, including: hardware (servers, networking gear), building (floor space leasing), energy (running hardware and cooling), service (administration, staffing, software maintenance), and network service. Other breakdown layouts of these factors are possible [14].
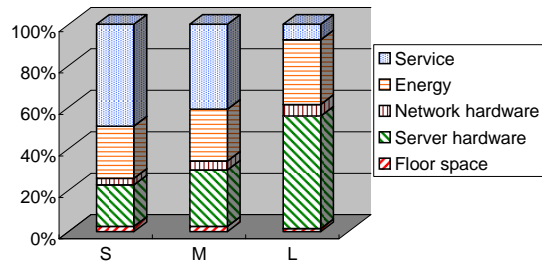

Figure 2: Impact of factors on the cost of one CPU cycle for (S)mall (up to a 1000 servers), (M)edium (up to 10000 servers), and (L)arge sized (clouds) data centers.

Server Hardware. Hardware costs include servers, racks, power equipment, network equipment, cooling equipment etc. Naturally, there are different choices for data centers to increase capacity. *Up-scaling* – the purchase of a smaller number of more expensive off-the-shelf multi-blade servers – is often considered in mid-size enterprises, and features lower software and infrastructure cost advantages. Scaling out – deploying massive numbers of low-cost, almost "expendable" custom-designed and often in-house built multi-CPU server boards – is a strategy available to large, cloud-size providers such as Google and Amazon. The advantages of this approach are low hardware costs, low inter-failure correlation and high overall efficiency factors. Sometimes these two approaches can be combined; e.g., servers embedded with 4-8 CPUs can be considered as scale-out architecture of scale-up nodes [50].

We note that these costs drop with time, likely even by the time this goes to print. For example, while many of the current documented mid-size deployments use single or multi-CPU System-X blade servers at around $1-2000 each [59], (with 2.1 GHz CPU and a life span of 5 years) and large data centers deploy custom setups at about $3000 for 4 CPUs, near-future developments could yield important changes. In one documented instance, e.g., Amazon is working with Rackable Systems to deliver an under $700 AMD-based 6 CPU board dubbed CEMS (Cooperative Expendable Micro-Slice Servers) V3. Such CEMS will significantly reduce hardware costs at the only likely expense of increased service costs – as the unit power source can become a single point of failure that will bring down 6 CPUs simultaneously – thus decreasing the average mean time to failure per CPU [54]. We will be conservative and empirically assume home PC prices of around $750/CPU, small and mid-size enterprise costs of around $1000/CPU (for 2 CPU blades) and cloud-level costs of no more than $500/CPU.

Energy. Energy in data centers does not only include power, computing and networking hardware but the entire support infrastructure, including cooling, physical security, and overall facilities. With the increasing density of today's rack structure, temperature rises more rapidly than in old server rooms [13]. For example, any additional 40 watts per square foot can lead to a rise of 25 degrees F in 10 minutes. A simple rough way to infer power costs is by estimating the Power Usage Efficiency (PUE) of the data center. The PUE is a metric defined by the GreenGrid Consortium to evaluate the energy efficiency of a data center [49] (PUE = Total Power Usage / IT Equipment Power Usage). PUE ranges from 1.13-1.21 for big providers such as Google and 1.22 for efficient data center containers by Microsoft, to over 2 for typical data centers [89, 101]. We will assume 1.2-1.5 PUE for large enterprises, 1.6-2 PUE for mid-size enterprises and 2-2.5 for small

enterprises. Costs of electricity are relatively uniform and documented [8]. Interestingly, residential pricing is cheaper, possibly up to the point of justifying running server farms in apartments [53].

Service. Evaluating the staffing requirements for data centers is an extremely complex endeavor as it involves a number of components such as software development and management, hardware repair, maintenance of cooling, building, network and power services [14].

We started attacking the problem analytically but quickly realized only sparse relevant supporting data sets are available. We then focused on how hardware failure rates (MTBF) and mean time to repair (MTTR) factors impact the number of required service personnel. While large scale failure data is hard to come by, especially for large cloud providers, we identified a relevant data set from the Los Alamos National Laboratory [71, 106] describing more than 23000 failures experienced over 9 years in the lab's data centers. Failure rates have been pegged at up to 3 failures per CPU per year. The *median* time to repair experi-

Figure 3: CPU cycle costs

enced was about 59 minutes (average was 417.54 mins). Under the assumption that the MTTR approximately reflects the time spent by personnel (arguable), this allows the estimation of the number of hardware maintenance man-hour figures and thus the required staff to server ratio. For large data centers, this yields a surprisingly close-to-reality ratio of CPUs to personnel of up to 1600:1 (for 3 failures/CPU). Naturally, hardware failures are only one of many tasks to be handled and thus in reality these ratios are much lower (below 1000:1).

We validated and deployed a set of commonly accepted rule of thumb values that have been empirically developed and validate well [56]. For example, the server to administrator ratio can vary from 2:1 up to experimental 2500:1 values [56] due to different degrees of automation and data management. In deployment, small to mid-size data centers feature a ratio of 100-140:1 whereas clouds can go up to 1000:1 [55, 48].

2.3 CPU Cycles

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

Figure 4: Current clouds' CPU cycle costs.

Armed with the above factors, the amortized cost of a CPU cycle in various environments characterized by the parameters in Figure 1 can be computed. This cost (Figure 3) ranges from 0.58 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 26.02 picocents (*1 US picocent* = 10^{-14} *USD*). Not surprisingly, the cost is cheaper in home environments (H) than in the small enterprise setting (S).

Validation. We validated these numbers with the pricing points of main current cloud providers: Amazon [1], Google [47] and Microsoft [2] Windows Azure [87] (Figure 4). For example, as of this writing, Amazon is featuring a multi-level pricing scheme in which 1.2 GHz virtual EC2 CPU cores are packaged in different offerings including single CPU to multi-core, multi-CPU settings. 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.

---

[2]Interestingly, it can be seen that the Microsoft basic configuration pricing is up to 15% lower than that of Amazon and Google, suggesting only minimal profit and the desire to attract a critical mass of initial adopters as they enter the market.
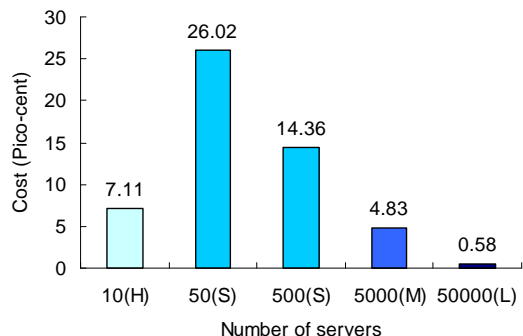
Disk	cap. (GB)	price (USD)	Adj. MTBF (mil.hrs)	amort. acq. (pcent/bit/yr)	power seek (W)	power2 idle (W)	power3 (W)	power cost (pcent/bit/yr)	total cost (pcent/bit/yr)	acq. %	avg. seek time (ms)	avg. seek4 cost (pcents)	power5 read (W)	read cost (pcent/bit)
Maxtor Diamond Max	500	53	0.35	32.89	13.6	8.10	10.85	237.62	270.50	12.16	9.00	377542	11.16	0.03
Hitachi Deskstar 7k500	500	67	0.29	49.89	15	9.60	12.30	269.37	319.26	15.63	8.50	407953		
Hitachi Ultrastar A7K1000	1024	153	0.35	46.36	14	9.00	11.50	122.97	169.33	27.38	8.20	417631		
WD Caviar GP Low Power	1024	103	0.29	37.45	7.5	4.00	5.75	61.49	98.93	37.85	8.90	271994	7.40	0.02
Seagate Barracuda 7200.10	750	63	0.35	26.06	12.6	9.30	10.95	159.87	185.93	14.02	9.25	369615	13.00	0.06
WD Caviar SE16	500	62	N/A		8.77	8.40	8.59	188.01			9.90		8.77	0.04
Samsung SSD	32	269	0.29	3129.65	1	1.00	1.00	342.19	3471.83	90.14	1.70	47912	0.5	0.0017
Intel SSD X18-M	80	389	0.35	1508.59	0.15	0.06	0.11	14.37	1522.96	99.06			0.15	0.0002
Intel SSD X25-M	160	765	0.35	1483.38	0.15	0.06	0.11	7.19	1490.57	99.52			0.15	0.0002

Figure 5: Magnetic disk storage costs.

