Trustworthy Distributed Systems through Integrity-Reporting

Jun Ho Huh and Andrew Martin

Oxford University Computing Laboratory

Parks Road, Oxford, OX1 3QD, UK

{jun.huh, andrew.martin}@kellogg.ox.ac.uk

Abstract With the growing influence of e-Science, substantial quantities of research are being facilitated, recorded, and reported by means of distributed computing. As a result, the scope for malicious intervention continues to grow, and so do the rewards available to those able to steal the models and data that have significant commercial value. Researchers are often reluctant to exploit the full benefits of distributed computing because they fear the compromise of their sensitive data or the uncertainty of the returned results. In this chapter, we propose two types of trustworthy distributed systems – one suitable for a computational system and the other for a distributed data system. Central to these systems is the novel idea of c*onfiguration resolver*, which, in both designs, is responsible for filtering trustworthy hosts and ensuring that jobs are dispatched to those considered trustworthy. Furthermore, the *blind analysis server* enables statistical analyses to be performed on sensitive raw data – collected from multiple sites – without disclosing it to anyone.
Keywords: trusted computing, trustworthy distributed systems, configuration verification server, blind data analysis, trustworthy grid

1 Introduction

In recent years, distributed systems have enjoyed a huge burst of popularity, most chiefly in the commodity computing model described as ‘Cloud Computing’. The term applies to a broad range of systems architectures, often categorized under the headings of Software/Platform/Infrastructure as a Service – and perhaps also subsumes one of its progenitors ‘Grid computing’.

A clear driver for such adoption is the benefit of using shared resources: load can be balanced across large numbers of hosts, peaks easily accommodated, and massive initiatives run as background tasks on systems which would otherwise be idle. To these is now added a ‘green’ agenda – that by taking advantage of economies of scale and careful location of data centers, the ‘carbon footprint’ of any given computational task can be minimized.

Demand for such models of computing also arises from unprecedented volumes of data for processing. Evidence-based science, and basic observations ‘born digital’ are driving this, as is closer scrutiny of results, giving rise to strong requirements on provenance, accurate data acquisition, and integrity in processing. This pattern is repeated across government, business, social networking, and more. Some applications give rise to more esoteric requirements, such as ‘digital rights management’ for all kinds of data, and elaborate rules for combining policies where several such managed data sources are merged – perhaps within a ‘blind analysis’ combination where neither data owner is permitted to see the other’s raw data, but both are interested in the results.

All of this takes place against a background of ever-increasing sophistication in attacks. Precisely because there are so many high-value digital assets being brought into existence, there are also many attackers seeking to subvert them. There is much money to be made from subverting online business and much political capital to be gained from manipulating certain scientific results. Even as processes are more transparent, there is ever more motivation to ‘tweak’ results just slightly in order to achieve a better result.

Our focus in this chapter is to explore how technologies allied to *Trusted Computing* can help to address these challenges, through the use of soundly-based practically-feasible architectures and designs. In the following section, we survey some motivating examples, and describe our perspective on the special challenges they represent: these requirements are summarized in Section 3. In Section 4, we present a sketch of the existing technologies for trust and virtualization which will form the basis of our architectures. This puts us in a position to discuss in Section 5 the ways that other authors have proposed for using those technologies in a grid context – and the measure of consensus achieved in such accounts so far. Section 6 surveys the remaining gaps.

In Section 7 we present our own substantive contribution: designs for trustworthy distributed systems, using in particular our concept of a *configuration resolver* to broker data about acceptable trustworthy system configurations. After this, we evaluate those designs against the requirements from Section 3. Section 9 is a different form of evaluation: a consideration of how these technologies could be deployed within the UK’s *National Grid Service*. Our conclusions and discussion of future work are at Section 10.

2 Motivating Examples

The domains of *e-Science* and *e-Health* provide us with good examples for motivation: they are somewhat more accessible than many other domains where confidentiality (of designs) is of greater concern. The following examples serve to illustrate the common security problems of sharing computational resources or aggregating distributed data within a virtual organisation.

2.1 **climate*prediction*.net** ***and Condor Grids***

The first example application arises with the climate*prediction*.net project (climateprediction.net 2009), which functions by distributing a high quality climate model to thousands of participants around the world. It stands (or falls) on its ability to ensure the accuracy of the climate prediction methods and collected data. As a politically-charged field it could become a target for moderately sophisticated attackers to subvert the results.

This project highlights a common dual pair of problems:

1. From the participant’s (resource provider) perspective, the untrusted code runs on their trusted system; they need to be convinced that the code is not malicious, and the middleware used by the code (if any) are trustworthy,
2. From the scientist's perspective, their trusted job is executed in an untrusted host without any assurance of the running environment; this host might return arbitrary or fabricated results never having run the original code, or steal their sensitive models and data.

Such *volunteer computing* models represent one of the most challenging possible environments for computational integrity – but also one of the greatest possible assets for their task. At its peak, for example, climate*prediction*.net was by some measure the largest computational climate model in the world. Some tasks are amenable to duplication – so that if participants return fake results, these are discovered by comparison with those from elsewhere – but this is plainly a waste of computational resources.

Similar threats undermine the security of a Condor system (Thain, Tannenbaum and Livny 2005) which allows relatively smaller jobs to be distributed in a Campus Grid setting (Wallom and Trefethen 2006). To mitigate the second problem it provides a digital certificate infrastructure for the scientist to identify resource/service providers and vice versa. Without robust mechanisms to safeguard the keys from theft, however, this solution offers only a modest improvement over legacy architectures. Moreover, rogue administrators might replace the compute nodes with malicious ones, tamper with them, or subvert their security configurations to steal sensitive data and/or return fabricated results.

Even grids or clouds constructed from nodes within a managed data-centre are subject to the same concerns: the system manager may be subverted by a competitor (through straightforward bribery, or through some sophisticated Trojan).