2.4 Storage

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. Figure 5 shows the costs of ownership and operation of a representative sample (by no means exhaustive) set of commonly available consumer-level disks at the time of this writing. Costs incorporate energy and amortized acquisition components. The dominant factor is energy at 60-70% of the total cost. We also note that in 2007 Schroeder et al. showed that actual observed MTBF numbers (for 100,000 disks over periods covering up to 5 years) are often up to about 3.4 times lower than advertised [107]. We considered this in Figure 5.

2.5 Network Service

Published network service cost numbers place network service costs for large data centers at around $13/ Mbps/ month and for mid-size setups at $95/Mbps/month [55] for *guaranteed* bandwidth. Similar pricing schemes have been quoted to us by network providers [35]. Home user and small enterprise pricing benefits from economies of scale, yet we note that the quoted bandwidth is not

provider	monthly	bandwidth (d/u)	picocent/bit
Optimum online	$29.95	15 Mbps /5 Mbps	77/231
Opt. online boost	$44.9	30 Mbps /5 Mbps	58/346
Optimum Lightpath	>$1000	5-1000 Mbps	5000 (est.)
Verizon Starter Plan	$19.99	1 Mbps/384 Kbps	771/2008
Verizon Power Plan	$29.99	3 Mbps/768 Kbps	386/1506
Verizon Turbo Plan	$42.99	7.1 Mbps/768 Kbps	233/2160
Mid-size	$95 (est.)	1 Mbps (dedicated)	3665 (est.)
Large/cloud	$13 (est.)	1 Mbps (dedicated)	500 (est.)

Figure 6: Different network service pricing levels [95, 35, 60, 115, 55].

guaranteed and refers only to the hop connecting the client to the provider. Figure 6 summarizes these costs. The retail numbers consider bulk-pricing with unlimited transfer caps, as is the case with most current providers. Recently however, several providers have (voiced intentions to) set transfer caps [10, 74].

Per bit transfer cost	
H → cloud	800
S → cloud	6,000
M → cloud	4,500

Figure 7: Transfer costs.

To evaluate the end-to-end picture we start by noting that, in current markets, costs will be incurred by both communicating parties. Also, transferring a bit from one application layer to another includes CPU overheads, e.g., bringing the bit from the hardware all the way to the application. We will empirically assume this involves at a minimum about 20 CPU cycles per 32 bit value, although real-life costs are likely application specific. This results in additional end-to-end costs of at least 90-100 picocents in the most favorable scenario (see below). 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 most favorable scenario, reliable transmission (ignoring window management, packet routing, parsing CPU cycles, and connection establishment) of a 1460 byte full MTU (in a 1500 byte packet = 12000 bits) from a home to a cloud application layer would cost often at least 2 packets (data + ACK packet = 320 bits, unless ACK optimizations are deployed) and 20 cycles/32 bit word on both the home and the cloud CPUs. We generalize the per bit transfer cost between computing environments a and b:

$$Trans_{a \to b} = \{(1 + \gamma)(S_p \times (U_a + D_b) + 0.5S_a \times (D_a + U_b)$$
$$+ TCP_{32} \times (c_a \times S_p/32 + c_b \times (S_p/32 + 0.5S_a/32))\}/PayloadSize$$

where γ is the TCP re-transmission rate, U_a, D_a, U_b, D_b are the upload and download costs per bit for a and b, c_a, c_b are their CPU cycle costs, S_p and S_a are the size of a data packet and a ACK packet, and TCP_{32} is the additional CPU cycles needed per 32bit word (e.g., go through the entire network stack, etc.) As an example, in the H $\to$ cloud case, considering a conservative 1% TCP/IP re-transmission rate this yields nearly 800 picocents per bit transfer from H $\to$ cloud.

If the protocols are not optimized to fully utilize payloads these costs can and will be higher, e.g., for 32 bit payloads, we would incur upwards of 10,000 picocents/bit.

3 Cryptography

So far we know that a CPU cycle will set us back 0.5-26 picocents, transferring a bit costs 800-6,000 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 unless specified otherwise. 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 [6] are likely cheaper per cycle than general-purpose CPUs).

	AES	**DES**	**TDES**
	128 bits	64 bits	64 bits
H	13	37	103
S	25	76	208
M	8	26	70
L	1	3	8

Figure 8: AES, DES costs (per bit).

Symmetric Key Crypto. We first evaluate the per-bit cost of AES, DES and TDES and illustrate in Figure 8. We base our results on experimental data on CPU cycle counts on real hardware [108, 22].
Modular Multiplication. Beyond Montgomery reduction [83, 90], [73, 76] which results in a cost of $2m^2 + 2m$ single-precision operations (m is the number of digits in the operands) a multitude of results have aimed at reducing the constants. For example, the method in [103] yields the following execution times:

$$t_{mul} \approx (m^2 + 7m)t_d + (4m^2 + 20m)t_a + (4m^2 + 2m)t_{mem} \quad (1)$$

where we denoted by t_d, t_a and t_{mem} the single-precision multiplication, addition and memory access times. For illustration purposes we will make a few simplifying assumptions. We will ignore additions and memory accesses as well as insignificant factors in (1). We will approximate the number of digits in the operands by $m \approx \frac{|N|}{d}$ where $|N|$ is the bit-size of N and d is the bit-size of a digit. We define:

	512bit	**1024bit**	**2048bit**
H	5.58E+5	2.15E+6	8.48E+6
S	1.12E+6	4.34E+6	1.71E+7
M	3.79E+5	1.46E+6	5.76E+6
L	4.55E+4	1.75E+5	6.92E+5

Figure 9: Modular Multiplication

$$t_{mul}(|N|) \approx (\frac{|N|}{d})^2 \times t_d \quad (2)$$

Normally [73, 76] we have $d = log_2(10) \approx 3.3$. The decision for the value of d should be made based on the computing platform and the programming language used to implement modular reduction [23]. To account for pipe-lining and inter-ALU optimizations on x86 platforms, we use the empirical approximation of $d \approx 5$ and assume 1 cycle for addition and 3 cycles for multiplication [61]. Figure 9 illustrates the resulting lower bound costs.

	1024 bit			2048 bit		
	KeyGen	Sign	Verify	KeyGen	Sign	Verify
H	1.17E+9	2.72E+7	8.40E+5	8.20E+9	1.70E+8	2.02E+6
S	2.37E+9	5.50E+7	1.69E+6	1.65E+10	3.43E+8	4.09E+6
M	7.97E+8	1.85E+7	5.70E+5	5.57E+9	1.15E+8	1.40E+6
L	9.57E+7	2.22E+6	6.85E+4	6.69E+8	1.38E+7	1.65E+5

Figure 10: RSA signatures on 59-byte messages. (picocents)

Modular Exponentiation. Fast modular exponentiation today is achieved using mainly repeated squaring techniques. The time to perform a modular exponentiation for an N-bit base and exponent values is $t_{exp}(|N|) = |N|t_{sq}(|N|)$

where $t_{sq}(|N|)$ is the time to perform a modular squaring operation. For example, S.-M. Hong et al. [112] provide an efficient modular squaring algorithm of m-digit values, using m^2 single-precision multiplications. Then, $t_{sq}(|N|) \approx t_{mul}(|N|)$, which is found in equation (2). Thus $t_{exp}(|N|) \approx |N|^3/M \times d^2$.

Key Sizes. For *computationally* bounded adversaries, it is important to assess associated bounds and relate them to the deployed privacy-enabling trapdoor. RSA Labs [7] has started evaluating and recommending key sizes for RSA since 1995 [5] when 768 bit sizes were deemed appropriate for most application. Both the RSA [4] and the National Institute of Standards and Technology (NIST) key schedules [94] proposed

	ECDSA-163		ECDSA-409		ECDSA-571	
	KG/SGN	Verify	KG/SGN	Verify	KG/SGN	Verify
H	30	70	250	500	570	1100
S	70	140	500	1020	1100	2220
M	20	50	170	340	370	740
L	2	6	20	40	45	90

Figure 11: ECDSA (NIST B-163 curve) signatures on 59-byte messages (curve over fields of size 2^{163},2^{409},2^{571} resp.). (**micro**cents)

1024 bits minimum until 2010 (corresponding to 80 bits of secrecy in a symmetric key scenario). Secrets that are to live beyond 2010, but not after 2030, are to be protected by minimum 2048 bit RSA keys (112 bits for symmetric keys), and 3072 bits RSA (128 bits for symmetric keys) are the minimum for anything that is to survive after 2030 [4, 3].

	512 bit		1024 bit		2048 bit	
	Encrypt	Decrypt	Encrypt	Decrypt	Encrypt	Decrypt
H	3.23E+6	4.36E+5	2.52E+7	1.72E+6	2.00E+8	6.84E+6
S	6.53E+6	8.82E+5	5.10E+7	3.48E+6	4.04E+8	1.38E+7
M	2.20E+6	2.96E+5	1.71E+7	1.17E+6	1.35E+8	4.65E+6
L	2.64E+5	3.56E+4	2.06E+6	1.40E+5	1.63E+7	5.58E+5

Figure 12: Cost of RSA. (picocents)

Additionally, the NESSIE (New European Schemes for Signatures Integrity and Encryption) Project [2] recommended a minimum of 1536 bits for RSA signatures in 2004.

RSA. Using modular exponentiation, RSA public key encryption takes $O(k^2)$, private key decryption $O(k^3)$, and key generation $O(k^4)$

steps, where k is the number of bits in the modulus [70]. Numerous algorithms aim to improve the speed of RSA, mainly by reducing the time to do modular multiplications. For using Montgomery reduction and Chinese remainder, the number of CPU cycles needed for RSA encryption/decryption becomes [68]:

$$T_{enc} = (\frac{5Pw}{8} + \frac{23Aw}{16})s^3 + (\frac{Pw}{2} + \frac{9Aw}{9})s^2 + (Aw - \frac{3P}{2} - \frac{A}{2})s + 2(w-1)(P+A)$$

$$T_{dec} = (w-1)(P+A) + (s^2+s)wA + (b_k-1)[\frac{3s^2+s}{2}P + \frac{7s^2+s}{2}A] + (h_k-1)[(2s^2+s)P + \frac{9s^2+3s}{2}A]$$

	1024 bit			2048 bit		
	KeyGen	Sign	Verify	KeyGen	Sign	Verify
H	1.42E+7	1.51E+7	1.83E+7	4.90E+7	4.98E+7	6.07e+7
S	2.87E+7	3.05E+7	3.69E+7	9.90E+7	1.01E+8	1.23E+8
M	9.67E+6	1.03E+7	1.24E+7	3.33E+7	3.38E+7	4.12E+7
L	1.16E+6	1.23E+6	1.49E+6	4.00E+6	4.06E+6	4.95E+6

Figure 13: DSA on 59-byte messages.

where s is the word size of modulus n, b_k is the bit size of public key k, h_k is the Hamming weight of k, w is the bit size of a word, and the CPU cycles needed for addition and multiplication of single-precision integers are A and P. We assume the private key has the same size as n. We illus-

trate the cost of RSA in Figure 12. Note that, as expected, DES/AES are 2-4 orders of magnitude faster and cheaper, often depending on implementation.

PK Signatures. We illustrate DSA, RSA and an ECC-based signature based on the standard NIST B-163 elliptic curve. Results are from the ECRYPT Benchmarking of Cryptographic Systems (eBACS) [17].

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

	MD5			SHA1		
bytes	4096	64	8	4096	64	8
H	40	90	460	100	220	1000
S	70	190	940	100	440	1880
M	20	60	320	70	150	640
L	3	8	30	8	17	80

Figure 14: Per-byte cost of hashing (varying inputs)

4 Outsourcing

Thus armed with an understanding of computation, network and crypto costs, we now explore whether securing clouds 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 [105], network security as well as general physical security [36, 96, 12]. Yet, *all of these issues have been and 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. We note however, that it can be easily shown that adding these in would only strengthen the discourse.

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 [72, 40]. This is especially important in view of recent sub-poenas and other security incidents involving cloud-hosted data [33, 34, 85]. 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 basic 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 (often 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. For clouds, the penalties of "getting caught" almost always will include the indirect (often steep) cost of a tarnished reputation (one of the central assets) expressed in a direct reduction in the number of clients and thus bottom-line revenue. This is likely to trickle down to inside enforcement mechanisms and policies. Moreover, proof of explicit malicious behavior is almost certainly leading to the demise of the company. 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 detection.

4.2 Existing Providers

Traditional out-sourcing: IBM Blue Cloud. IBM re-branded its traditional outsourcing paradigm Blue Cloud, added virtualization functionality based on Linux, Hadoop, Xen, or PowerVM and provides administration functionality through several third party vendors, e.g., 3Tera and oppsource.

9

Clouds: Amazon AWS. Amazon offers a set of storage (S3) and computation (EC2) outsourcing facilities structured and deployed as web services. These include a number of additional mechanisms and information portals such as queuing, website statistics, e-commerce, payments, billing, structure database, authentication, shipment, as well as a content delivery network infrastructure. The main computation outsourcing unit is a virtual machine image that is networked and loaded transparently. Third parties such as rightScale provide access and effective control.

Clouds: Google Apps. The Google App engine allows the transparent deployment of python applications in its infrastructure. It features dynamic web serving, persistent storage, load balancing, authentication and mail APIs. Apps run in sandboxed platform-independent environments.

Clouds: Windows Azure. Recently, Microsoft has also introduced its cloud, dubbed Windows Azure aimed at hosting .Net apps in both managed and un-managed modes. Windows Server is the base OS and Hyper-V is the virtualization layer.