2.2 Healthcare Grids

In ‘e-Health’, it is not hard to imagine instances where the clinical data is highly sensitive, and only the processed subsets may be released; nor is it hard to imagine scenarios where reconciliation of data from different sources is needed, but neither clinic trusts the other to see the raw data. Such data cannot normally be made available outside the healthcare trust where it is collected, except under strict ethics committee guidance, generally involving *anonymisation* of records before release (Power, et al. 2002).

Nevertheless, anonymisation reduces the amount of information available, precision of estimates and flexibility of analysis; and as a result, bias can be introduced (Duncan and Pearson 1991). For example[1], a researcher might be looking at association between age, diet and progression of colon cancer, and is aware that the risk immensely increases when one reaches the age of 50. Patient records for the first two attributes would be accessed through a GP practice and the third through a specialist clinic. The National Health Service (NHS) number (Freeman 2001) uniquely identifies a patient across the grid to enable the linking of data. In this scenario a graph plotted with anonymised age – ‘30-35’, ‘35-40’ ... ‘65-70’ – is likely to miss out the important micro-trends all together; in fact, these would be better-observed with datasets closer to age 50. A supporting graph plotted with the *actual age*, say, between 45 and 55, would show these trends more clearly and improve the quality of the results.

Moreover, this distributed query would require a concrete identifier, such as the NHS number, to join patient records collected from the GP and specialist clinic. In reality, however, it is unlikely that either would give out such potential identifiable information without the necessary confidentiality guarantees. Hashing NHS number can provide some assurance but it would still be vulnerable to brute force attacks. These problems require a trustworthy application to perform *blind* reconciliation and analysis of the data from mutually-untrusting security domains: the researcher would only see this application running and the end results; the raw data should never be accessible to the researcher.

3 Security Requirements

The motivational examples have in common a likely reliance upon technical measures of security aimed at substantially enhancing protection for job/result integrity and confidentiality. Mindful of the security challenges discussed from

[1] This example has been developed with help from David Power and Mark Slaymaker who are involved in the GIMI project (Simpson, et al. 2005), and Peter Lee who is an intern at the Auckland Hospital.

these examples, this section identifies a set of security requirements for designing trustworthy distributed systems.

1. **Secure Job Submission** Both the integrity and confidentiality of the job secrets should be protected upon job submission. Attackers should not be able to steal or tamper with the job secrets that are being transferred via untrusted midddleware services.
2. **Authorisation Policy Management** When the job arrives at the participant system, the job owner's rights should be evaluated against the authorisation policies. The job should only be processed further if the job owner is authorised to run their queries or codes on the participant system.
3. **Trustworthy Execution Environment** A trustworthy job execution environment should be provided – the jobs should be executed free from any unauthorised interference (e.g. attempts to modify data access query or model code), and the confidentiality of the job secrets should be protected from processes running outside this environment. The user, before submitting the job, should be able to verify that this trustworthy execution environment is guaranteed at the participant system. Similarly, the participant should be ensured that only a verified, integrity protected environment is used in their system for executing the jobs.
4. **Job Isolation** The jobs should be isolated from each other and from the host. This is to prevent rogue jobs from compromising the host, or stealing the secrets and results of other jobs running in the same host. Job isolation should also prevent a malicious host from compromising the job integrity, confidentiality and availability.
5. **Protecting the Results** The integrity and confidentiality of the results should be protected from adversaries, malicious hosts and jobs trying to corrupt/read the results.
6. **Digital Rights Management** In distributed data systems, unauthorised access or modification of sensitive data should be prohibited wherever they may be processed.
7. **Blind Analysis of Data** The raw data should not be disclosed to the end user. Only the processed, anonymised results should be made accessible for analysis.

4 Trusted Computing and Virtualization

The security requirements strongly indicate the need for the user to verify integrity of remote job execution environments, and the data owner to retain control over their data regardless of the system to which it migrates.

Faced with the prospect of modern PCs (and other devices) having so much software that their behaviour is unpredictable and easily subverted, the Trusted

Computing Group (TCG Backgrounder 2006) has developed a series of technologies based around a Trusted Platform Module (TPM) – a hardware chip embedded in the motherboard – which helps to provide two novel capabilities (Grawrock 2006): a cryptographically strong identity and reporting mechanism for the platform, and a means to *measure* the software loaded during the platform's boot process. These include, for example, the BIOS, bootloader, operating system and applications (see Figure 1). Further details of the TPM's functionality is defined in the TPM Main Specification (TPM Main Specification Version 1.2 2003) published by the Trusted Computing Group.

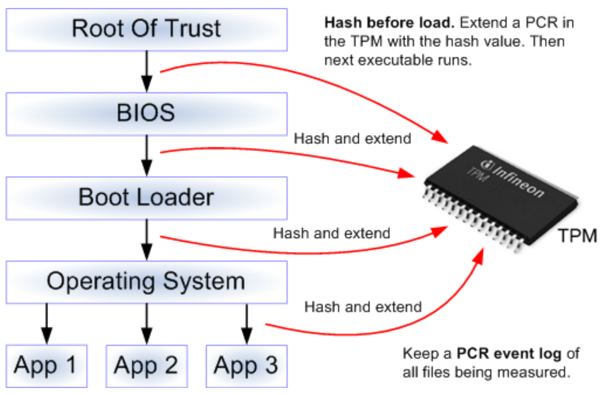
Root Of Trust
Hash before load. Extend a PCR in the TPM with the hash value. Then next executable runs.
BIOS
Hash and extend
Boot Loader
Hash and extend
Infineon
TPM
TPM
Operating System
Hash and extend
App 1
App 2
App 3
Keep a PCR event log of all files being measured.

Figure 1 Authenticated Boot

Measurements are taken by calculating a cryptographic hash of binaries before they are executed. Hashes are stored in Platform Configuration Registers (PCRs) in the TPM. They can only be modified through special TPM ordinals, and the PCRs are never directly written to; rather, measurements can only be *extended* by an entity. This is to ensure that no other entity can just modify or overwrite the measured value. A 20-byte hash of the new measurement is generated based on the PCR's current value concatenated with the new input, and a SHA-1 performed on this concatenated value.