Clouds/Managed servers: RackSpace.com. RackSpace is a company that provides both managed and un-managed hosting as well as a cloud-computing interface dubbed Mosso. As such it lies at the crossroads between clouds and simple rack hosting. As a managed provider, it offers the ability to lease a number of server units together with pre-installed software that will be managed as part of the service. Clients can run logic and place data on these units, yet their management is controlled by the provider.

Un-managed hardware: Hostway.com. Hostway offers un-managed custom Dell server configurations. In un-managed setups clients lease an actual piece of hardware and are provided with full control thereof, including superuser access. The server is networked and possibly minorly managed at OS level (to install patches and software updates) by the provider. DNS, network connectivity, hardware uptime (and often backups) are guaranteed.

Virtually *all of the above operate in the trusted model*. Clients are offered written contractual guarantees for the cloud's compliance with certain integrity policies (and very rarely confidentiality assurances) without any technological safeguards and illicit behavior detection mechanisms (except, as noted above, traditional security against outsiders).

The reasons are multiple. In consumer markets where competitive pricing is available, current revenue and business models are heavily advertisement-driven and require direct access to client data (emails, documents), access patterns (visited sites, searches, buying patterns), health (see Google Health), and social networks. The data mining market's direct and indirect worth (through advertisements) are valued in the billions and increasing, with no end in sight. Not too long ago, Google spent over $3 billion to acquire DoubleClick, a data mining and web-click tracking company [104, 67].

A second related reason for the use of the trusted model in consumer markets is direct cost, especially in the case of free services. This is illustrated best by the cost-saving behavior of (email) service providers such as Google and Yahoo, that have been (and still are) operating for years with no in-transit confidentiality of traffic, mainly to avoid the significantly increased loads that would ensue due to enabling SSL connections. This led to the development of simple session hi-jacking tools [30]. Ironically, the un-secured traffic is preceded by SSL-secured password authentication.

4.3   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.

Note: As mentioned above, *we are not concerned here with traditional security* suspects such as authentication, certification, inter-client isolation, network security (e.g., strength and overheads of SSL) as well as general physical security. All of these are mainly protecting against *outsiders* and have been addressed extensively in existing research [36, 96, 12].

In the following **we aim to 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. *We focus here on the case of a client outsourcing its sensitive data and computation to a cloud provider for ulterior access by the same client (e.g., corporation etc).* We do not address the more complicated scenario in which the cloud client deploys the cloud as a publisher towards its clients in turn etc. Space and scope constraints prevent us from doing that.

4.3.1 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 [29], we show that, to make (basic, unsecure) outsourcing cost effective, the cost savings (mainly from cheaper CPU cycles) need to outweigh the cloud's distance from clients. This suggests a boundary condition on how compute-intensive an application needs to be before it is worth outsourcing at all:

$$Savings = Cycles \times c_a - Cycles \times c_b - Trans_{a \to b} \geq 0 \Leftrightarrow Cycles \geq \frac{Trans_{a \to b}}{c_a - c_b} \qquad (3)$$

c_x denotes the CPU cycle cost for scenario $x \in \{H, S, M, L\}$, and $Trans_{a \to b}$ is the network cost per unit data. Equation 3 defines what we called the **minimal CPU-intensive requirement** principle [29].

To illustrate, consider the $H \to cloud$ scenario. We now know (Section 2.5) that the cost of reliably transferring 32 bits across application layers can be anywhere 25,000 and 320,000 picocents depending on the nature of the connection and whether connection establishment costs are amortized across multiple sends. The minimal CPU-intensive requirement principle then dictates a compute-intensity of tasks of *at least 3,800 CPU cycles per every 32 bit of outsourced data* before it is worth outsourcing them.

Clarification: For simplicity and without loss of generality, the *minimal CPU-intensive requirement* principle specifically refers to network costs that cannot be amortized over multiple transactions, hence the wording "per every 32 bit of outsourced data". Yet, often applications involve significant amounts of already cloud-hosted data inputs, and in such cases, the principle simply refers to any data that is transfered to/from the cloud. In reality, intra-cloud data access is also coming at a significant cost of comparable magnitude with the client-cloud rates, thus only (minorly) altering the principle's operating points. A discussion of this is out of scope here.

4.3.2 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. For the $H \to cloud$ 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 800 picocents per accessed bit, a cost that is not amortized and two orders of magnitude higher.

From a pure technological cost-centric point of view, it is thus simply not effective to store data remotely. Depending on the application network footprint, **amortized outsourced storage costs can be upwards of**

2+ orders of magnitude higher than local storage for the $H \rightarrow cloud$ 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? We first survey existing work.

Several existing systems encrypt data before storing it on potentially data-curious servers. Blaze's CFS [18], TCFS [26], EFS [88], StegFS [81], and NCryptfs [117] are file systems that encrypt data before writing to stable storage. NCryptfs is implemented as a layered file system [57] and is capable of being used even over network file systems such as NFS. SFS [51] and BestCrypt [62] are device driver level encryption systems. Tripwire [65, 66] is a user level tool that verifies file integrity at scheduled intervals of time. File systems such as I^{3}FS [64], GFS [44], and Checksummed NCryptfs [111] perform online real-time integrity verification. Venti [102] is an archival system that performs integrity assurance on read-only data. Mykletun et al. [91, 92] explore the applicability of signature-aggregation schemes to provide computation- and communication efficient data authentication and integrity of outsourced data.

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 8 picocent/byte on the server and 100 picocents/byte on the client side. In total (client+server), it would cost about 10 to 20 picocents/bit. This is at least 5 times lower than the cost of storing the bit for a year and at least two orders of magnitude lower than the costs incurred by transferring the same bit (at 800+ picocents/bit). Thus, **for outsourced storage, integrity assurance overheads are negligible.**

$$Cost_{HMAC} \approx 10 \, picocents/bit$$

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. In many systems, a chain only includes a set of blocks. The cost of crypto-hash chains can be formulated as

$$Cost_{chashchains} = \frac{C_h \times S_{block} \times N_{block} + C_{sig}}{N_{block}}$$

where C_h denotes the hash cost, S_{block} is the size of the block, N_{block} refers to the number of blocks, and C_{sig} is the cost of a public key signature (or verification).

For an average of twenty 4096 byte blocks[3] secured by a single hash-chain signed using 1024-bit RSA, would yield an amortized cost approximately 450,000 picocents per 4096-byte block (14 picocents/bit) for client read verification and 450 picocents/bit for write/signatures. This is **1-45 times more expensive than the MAC based case**.

4.3.3 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. This limits the

---

[3]Douceur et al. [41], 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 [9, 41].

applicability of outsourcing, as the type of processing primitives available will be reduced dramatically. It becomes important to provide mechanisms for server-side data processing. Untrusted servers should be able to query encrypted data on behalf of clients with confidentiality.

One of the first processing primitives that has been explored allows clients to search directly in remote encrypted data [113, 45, 28, 20, 46, 24, 118, 37, 11, 16]. 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.