A PCR can be either static or dynamic. A static PCR can reset only when the TPM itself resets – the PCR cannot be reset independently. Static PCRs are normally used to store the measurements. The chapter refers to a static PCR whenever a PCR is mentioned. A dynamic PCR, on the other hand, can be reset independently from the TPM, so long as the process resetting the PCR is under sufficient protection. We refer to such dynamic PCRs as ‘resettable PCRs’.

In a trustworthy system, every executable piece of code in the *authenticated boot* process will be measured and PCRs extended sequentially (*transitive trust*). The notion of transitive trust provides a way for a relying party to trust a large group of entities from a single root of trust: the trust is extended by measuring the next entity before it is loaded, storing the measurement in the TPM, and passing the control to the measured entity (see Figure 1).

Hence, any malicious piece of code (e.g. a rootkit) executed during the boot process will also be recorded and identified. A ‘PCR event log’ is created during boot process and stores all of the measured values (and a description for each) externally to the TPM. These values can be extended in software to validate the contents of the event log. The resulting hash can be compared against the reported, signed PCR value to see if the event log is correct.

4.1 Sealed Storage

Trusted computing provides the means to *seal* (encrypt) data so that it will only successfully decrypt when the platform measurements are in a particular state (Grawrock 2006). The seal process takes external data (information the TPM is going to protect) and a specified PCR value, encrypts the data internally to the TPM using a *storage key*, and creates a sealed data package.

An application – responsible for keeping track of this package – sends the package back to the TPM to recover the data. A nonce, known only to an individual TPM, is also included in the package to ensure that only the TPM responsible for creating the package can unseal it.

The whole purpose of sealing is to prevent any unauthorised attempt to unseal the package. The TPM enforces two restrictions upon decrypting the sealed package:

- ensures that the package is only available on the TPM that created it – the TPM checks whether the nonce included in the package matches the one held internally; and
- compares the current PCR value to the specified PCR value stored in the sealed package – the operation aborts if these values do not match.

The implication is that the external data only becomes available to an application when the correct value (an acceptable platform configuration) is in the specified PCR. Section 7 discusses the use of sealing to protect sensitive data from a malicious/compromised host.

4.2 Remote Attestation

Sealed storage provides a high degree of assurance that the data is only available if the acceptable configuration is present. But how does an external application – that has not performed the seal operation – know that such a configuration is present in a remote platform? Trusted computing provides the means to undertake *remote attestation* (Grawrock 2006): proving to a third party that (in the absence of hardware tampering) a remote platform is in a particular software state.

Remote attestation involves the TPM reporting PCR value(s) that are digitally signed with TPM-generated 'Attestation Identity Keys' (AIKs), and allowing others to validate the signature and the PCR contents. The application wanting to attest its current platform configuration would call the TPM_Quote command specifying a set of PCR values to quote, an AIK to digitally sign the quote, and a nonce to ensure its freshness. The TPM validates the authorisation secret of the AIK, signs the specified PCRs internally with the private half of the AIK, and returns the digitally signed quote.

The external application validates the signature by using the public half of the AIK, and validates the AIK with the AIK credential – a certificate issued by a trusted Certificate Authority (a 'Privacy CA') which states the platform has a valid TPM. The PCR log entries are then compared against a list of 'known-good' values to check if the reported PCRs represent an acceptable configuration. This list is often referred to as an 'application whitelist'.

Attestation can be used on a platform that supports authenticated boot (see Figure 1) to verify that only known pieces of software are running on it. Additions or modifications to any executable will be recorded during the boot process, and noticed when log entries and PCR values are checked. With such mechanisms in place, the external application can, in theory, identify whether a remote platform has been compromised by a malware or not.

4.3 Runtime Attestation Model

Figure 2 gives an overview the Trusted Computing Group's runtime attestation model (TCG 2006). In a trusted platform, the Platform Trust Services (PTS) provide the capability to select hardware and software components to be measured during the authenticated boot process. They are also responsible for computing the measurements of the selected components and the creation of an integrity report containing these measurements. The Verifier checks the incoming integrity reports using the Policy Database, Configuration Management Database (CMDB), and Reference Manifest (RM) Database. These databases hold known-good configurations for platforms. If an attesting platform has an unknown or unexpected configuration, the Verifier informs the Relying Parties not to trust this platform.

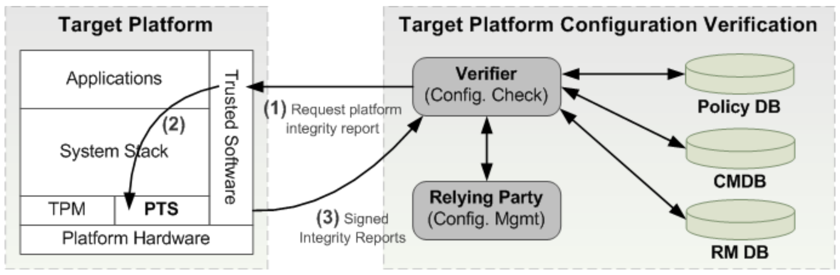
Target Platform
Applications
System Stack
(2)
TPM
PTS
Platform Hardware
Trusted Software
(1) Request platform integrity report
(3) Signed Integrity Reports
Target Platform Configuration Verification
Verifier (Config. Check)
Relying Party (Config. Mgmt)
Policy DB
CMDB
RM DB

Figure 2 TCG's Runtime Attestation Model

The proposed systems in Section 7 are inspired by this runtime attestation model. Each administrative domain is managed by a central Verifier – which we have called the 'configuration resolver' – that checks the configurations of the participant platforms when they first register with the Verifier. Only those verified to be trustworthy become available to the Relying Party (the end users).

4.4 Virtualization

Virtualization is a key technology used in many trusted computing solutions to provide strong isolation for the trusted (TPM-measured) applications – this combination is referred to as 'trusted virtualization'. Virtualization allows a single physical host to share the computing resources between multiple operating systems (Sugerman, Venkitachalam and Lim 2001) (Xen 2005). Each operating system runs in a Virtual Machine (VM) of its own, where it is made to believe that it has dedicated access to the hardware. A virtual machine is also referred to as a 'compartment'.

A thin layer of software called a Virtual Machine Monitor (VMM) operates on top of the hardware to isolate virtual machines and mediate all access to the physical hardware and peripherals. A virtual machine runs on a set of virtual devices that are accessed through virtual device drivers. Typically, a highly privileged 'monitor virtual machine' is created at boot time and serves to manage other virtual machines.

Numerous design efforts have been made to remove avoidable inter-virtual-machine communication mechanisms such as might be exploited to undermine the isolation guarantees. The aim is to make a virtual machine behave in the same way (and have the same properties) as a physically isolated machine. In such designs the virtual machine monitor ensures that all memory is cleared before being reallocated and each virtual machine has its own dedicated memory and disk space. Both Intel and AMD processors now provide hardware support for full, efficient virtualization (Adams and Agesen 2006) (Strongin 2005). With help from these processors, the virtual machine monitor can simulate a complete hardware environment for an unmodified operating system to run and use identical sets of instructions as the host. Hardware virtualization can also speed up the execution of virtual machines by minimising the virtualization overhead.

The majority of current grid middleware solutions, including the Globus Toolkit (Foster, et al. 1998), rely on operating systems' access control mechanisms to manage isolation between user accounts. For example, operating system enforced access control policies prevent malicious software (installed by a third party unknown to the host) from gaining unauthorised access to the jobs running under different user accounts. However, millions of lines of code contained in a mainstream operating system must be trusted to enforce these policies correctly (Sadeghi and Stüble 2004). A single security bug in any one of the privileged components might be enough for an attacker to hijack it, elevate its privileges, and take control of the host and the jobs running inside.

Virtualization, on the other hand, is capable of providing much stronger isolation through the relatively smaller virtual machine monitor and monitor virtual machine (Hohmuth, et al. 2004). A malware (through privilege escalation) would have to compromise both components – which are designed to resist such attacks – in order to break the isolation (Stumpf, et al. 2007). In a trustworthy, virtualized system, these two components (as well as other trusted software) would be measured during authenticated boot and their integrity would be reported through attestation.

A few authors (Figueiredo, Dinda and Fortes 2003) (Keahey, Doering and Foster 2004) have discussed the benefits of isolating the jobs and trusted applications in their own virtual machines:

- job isolation prevents a rogue job from compromising the host or other jobs running in the same host;
- in-memory attacks aimed at modifying the behavior of the trusted applications are made more difficult; and
- privilege-escalation attacks are limited to the isolation boundaries of a virtual machine.

5 An Emergent Consensus View

Great strides have been made in using trusted virtualization to design security architectures that aim to satisfy the requirements identified in Section 3. This section identifies similarities between these trusted virtualization approaches, establishes an 'emergent consensus view' (see Figure 3), and demonstrates its shortcomings in the areas of platform configuration discovery/verification and provision of trustworthy execution environment.

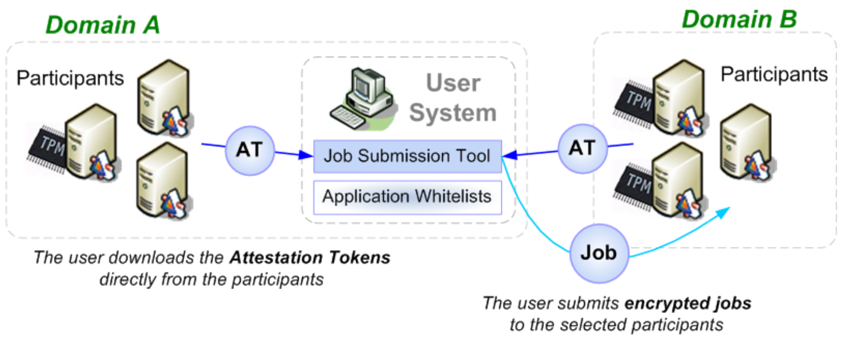
Domain A
Participants
TPM
AT
User System
Job Submission Tool
Application Whitelists
Domain B
Participants
TPM
TPM
AT
Job
The user downloads the **Attestation Tokens** directly from the participants
The user submits **encrypted jobs** to the selected participants

Figure 3 A Consensus View

5.1 Attestation Tokens and Sealed Key Approach

The term ‘attestation token’ is commonly used to describe a participant's credentials (Löhr, Ramasamy and Sadeghi 2007) (Yau, et al. 2008). Typically, it contains the participant’s platform configurations and the public half of a non-migratable TPM key. The private half is bound to the platform’s TPM and PCR values corresponding to its trusted computing base. Information contained in an attestation token should be sufficient for a user to verify the identity and trustworthiness of the platform.

Löhr et al (Löhr, Ramasamy and Sadeghi 2007) combine the Perseus virtualization framework and remote attestation to propose a Trusted Grid Architecture. In the Trusted Grid Architecture, users collect attestation tokens of service providers, and verify their platform configurations using a locally managed whitelist (see Figure 3). Upon job submission, the job secret is encrypted with a service provider’s public key (obtained from their attestation token), guaranteeing that only a securely configured platform will be able to access the private key and decrypt it. If the service provider's trusted computing base has changed, the private key will no longer be accessible to process the job further.

The virtualization layer is extended to include services that support secure job transfer and execution. Grid jobs are transferred across an encrypted, integrity-protected communication channel established with trusted middleware components, and their data is written to disk using a secure storage service. The attestation service uses the attestation token to verify the state of the trusted software layer prior to submitting the job data. The job data is encrypted using the public key (obtained from the token) so that only a securely configured software layer can decrypt it and execute the job.