For example Song et al. [113] propose a scheme for search on encrypted data in a scenario where a mobile, bandwidth restricted user, wishes to store data (e-mail) on an untrusted server. The scheme requires the user to split the data into fix-sized words, encrypt each word separately using a symmetric key protocol and xor the result with a structure containing a pseudo-random bit string and a mapping of the string under a secret key, using a pseudo-random function. The secret key is made dependent on the encrypted word. The resulting data is stored on the server. The structure enables the detection of keyword matches, without revealing the server the keyword or the contents of the stored data. The drawbacks of the scheme are fix-sized words, the complexity of the encryption and search ($O(n)$ where n is the number of words) and the impossibility of verifying the correctness of the results returned by the server. The paper also discusses the use of an encrypted index, allowing the whole data to be encrypted as a block. Additionally, this scheme leaks correlations between searched keywords and matched documents are naturally exposed. Moreover, the untrusted server can further perform any combinations of conjunctive keyword searches with keywords previously searched by the client. Upon adding new documents, the server can search within them with tokens revealed by previous keyword searches.

Eu-Jin Goh [45] proposes to associate indexes with the documents stored by the server. More precisely, the index of a document is a Bloom filter [19] containing a codeword for each each unique word in the document. The codeword of a word is derived by twice applying a pseudo-random function to the word. The size of document indexes, as documented in the paper, is proportional to the document size. Chang and Mitzenmacher [28] propose a similar approach, where the index associated with documents consists of a string of bits of length equal to the total number of words used (dictionary size). Two solutions are given, one where the dictionary of words can be stored at the client and one where it has to be stored encrypted at the server.

An interesting version of searching on encrypted data is proposed by Boneh et al. [20], where e-mails encrypted by senders with the public key of the intended receiver are stored on untrusted mail servers. The paper presents protocols allowing receivers to search. In the first protocol, a non-interactive searchable encryption scheme, is based on a variant of the Diffie-Hellman problem and uses bilinear maps on elliptic curves. The second protocol, using only trapdoor permutations, needs a large number of public/private key pairs. Both protocols require the individual encryption of each word.

Golle et al. [46] extend the above problem to conjunctive keyword searches on encrypted data. They propose two solutions. The first solution requires the server to store capabilities for conjunctive queries, with sizes linear in the total number of documents. The paper claims that a majority of the capabilities can be transferred offline to the server, but this only assumes that the client knows beforehand its future conjunctive queries. The second solution requires much less communication between the client and the server, proportional with the number of keywords in the conjunctive search, but doubles the size of the data stored by the server. A severe limitation of these schemes is the requirement of specifying the exact positions where the search matches have to occur, hardly usable in practice.

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 the returned 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:

$$Cost_{search} = c_s \times h_{btree}(\gamma \log B + cycles_r)$$

where c_s is the CPU cycle cost in the server, h_{btree} is the height of the B-tree, γ denotes the number of CPU cycles needed per comparison, and $cycles_r$ refers to CPU cycles in reading a node.

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, $H \rightarrow cloud$ outsourcing would result in CPU-related savings of around 600-1,800 picocents per access.

Yet, these savings do not even cover the transfer costs of the 32 bit keyword from $H \rightarrow cloud$ (which we know to be upwards of 25,000 picocents from Section 2.5), not to mention the cost of transferring the query results back to H. Thus in this case, searching on remote encrypted data is up to 12 orders of magnitude more expensive than local hosting.

When the database size is relatively small, other index constructs such as hashtables can be deployed. For hashtable based indexes the average-case lookup would take a hash operation and roughly a bit more than one access[4].

In this case performing (unsecured) lookup on the cloud would save around 800 picocents per lookup. However, we've seen above that the network cost of simply sending the 32-bit keyword over is at least 25,000 picocents. Thus, similarly to the case of storage outsourcing, **searching on remote outsourced data is not CPU-intensive enough to offset the network overheads and often features 12+ orders of magnitude higher costs**, even in the absence of security!

Cost of Security. Yet, if other considerations mandate its deployment, what are the overheads associated with securing an outsourced oblivious search mechanism? As we already evaluated integrity constructs for outsourced data sets we focus here on confidentiality. Specifically, the server is enabled to perform the search yet should find out as little as possible about it. This would involve the evaluation of a trapdoor or random oracle-based construct for index access. In a typical case, for each new access a client would need to perform an initial construction of a search token (e.g., an encryption or a crypto-hash of portions thereof) which the server then uses to perform the search on the index. The search will involve at least one (e.g., if a hashtable index is used this could be exactly one, or $O(\log n)$ for a B-tree) additional crypto construct evaluation. Finally, the client will need to decrypt the received matching data items. We observe now a security cost which becomes a function of query selectivity (s) and outsourced data size (n). Thus a lower bound cost for searching on encrypted data is given by

$$Cost_{encrypted\_search} \geq O(ns) \times C_{decrypt} + C_{encrypt} + kC_{crypto\_eval}$$

---

[4]As storage costs are 1-2 order of magnitude lower than CPU cycle costs, one might assume space for a well-sized hashtable in the hope of reaping outsourcing benefits.

where $C_{decrypt}, C_{encrypt}, C_{crypto\_eval}$ denote the cost of a decryption, an encryption and a crypto construct evaluation respectively, k is the number of evaluations. The encryption and crypto evaluation, for example could be a mix of public key constructs plus underlying symmetric ciphers [16] yielding at least 10^5 pico-cents per lookup. This alone is 1.5 times more than the minimum search token transfer costs, thus further enforcing the case against secure searching on outsourced data! Finally, in some mechanisms, clients need to (decrypt and) prune unwanted results, a by-product of preserving privacy.

4.3.4 Oblivious Data Access

Encryption provides important privacy guarantees at low cost. However, it constitutes only a first step, as significant amounts of information is still leaked through access pattern to encrypted data. For example, by correlating access frequencies to external events and knowledge, an access-curious storage provider can determine with increasing accuracy the semantics of specific stored records. This is often un-acceptable, especially in scenarios involving high-risk or costly and sensitive information.

Private Information Retrieval, (PIR) was first proposed as a theoretical primitive for accessing out-sourced data, while preventing the data stores to learn anything about the client's access patterns [31]. PIR protocols' main goal is to hide access patterns (but not the actual data content). In initial results, Chor et al. [32] proved that in an information theoretic setting in which queries do not reveal any information at all about the accessed data items, any solution requires $\Omega(n)$ bits of communication. To avoid this over-head, they show that if multiple non-communicating databases hold replicated databases, PIR schemes exist which require only sub-linear communication overheads. Later, Cachin et al. [25] showed how to construct a single-database computational PIR scheme for which the communication complexity is poly-logarithmic in the size of the database. Chang [27] introduced a single database computational PIR scheme for which the server side communication complexity is $O(log(n))$. The scheme relies on the Paillier crypto-system [97]. Numerous other results are surveyed by Gasarch [43].

Sion et al. have shown [110] that deployment of existing non-trivial single server PIR protocols on real hardware of the recent past would have been orders of magnitude more time-consuming than trivially transferring the entire database. Yet, despite the fact that efficiency-wise PIR is impractical, does the same hold when considering costs, e.g., in the $H \rightarrow cloud$ scenario? It may seem that expensive networking, coupled decreased cloud cycle costs favor the case for non-trivial PIR.