The Trusted Grid Architecture, however, provides no mechanism for verifying integrity of the job execution virtual machine and the returned results. It also fails to isolate amply the trusted components. Most of its security controls are enforced in a virtual machine that also contains a large amount of untrusted software. For example, the grid management service runs in the same compartment as the storage encryption service. This extra complexity increases the likelihood of vulnerabilities and makes attestation less meaningful. Moreover, the paper does not discuss how the users collect the attestation tokens and how the application whitelists are managed in a distributed environment.

Due to the lack of useful semantic (including security) information that can be conveyed through the standard binary attestation, many (Vejda, et al. 2008) (Mao, Yan and Chen 2006) have suggested the use of 'property-based attestation' (Sadeghi and Stuble, Property-based Attestation for Computing Platforms 2004) to report more security-relevant information. Security-relevant properties of the platform are attested rather than the binary measurements of the software. In consequence, trust decisions made based on integrity-reports are simplified.

5.2 Grid Middleware Isolation

Cooper and Martin (2006) make a strong argument that the complex grid middleware services, which usually have a high likelihood of vulnerabilities, can not be trusted to secure the users' data and credentials. For example, at least five different vulnerabilities had been found in the Globus Toolkit (Foster, et al. 1998) that allow unauthorised users to compromise the middleware (Cooper and Martin 2006).

In the architecture proposed by Cooper and Martin, the middleware stack is isolated in an untrusted compartment of its own and is not relied upon to perform trusted operations. As a result, even if an attacker manages to compromise the middleware, they would not have gained sufficient privileges to undermine the security of a distributed system.

5.3 Job Isolation

The use of virtual machine isolation has been discussed many times as a solution to the 'malicious host problem' (Cooper and Martin 2006) (Wang and Wang 2008) (Vejda, et al. 2008). Typically, a job runs on a separate, dedicated virtual machine, where its code is executed free from unauthorised interference. The job secrets are decrypted inside the virtual machine and protected from rogue virtual machines or host. From a participant's perspective, job isolation could also protect their host from rogue jobs (Pradheep, et al. n.d.).

Terra (Garfinkel, et al. 2003) is a virtualization architecture developed on the VMware virtualization platform (Sugerman, Venkitachalam and Lim 2001). VMware is modified to support encrypted and integrity protected disks. Using their trusted virtual machine monitor, existing applications can either run in a standard virtual machine, or in a 'closed-box' virtual machine that provides the functionality of running on a dedicated closed platform. The trusted virtual machine monitor protects confidentiality and integrity of the contents of the closed-box by intercepting the disk I/O requests and encrypting the disk sectors. The closed-box is strongly isolated from the rest of the platform. Hardware memory protection and secure storage mechanisms intend to protect the contents from rogue administrators.

The authors suggest that Terra could be used to enable a secure grid platform. A closed-box would isolate the job and protect its contents from a malicious host. This closed-box would access its own integrity measurement by performing a system call through the trusted virtual machine monitor. The job owner would use this measurement to verify the integrity of the job execution environment.

5.4 Trusted Execution Environment

Cooper and Martin's architecture (2006) aims to provide a 'trusted execution environment'. A grid job is encrypted and runs on an integrity protected virtual machine where it cannot be accessed from the host platform; the data is safely decrypted inside this virtual machine during execution. Remote attestation is used to verify this environment before dispatching the job.

Their solution works by distributing a job composed of two virtual machines: the first virtual machine runs the job, and the second enforces the trusted execution environment. This second virtual machine, referred to as the 'job security manager', isolates the security layer from the job, and allows the solution to work seamlessly with all legacy virtualization and middleware software.

One potential loophole comes from the fact that they are less concerned about the 'malicious code' problem – untrusted code running on a participant's platform. The job owner specifies the virtual machine instance and its security configurations are not checked before being used. The system relies on virtualization alone to isolate rogue jobs from the host.

The type of attacks a malicious virtual machine can perform would be restricted if virtualization offers complete isolation but, no existing solution guarantees this property right now (although, it is the objective of many). For example, in Xen, each virtual machine has two rings, one for sending requests and one for receiving responses, and these form the inter-virtual-machine communication mechanism (Barham, et al. 2003). A rogue job could potentially hijack a privileged process and manipulate this communication channel to perform buffer overflow attacks on the privileged virtual machine.

6 Missing Pieces and Potential Solutions

Having described the consensus view, this section identifies the missing components and suggests potential solutions.

The first missing piece is a platform configuration discovery service. In the Trusted Grid Architecture (Löhr, Ramasamy and Sadeghi 2007), the users are expected to collect the attestation tokens directly from the participants. How the users would actually manage this process, however, is not considered in depth. Generally, it is assumed that a central service is already available for the users to discover participants' platform configurations. In consequence, various security and management issues associated with developing a 'match-making service' as such are often overlooked.

In the consensus view, the burden of performing attestation and managing the application whitelists rests with the users. This seems unrealistic in large-scale distributed systems, however, since the whitelist entries will be modified and updated constantly. An average user will not have sufficient resources to cope with these changes. Referring back to the Trusted Computing Group's runtime attestation model (see Section 4.3), the 'Configuration Verifier' is missing in the consensus view. Some suggest passing on the problem to a trusted third party (Vejda, et al. 2008) (Nagarajan, Varadharajan and Hitchens 2007), but further elaboration is needed.

Something like the Configuration Verifier could be configured to centrally manage the application whitelists and perform configuration verification (attestation) on behalf of the users. It would be responsible for keeping up-to-date whitelists through various vulnerability tests and data collected. This type of service is described by the Trusted Computing Group as an aggregation service and has been suggested in a number of projects (Yau, et al. 2008) (Wang and Wang 2008). For instance Sailer et al (Sailer, et al. 2004) encourage the remote users to keep their systems at an acceptable patch level using a package management database. This database gets updated whenever a new patch is released, so that the new versions are added to the whitelist and the old versions are removed.