For consistency we start our reasoning with the protocol [69] considered also by the authors of [110]. The hardness problem of choice is the quadratic residuosity (QR) assumption and its equivalent, factoring. The n bits of the database are organized logically at the server as a bi-dimensional matrix M of size $\sqrt{n} \times \sqrt{n}$. To retrieve bit $M(x, y)$ privately, the client: (i) chooses two random prime numbers p and q of similar bit length, and sends $N = pq$ to the server; (ii) generates $\sqrt{n}$ numbers $s_1, s_2, \ldots, s_{\sqrt{n}}$, such that s_x is a quadratic non-residue (QNR) and the rest are quadratic residues (QR) in $\mathbb{Z}_N^*$; (iii) sends $s_1, s_2, \ldots, s_{\sqrt{n}}$ to the server. For each "column" $j \in (1, \sqrt{n})$ in the $\sqrt{n} \times \sqrt{n}$ matrix, the server: (iv) computes the product $r_j = \prod_{0 < i < \sqrt{n}} q_{ij}$ where $q_{ij} = s_i^2$ if $M(i, j) = 1$ and $q_{ij} = s_i$ otherwise. (v) sends $r_1, \ldots, r_{\sqrt{n}}$ to the client. The client then checks if r_y is a QR in $\mathbb{Z}_N^*$ which implies $M(x, y) = 1$, else $M(x, y) = 0$.

Without loss of generality, let us consider 1-bit data items. The baseline protocol to compare against would involve simply transferring the n data items to the client, costing around $n \times 800$ picocents. For consistency with the results in [110] we also consider 1024 bit values here while acknowledging their lack of security. In this case, it is straightforward to show that for $n < 1MBit$ the protocol transfers more than n bits, being thus trivially less efficient and more costly that the baseline. Then let $n \geq 10^6$ (at least). The total cPIR costs include at the server $n/2$ 1024-bit modular multiplications and $\sqrt{n}$ 1024-bit message transfers, and, at the client $\sqrt{n} - 1$ 1024-bit modular multiplications, two 512-bit modular exponentiations and the sending of $\sqrt{n}$ 1024-bit values to the server.

$$C_{cPIR} = C_{mult}^s \times \frac{n}{2} + C_{mult}^c \times (\sqrt{n} - 1) + 2C_{exp} + 2\sqrt{n} \times Trans_{c \leftrightarrow s}$$

C_{mult}^s and C_{mult}^c denote the modular multiplication cost in the server and in the client. This yields a minimum cPIR cost of about 89 **milli**cents (increasing mainly linearly with n) dominated 95.70% by server-side modular arithmetic. For the same setting, the trivial baseline yields *two orders of magnitude lower costs*, at around 0.8 **milli**cents (also increasing linearly with n)! Moreover, as the considered protocol is considered the most computation efficient amongst existing single-server PIR results [110], and computation is the main reason for this conclusion, this result is likely to hold for all existing single-server PIR techniques. We conclude that **single-server computational PIR is 2+ orders of magnitude more expensive than transferring the entire database to the client.** This is indeed remarkable and despite appearances is **not** the commonly accepted "PIR is expensive" statement, but rather says that the mere idea of PIR does not function given today's tools. In other words **today's cryptography simply lacks power to efficiently support private access outsourcing** (even when compared with simple baselines, such as transferring the entire database to the client).

Above we have discussed [69] for consistency with previous analysis, and illustration purposes. Recently, a more computationally efficient protocol has been proposed, based on a new security assumption, the Hidden Lattice Problem (HLP) [82]. The authors suggest that its communication complexity is significantly higher than that of existing protocols, and the protocol features a query preparation stage in which 2-4 orders of magnitude more data is sent to the server before the query can get executed. Nevertheless, this and other similar mechanisms show promise in getting close to the trivial baseline costs (of transferring the entire data over to the client). Naturally, this is conditional on the successful vetting of the HLP assumption which lies at the foundation of the result.

4.3.5 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 from Section 4.3.1 that (i) the tasks to be outsourced should cost at least 3,800 cycles per 32 bit data word before it makes sense to outsource them *even in the un-secured case*! In other words, outsourcing peanut counting will never be profitable. And we also know that (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 hundreds 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. [52] 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. In [58] the authors explore optimal bucket sizes for certain range queries. One of the main drawbacks of these mechanisms is the fact that they leak information to the server, at a level corresponding to the granularity of the partitioning function. Ultimately, true (computational) confidentiality cannot be achieved by partitioning schemes alone.

Recently, Ge et al. [114] discuss executing aggregation queries with confidentiality on an untrusted server. Unfortunately, due to the use of extremely expensive homomorphisms (Paillier [97, 98]) this scheme leads to impractical processing times for any reasonably security parameter settings (e.g., for 1024 bit fields, 12+ days *per query* are required). Current homomorphisms are simply not fast enough to be usable here.

Other researchers have explored the issue of **correctness** in settings with potentially malicious servers. Informally, a query mechanism is *correct* if the server is bound to the sequence of update requests performed

by the client. Either the server responds correctly to a query, or its malicious behavior is immediately detected by the client. In applied settings, *correctness* can be decomposed into two protocol properties, namely *data integrity* and *query completeness* – ensuring queries are executed against their entire target data sets and results are not 'truncated". In a publisher-subscriber model, Devanbu et al. deployed Merkle trees to authenticate data published at a third party's site [39], and then explored a general model for authenticating data structures [79, 80]. Hard-to-forge verification objects are provided by publishers to prove the authenticity and provenance of query results. In [92, 91], mechanisms for efficient integrity and origin authentication for selection predicate query results are introduced. Different signature schemes (DSA, RSA, Merkle trees [84] and BGLS [21]) are explored as potential alternatives for data authentication primitives. Similarly, in [77], digital signature and aggregation and chaining mechanisms are deployed to authenticate selection and projection operators. In [100, 75] *verification objects* VO are deployed to authenticate data retrieval in "edge computing" In [99, 93] Merkle tree and cryptographic hashing constructs are deployed to authenticate range query results. [38] proposes an approach for signing XML documents allowing untrusted servers to answer certain types of path and selection queries. In [109], Sion et al. explore query correctness by first considering the query expressiveness problem, where they proposed a novel method for proofs of *actual* query execution in an outsourced database framework for *arbitrary* queries.

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., we will 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 $25,000 \times 2 = 50,000$ picocents, a traversal of an entire database of size 10^5 will generate CPU cycle cost savings enough to offset the network cost. And with a larger database, outsourcing would lead to more cost savings. Thus it seems that for very selective queries outsourcing makes sense.

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, we aim to reason generally and gain an insight into the cost 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 40 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: $Savings = kn \times (c_c - c_s)$, $Cost_{hashtree} = C_{verify} + nsC_{hash} \log n$, $Cost_{trans} = nsBTrans_{s \to c}$, where k is the number of CPU cycles per data item, c_c and c_s are the CPU cycle costs in the client and server, C_{sign_c} and C_{hash_c} is the costs of a signature verification on the client and a hash operation on the server respectively, B denotes the hash tree node size in bits, s is the selectivity factor and $Trans_{s \to c}$ denotes the per-bit network transfer cost from server to client. The outsourcing make economical sense

when $Savings \geq Cost_{hashtree} + Cost_{trans}$. The boundary condition can be rewritten as:

$$s \leq \frac{kn(c_c - c_s) - C_{sign_c}}{n(BTrans_{s \to c} + C_{hash_c} \log n)}$$

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 $s \leq 0.0001$ before outsourcing starts to make economical sense. In a more typical scenario of $s = 0.001$ (queries are returning 0.1% of the tuples in the database), a loss of over 0.2 US cents will be incurred for each outsourced query.