From the participants' perspective, an 'integrity-report based job verification' mechanism is also missing. Only the users (job owners) are capable of verifying the participants' platform configurations, and not vice versa. The participant usually relies on a basic digital certificate to identify the users and authenticate the job virtual machines. This provides no assurance for the security state of the job virtual machines.

In the context of data grids – where the jobs might try to access sensitive data – the data owner should have full control over the software used for executing the query and protecting the accessed data. Virtual machine isolation can only prevent other rogue virtual machines from stealing the accessed data. If the job virtual machine itself is malicious and tries to run malicious queries on the database, then isolation will not be sufficient.

Basic PKI-based encryption and digital signatures are often used to protect the data once they leave the data owner's platform (Luna, et al. 2008). However, considering the number of connected nodes and the security threats associated with each, these security measures alone cannot provide the necessary confidentiality, privacy and integrity guarantees. A more reliable Digital Rights Management system is needed to allow the data owner to maintain full control over their data. The data access policies and privacy policies need to be consistently enforced throughout the distributed system. The end user should only be able to access the processed, anonymised results which are just sufficient to perform the requested analysis.

Meanwhile, Australia's Commonwealth Scientific and Industrial Research Organisation (CSIRO) has developed the Privacy-Preserving Analytics (PPA) software for analysing sensitive healthcare data without compromising its privacy (O'Keefe 2008). Privacy-Preserving Analytics allows analysis of original raw data but modifies output delivered to the researcher to ensure that no individual unit record is disclosed, or can be deduced from the output. This is achieved by shielding any directly identifying information and deductive values that can be matched to an external database. Duncan and Pearson (1991) discuss the benefits of being able to access the raw data:

- no information is lost through anonymising data prior to release and there is no need for special techniques to analyse perturbed data;
- it is relatively easier to anonymise the output than modifying a dataset when it is not known which analyses will be performed; and
- clinical decisions will be based on more reliable information and treatments can be more tailored to individuals with the likelihood of problems.

Privacy-Preserving Analytics (or any other secure analysis tools available), combined with remote attestation, could provide the necessary confidentiality and privacy guarantees for the data owner to freely share their raw data in the virtual organisation. For instance, attestation could verify that a trustworthy Privacy-Preserving Analytics server is responsible for performing data reconciliation and anonymising the output before releasing the results to the researcher.

7 Trustworthy Distributed Systems

We propose two types of distributed systems that aim to satisfy the security requirements (see Section 3), and bridge the gaps identified in the consensus view (see above). A configuration management server called the 'configuration resolver' plays a central role in both systems, maintaining an up-to-date directory of trustworthy participants and handling the job distribution process.

Section 7.1 describes how the configuration resolver manages configuration verification and job distribution processes. Based on the new security primitives that make use of the resolver, Sections 7.2 and 7.3 describe a computational system and a distributed data system that are trustworthy.

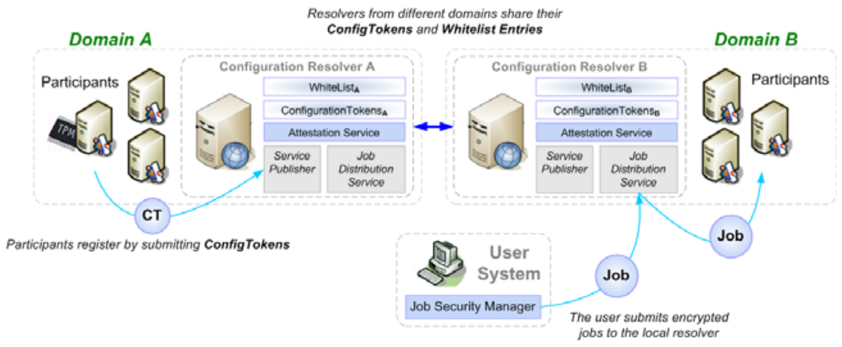

Figure 4 Consensus View with the Configuration Resolver

7.1 The Configuration Resolver

Building on the consensus view of the trusted distributed systems, a central configuration management server is added to each administrative domain to manage the trustworthy participants' platform configurations and a whitelist of locally acceptable platform configurations (see Figure 4). This configuration management server is referred to as the 'configuration resolver'. To become part of the trusted domain, a participant registers with the local configuration resolver by submitting its Configuration Token (CT). The token content is shown below.

$$CT = (\text{PCR Log} , AIK , \{cred(AIK)\}_{CA} , P_K , \{cred(P_K)\}_{AIK} , \{Description\}_{SK})$$

This token includes the Attestation Identity Key (AIK) and an AIK credential issued by the Certificate Authority ($\{cred(AIK)\}_{CA}$). A public key credential, signed by this AIK, is also included to state that the private half has been sealed to *two* PCR values which correspond to (1) a trustworthy authenticated boot process, and (2) per-job virtual machine image files (see Figure 5). The PCR Log contains the full description of the authenticated boot process and the virtual machine image files. In addition, a service Description is included, signed by the private half of the sealed public key, demonstrating that the users should use this public key when submitting jobs to this participant.

The resolver verifies the trustworthiness of the platform by comparing the PCR Log with the whitelist. If the platform is trustworthy, its configuration token is added to the resolver's token repository, ensuring that only the trustworthy participants are ever advertised through the resolver. There, the burden of verifying the integrity reports rests on the resolver.