And the above results hold only for the outsourcing-favorable scenario in which only hash trees are deployed. In the case of signature aggregations [93, 91, 92], the overheads include $Costagg = ns(C_{sign_c} + C_{sign_s})$ (client/server side aggregation/verification assuming a similar cost as signatures), and the boundary condition becomes

$$s \leq \frac{kn(c_c - c_s)}{n(BTrans_{s \to c} + (C_{sign_c} + C_{sign_s}))}$$

resulting in $s \leq 10^{-6}$ before outsourcing breaks even in terms of costs when compared with local execution.

5   To Conclude

We explored whether cryptography can be deployed to **secure cloud computing against insiders**.

In the process we evaluated CPU cycles at a price of no less than 0.58 picocents, and saw that a bit cannot be transferred without paying at least 800 picocents, and stored a year without a pocket setback of at least 100 picocents. We estimated common cryptography costs (TDES, AES, MD5, SHA-1, RSA, DSA, and ECDSA) and finally explored outsourcing of data and computation to untrusted clouds.

We concluded that, from a purely technological cost-centric perspective, **cloud computing can be profitable for computation intensive tasks**, specifically, **when the computation cost savings are sufficient to offset their distance from clients**. This happens today for tasks requiring *several thousand CPU cycles per 32-bit transfered input data* (**minimal CPU-intensive requirement** principle).

We then showed that deploying the cloud as a simple remote encrypted file system is extremely unfeasible if considering only core technology costs. Similarly, existing single server cryptographic oblivious data access protocols are not only time-impractical (this has been shown previously) but also (surprisingly) orders of magnitude more dollar expensive than trivial data transfer.

Finally we 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 were able to find 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., 0.00001%) – confirming the minimal CPU-intensive requirement principle on cloud feasibility.

The scope did not permit us to explore the fascinating broader issues at the intersection of technology with business models, risk, behavioral incentives, socio-economics, and advertising markets.

We illustrate in a cloud computing setting, yet we (secretly) hope this type of reasoning will initiate a new current of practical, bottom-line aware designs of security protocols.

References

[1] Amazon Elastic Compute Cloud. Online at http://aws.amazon.com/ec2.
[2] New European Schemes for Signatures Integrity and Encryption. Online at http://www.cryptonessie.org/.
[3] RSA Factoring Challenge. Online at www.rsasecurity.com/rsalabs/challenges/factoring/.

[4] TWIRL and RSA Key Size. Online at `http://www.rsasecurity.com/rsalabs/node.asp?id=2004`.

[5] RSA CryptoBytes, Summer 1995. Online at `ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto1n2.pdf`, 1995.

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

[7] RSA Labs. Online at `http://www.rsasecurity.com/rsalabs`.

[8] Energy Information Administration. "average retail price of electricity to ultimate customers by end-use sector, by state". Online at `http://www.eia.doe.gov/cneaf/electricity/epm/table5_6_a.html`.

[9] 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.

[10] Chloe Albanesius and PC Magazine. Comcast to Cap Data Transfers at 250 GB. Online at `http://www.pcmag.com/article2/0,2817,2329170,00.asp`.

[11] 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.

[12] amazon.com. Amazon Web Services: Overview of Security Processes. Online at `http://s3.amazonaws.com/aws_blog/AWS_Security_Whitepaper_2008_09.pdf`, September 2008.

[13] AMD. Power and cooling in the data centers. Power and cooling in the data centers, Online at `http://enterprise.amd.com/Downloads/34146A_PC_WP_en.pdf`.

[14] APC. Determining total cost of ownership for data center and network room infrastructure. Online at `http://www.apcmedia.com/salestools/CMRP-5T9PQG_R3_EN.pdf`.

[15] Andy Bechtolsheim. Cloud computing. Online at `http://netseminar.stanford.edu/seminars/Cloud.pdf`.

[16] 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.

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

[18] 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.

[19] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[20] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Proceedings of Eurocrypt 2004*, pages 506–522. LNCS 3027, 2004.

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

[22] Antoon Bosselaers. Fast implementations on the pentium. Online at `http://homes.esat.kuleuven.be/~bosselae/fast.html`.

[23] Antoon Bosselaers, Rene Govaerts, and Joos Vandewalle. Comparison of three modular reduction functions. In *CRYPTO '93: Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, pages 175–186, 1994.

[24] R. Brinkman, J. Doumen, and W. Jonker. Using secret sharing for searching in encrypted data. In *Secure Data Management*, 2004.

[25] C. Cachin, S. Micali, and M. Stadler. Private Information Retrieval with Polylogarithmic Communication. In *Proceedings of Eurocrypt*, pages 402–414. Springer-Verlag, 1999.

[26] 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.

[27] Y. Chang. Single-Database Private Information Retrieval with Logarithmic Communication. In *Proceedings of the 9th Australasian Conference on Information Security and Privacy ACISP*. Springer-Verlag, 2004.

[28] Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. Cryptology ePrint Archive, Report 2004/051, 2004. `http://eprint.iacr.org/`.

[29] Y. Chen and R. Sion. Clouds: So fast yet so far. Technical report, Department of Computer Science, Stony Brook University, 2009.

[30] Humphrey Cheung. Point and click gmail hacking at black hat. Online at `http://www.tgdaily.com/content/view/33207/108`.

[31] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 41–50, 1995.

[32] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proceedings of FOCS*. IEEE Computer Society, 1995.

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

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

[35] Personal Communication. "inquiry with optimum lightpath costumer service, november, 2008".

[36] Craig Balding. Cloud Computing and Security Blog. Online at `http://cloudsecurity.org`.

[37] 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.

[38] Premkumar T. Devanbu, Michael Gertz, April Kwong, Chip Martel, G. Nuckolls, and Stuart G. Stubblebine. Flexible authentication of XML documents. In *ACM Conference on Computer and Communications Security*, pages 136–145, 2001.

[39] 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.

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

[41] 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.

[42] John Evdemon. Internet scale computing. Online at `http://blogs.msdn.com/jevdemon/archive/2007/10/24/internet-scale-comput%ing.aspx`.

[43] W. Gasarch. A WebPage on Private Information Retrieval. Online at `http://www.cs.umd.edu/~gasarch/pir/pir.html`.

[44] 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.

[45] E. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/2003/216/`.

[46] P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In *Proceedings of ACNS*, pages 31–45. Springer-Verlag; Lecture Notes in Computer Science 3089, 2004.

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

[48] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. The cost of a cloud: Research problems in data center networks. In *SIGCOM Computer Communications Review*, 2009.

[49] The Green Grid. Green grid metrics: Describing data center power efficiency. Online at `http://www.thegreengrid.org/gg_content/Green_Grid_Metrics_WP.pdf`.

[50] The Clipper Group. Scale-up and scale-out architectures-ibm provides choice with the xseries. Technical report, 2005.

[51] P. C. Gutmann. Secure filesystem (SFS) for DOS/Windows. Online at `www.cs.auckland.ac.nz/~pgut001/sfs/index.html`, 1994.

[52] 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.

[53] James Hamilton. Perspectives Blog. Online at `http://mvdirona.com/jrh/work/`.

[54] James Hamilton. Where does the power go and what to do about it. Online at `http://mvdirona.com/jrh/TalksAndPapers/JamesHamilton_WhereDoesThePowerG%o.pdf`.

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

[56] James Hamilton. On designing and deploying internet-scale services. Technical report, Windows Live Services Platform, Microsoft, 2008.

[57] J. S. Heidemann and G. J. Popek. File system development with stackable layers. *ACM Transactions on Computer Systems*, 12(1):58–89, February 1994.