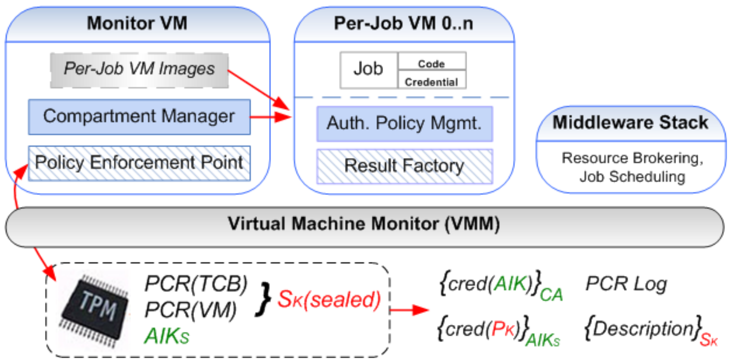

Figure 5 Participants' Trusted Computing Base

As a minimum, the authenticated boot process will measure the BIOS, bootloader, virtual machine monitor, and privileged monitor virtual machine. Therefore, the first PCR value is sufficient to state that the platform is running in a virtualized environment and its monitor virtual machine is securely managing the per-job virtual machines (see Figure 5). Additionally, the second PCR value guarantees the exact software and security configurations of a per-job virtual machine (job execution environment). This second value is stored in a *resettable* PCR (see Section 4) since the virtual machine image files are re-measured and verified at runtime. These security properties allow the user to have strong confidence in the correctness of the data or computational results returned from this platform.

Note, in contrast to the reviewed approaches (see Section 6), the participant controls the virtual machine instances that are allowed to be used in their platform. This is responsible for meeting Requirement 3 (see Section 3). However, this also restricts the number of software environments that the user can choose from, and will affect the overall usability of job submission.

To improve usability and flexibility, the resolver allows a participant to submit multiple configuration tokens (for a same platform), all representing the same authenticated boot process but each sealed to a different per-job virtual machine image. Such tokens could be used to offer multiple services by configuring each software environment to provide a different service, or to offer multiple software environments for a single service, providing the user with more options.

The configuration resolver performs a range of security and platform configuration management functions through the following services (see Figure 4):

- an internal ‘attestation service’ is responsible for performing all attestation related functions to ensure that only trustworthy participants register with the resolver;
- an external ‘service publisher’ provides the necessary APIs for the participants to register and advertise their services through the resolver. It makes use of the attestation service;
- the users submit jobs through an external ‘job distribution service’, which selects the most suitable sites by looking up the service Descriptions and dispatches the jobs to them; and
- an external ‘whitelist manager’ allows the domain administrators to efficiently update the whitelist entries.

Each participant becomes a member of the resolver’s WS-ServiceGroup (Maguire and Snelling 2004) and has a ServiceGroupEntry that is associated with them. An entry contains service information by which the participant’s registration with the resolver is advertised. The configuration tokens are categorised and selected according to the type of services they advertise. It is assumed that there is a public key infrastructure available to verify the participant’s identity.

7.2 Computational Distributed System

In an idealised computational distributed system, the user would not care about where their job travels to as long as their sensitive data and results are protected. It would therefore make sense for the resolver to perform trusted operations like selecting suitable sites and dispatching jobs on behalf of the Job Owner (JO). The resolver’s TPM is used to measure the configurations of its external and internal services, and generate an attestation token (AT(CR)) to attest its security state to the users:

$$AT(CR) = (\text{ PCR Log }, AIK(CR) , \{cred(AIK)\}_{CA} , P_K(CR) , \{cred(P_K)\}_{AIK})$$

Much like the configuration token described previously, the resolver's public key credential ($\{cred(P_K)\}_{AIK}$) identifies the corresponding private key as being sealed to its trustworthy state. The PCR Log describes the fundamental software stack and the services that have been measured during the resolver’s authenticated boot process.

In the user system, all the job security functions are enforced by the ‘job security manager’ virtual machine (see Figure 6): it is designed to perform a small number of simple security operations to minimise the attack surface. Attestation of the job security manager, monitor virtual machine and virtual machine monitor is sufficient to be assured that the job security functions have not been compromised – these components form the trusted computing base of the user system. Upon installation of this architecture, the user system will be capable of securely submitting jobs to the resolver and verifying the returned results.

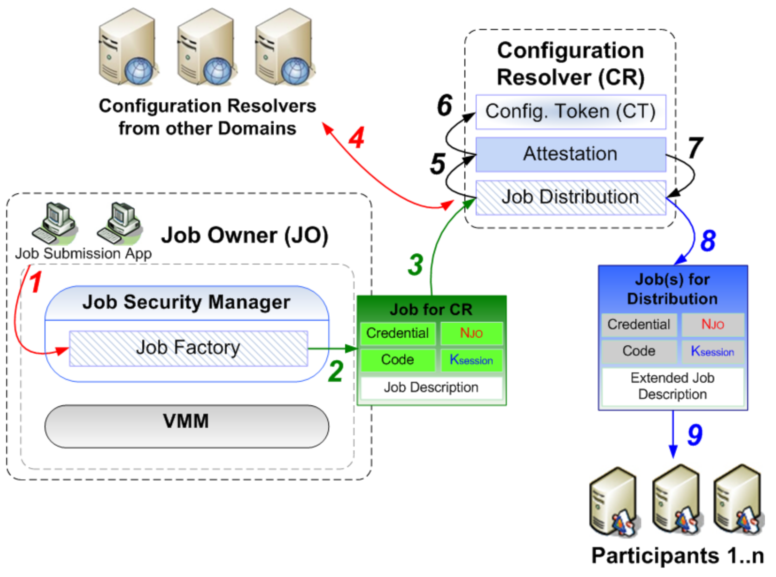

Figure 6 Creation and Distribution of an Encrypted Job

7.2.1 Creation and Distribution of an Encrypted Job

All end user interactions are made via the external ‘job factory’. It provides the minimal interface (APIs) necessary for development of a job submission application. Such an application should be designed to allow the user to specify the job description (requirements), the credentials, and the code to be executed.

Imagine that a scientist (the job owner in this scenario) is carrying out an experiment that aims to predict the future climate state. The scientist submits the prediction model code through their job submission application and specifies the job description (**1**, Figure 6). The job factory creates a secure job containing the following attributes (**2**, Figure 6):

$$\text{Job} = (\ \{\text{Credential, Code}, K_{session}, N_{JO}\}_{PK(CR)}\ ,\ \text{Job Description}\ )$$

A symmetric session key ($K_{session}$) is included as part of the job secret; it will be used by the participant to encrypt the generated results. This session key is sealed to the PCR corresponding to the user system's trusted computing base; this prevents a compromised job security manager from decrypting the returned results. N_{JO} represents the job owner's nonce.