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

[59] IBM. IBM blade servers. Online at `http://www-03.ibm.com/systems/bladecenter/hardware/servers/`.

[60] Informationweek. Verizon to offer high-speed fiber-optic network services for businesses. Online at `http://www.informationweek.com/news/telecom/showArticle.jhtml?articleID%=193501707`.

[61] Intel. *Intel 64 and IA-32 Architectures Optimization Reference Manual*, 2008.

[62] Jetico, Inc. BestCrypt software home page. `www.jetico.com`, 2002.

[63] Saul Hansell John Markoff. Hiding in plain sight, google seeks more power. Online at `http://www.nytimes.com/2006/06/14/technology/14search.html`.

[64] 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.

[65] G. Kim and E. Spafford. Experiences with Tripwire: Using Integrity Checkers for Intrusion Detection. In *Proceedings of the Usenix System Administration, Networking and Security (SANS III)*, 1994.

[66] G. Kim and E. Spafford. The Design and Implementation of Tripwire: A File System Integrity Checker. In *Proceedings of the 2nd ACM Conference on Computer Commuications and Society (CCS)*, November 1994.

[67] Marshall Kirkpatrik. Do you trust google to resist data mining across services? Online at `http://www.readwriteweb.com/archives/do_you_trust_google_to_resist_data%_mining_across_services.php`.

[68] Cetin Kaya Koc. High-speed rsa implementation, rsa labs, tr 201. Online at `ftp://ftp.rsasecurity.com/pub/pdfs/tr201.pdf`, 1994.

[69] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proceedings of FOCS*. IEEE Computer Society, 1997.

[70] RSA Lab. How fast is the RSA algorithm? Online at `http://www.rsa.com/rsalabs/node.asp?id=2215`.

[71] Los Alamos National Laboratory. Raw failure data. Online at `http://www.lanl.gov/projects/computerscience/data/`.

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

[73] J.-Y. Leu and A.-Y. Wu. A scalable low-complexity digit-serial VLSI architecture for RSA cryptosystem. In *Proceedings of the IEEE Workshop Signal Processing Systems SIPS*, 1999.

[74] Tom Lowry and Business Week. Time Warner Cable Expands Internet Usage Pricing. Online at `http://www.businessweek.com/technology/content/mar2009/tc20090331_72639%7.htm`, March 2009.

[75] 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.

[76] Mads Oesterby Olesen and Henrik Sandmann and Christopher Mosses. Implementing Fast Modular Arithmetic (Course). Online at `http://www.daimi.au.dk/~cmosses/crypt/`, 2002.

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

[78] Om Malik. And now google is making its own 10-gigabit switches. Online at `http://gigaom.com/2007/11/18/google-making-its-own-10gig-switches/`, 2007.

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

[80] 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.

[81] A. D. McDonald and M. G. Kuhn. StegFS: A Steganographic File System for Linux. In *Information Hiding*, pages 462–477, 1999.

[82] Carlos AGUILAR MELCHOR and Philippe GABORIT. A lattice-based computationally-efficient private information retrieval protocol. Cryptology ePrint Archive, Report 2007/446, 2007. `http://eprint.iacr.org/`.

[83] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

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

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

[86] Cade Metz. Google demanding intel's hottest chips? Online at `http://www.theregister.co.uk/2008/10/15/google_and_intel/`.

[87] Microsoft. Windows Azure. Online at `http://www.microsoft.com/windowsazure/`.

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

[89] Rich Miller. Microsoft: Pue of 1.22 for data center containers. Online at `http://www.datacenterknowledge.com/archives/2008/10/20/microsoft-pue-of%-122-for-data-center-containers/`.

[90] P. Montgomery. Modular Multiplication without Trial Division. *Mathematics of Computation*, 44(170):519–521, 1985.

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

[92] 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.

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

[94] National Institute of Standards and Technology (NIST). The key management guideline. Online at `http://csrc.nist.gov/groups/ST/toolkit/key_management.html`, March 2007.

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

[96] Oreilly.com. Twenty Rules for Amazon Cloud Security By George Reese. Online at `http://broadcast.oreilly.com/2008/11/20-rules-for-amazon-cloud-security%.html`, November 2008.

[97] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of EuroCrypt*, 1999.

[98] Pascal Paillier. A trapdoor permutation equivalent to factoring. In *PKC '99: Proceedings of the Second International Workshop on Practice and Theory in Public Key Cryptography*, pages 219–222, London, UK, 1999. Springer-Verlag.

[99] 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.

[100] 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.

[101] U.S. Environmental Protection Agency ENERGY STAR Program. Report to congress on server and data center energy efficiency public law 109-431, 2007.

[102] S. Quinlan and S. Dorward. Venti: a new approach to archival storage. In *Proceedings of the First USENIX Conference on File and Storage Technologies (FAST 2002)*, pages 89–101, Monterey, CA, January 2002. USENIX Association.

[103] Rainer Bluemel and Ralf Laue and Sorin A. Huss. A highly efficient modular Multiplication Algorithm for Finite Field Arithmetic in GF(P). In *Proceedings of ECRYPT Workshop, Cryptographic Advances in Secure Hardware*, 2005.

[104] Marketplace Morning Report. Ok'd deal makes 'data-mining colossus'. Online at `http://marketplace.publicradio.org/display/web/2007/12/20/google_double%click`.

[105] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, pages 199–212, New York, NY, USA, 2009. ACM.

[106] Bianca Schroeder and Garth A. Gibson. A large-scale study of failures in high-performance computing systems. In *DSN '06: Proceedings of the International Conference on Dependable Systems and Networks*, pages 249–258, Washington, DC, USA, 2006. IEEE Computer Society.

[107] Bianca Schroeder and Garth A. Gibson. Disk failures in the real world: what does an mttf of 1,000,000 hours mean to you? In *FAST '07: Proceedings of the 5th USENIX conference on File and Storage Technologies*, Berkeley, CA, USA, 2007. USENIX Association.

[108] Bruce Schulman. Finding the right processing architecture for aes encryption. Online at `http://www.eetimes.com/story/OEG20030618S0012`.

[109] Radu Sion. Query execution assurance for outsourced databases. In *Proceedings of the Very Large Databases Conference VLDB*, 2005.

[110] Radu Sion and Bogdan Carbunar. On the Computational Practicality of Private Information Retrieval. In *Proceedings of the Network and Distributed Systems Security Symposium*, 2007. Stony Brook Network Security and Applied Cryptography Lab Tech Report 2006-06.

[111] 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`.

[112] S.M.Hong, S.Y.Oh, and H.S.Yoon. New modular multiplication algorithms for fast modular exponentiation. In *Advances in Cryptology-EUROCRYPT*, 1996.

[113] D. Xiaodong Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P 2000)*. IEEE Computer Society, 2000.

[114] 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.

[115] Verizon. High speed internet plans. Online at `http://www22.verizon.com/Residential/HighSpeedInternet/Plans/Plans.htm`.

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

[117] C. P. Wright, M. Martino, and E. Zadok. NCryptfs: A Secure and Convenient Cryptographic File System. In *Proceedings of the Annual USENIX Technical Conference*, pages 197–210, San Antonio, TX, June 2003. USENIX Association.

[118] Zhiqiang Yang and Sheng Zhong and Rebecca N. Wright. Privacy-Preserving Queries on Encrypted Data. In *ESORICS*, 2006.