Before encrypting the job secret, the trustworthiness of the resolver is verified by comparing the PCR Log (obtained from the resolver's attestation token, AT(CR)) against the locally managed whitelist of known good resolver configurations. If the resolver is trustworthy, the job secret – containing the user Credential, Code, session key and nonce – is encrypted with the resolver's public key. This sealed key approach ensures that the secret is only accessible by a securely configured resolver. The job is then submitted to the local resolver's job distribution service (**3**, Figure 6).

When the job arrives, the distribution service first attempts to decrypt the job secret with the sealed private key. Then it communicates with the local resource broker – that is linked to the VO-level central information service – to discover resource availability. It requests configuration tokens – for those with available resources – from the resolvers managing other administrative domains (**4**, Figure 6). This request contains the job requirements, specifying the required software and hardware capabilities. Such information is obtained from the Job Description.

The resolvers from other domains select the tokens that match the job requirements and return them to the local resolver. The local resolver uses its internal attestation service to iterate through each token and verifies the integrity-report by comparing the PCR values against the local whitelist (**5**, **6**, **7**, Figure 6). Only those with acceptable configurations (for running the climate prediction model code) are selected and merged with the locally filtered tokens. Finally, the most suitable participant is selected from this merged list to run the job.

The job is recreated for the selected participant: during this process, the job secret is encrypted using the target participant's public key, and the Job Description is extended with the host address (**8**, Figure 6). These jobs are dispatched to the job-manager of the selected participant that reads the unencrypted Job Description and schedules the job. On its turn, the job is forwarded to the participant's policy enforcement point (**9**, Figure 6). Note, middleware services such as the job-manager can only read the extended Job Description for scheduling the jobs.

7.2.2 Operations of the Trustworthy Execution Environment

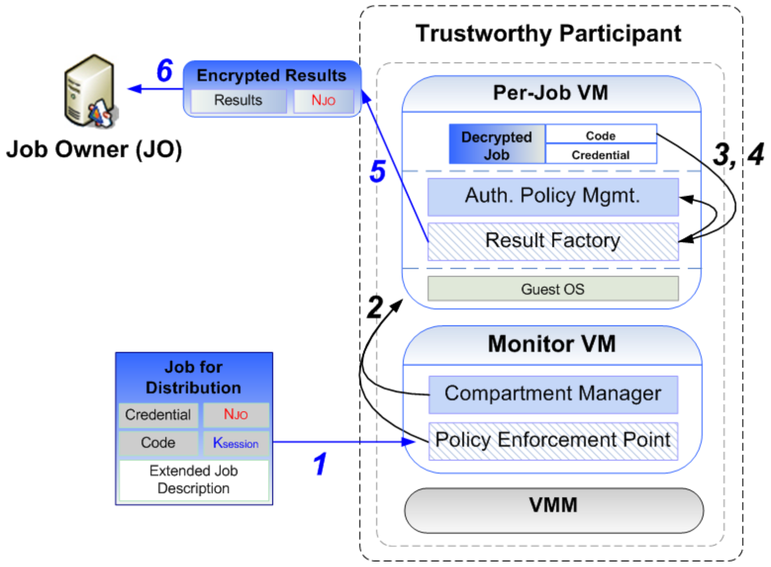

Figure 7 Operations of a Per-Job Virtual Machine

Figure 7 demonstrates how the job gets processed at the participant system. Any security processing required before becoming ready to be deployed in a per-job virtual machine is done through the policy enforcement point. It first measures the selected per-job virtual machine image (and the configuration files), and resets the resettable PCR with the new value. Typically, this image consists of a security patched operating system and trustworthy middleware stack – the ‘authorisation policy management service’ and the ‘result factory’ (see Figure 7).

In order to decrypt the job secret, the policy enforcement point attempts to unseal the private key sealed to the participant’s trusted computing base and the virtual machine image. The private key will only be accessible if the platform is still running with trustworthy configurations *and* the image files have not been modified. This is intended to guarantee that only an integrity protected virtual machine has access to the job secret.

If these security checks are passed, the ‘compartment manager’ allocates the requested size of memory, CPU time and speed (specified in the Job Description), launches a virtual machine from the verified image, and deploys the decrypted job (**2**, Figure 7). Inside this virtual machine, the policy management service decides whether the scientist is authorised to run their prediction model in the participant platform. If the conditions are satisfied, the model is executed to simulate a probabilistic climate forecast (**3**, **4,** Figure 7). The result factory generates a secure message containing the simulation Results (**5**, Figure 7):

$$R = \{Results, N_{JO}\}_{K_{session}}$$

The job owner’s nonce (N_{JO}) is sufficient to verify that the results have been generated from an integrity protected virtual machine and unmodified code has been executed. The entire message is encrypted with the job owner’s symmetric session key ($K_{session}$), which is protected by the job owner’s TPM. This prevents attackers from stealing or tampering with the Results.

7.2.3 Verification of the Results

At the job owner’s system, the job factory receives the message R and decrypts it using the sealed session key (**6**, Figure 7). Note, if the job factory has been modified during the job execution period, the session key will no longer be accessible as the PCR value (corresponding to the trusted computing base) would have changed. Hence, a compromised job factory can neither read the Results nor return fabricated Results to the original application.

The decrypted message is forwarded to the job factory which compares the returned nonce (N_{JO}) with the original. A matching value verifies the accuracy and the integrity of the results. These are then delivered to the scientist’s application.

7.3 Distributed Data System

One of the pieces missing from the consensus view is a trustworthy, privacy-preserving analysis tool. As a potential solution, some combination of the ‘Privacy-Preserving Analytics’ software and attestation has been discussed in Section 6 to enable blind analysis of distributed data. This section expands on the idea and describes a ‘Blind Analysis Server’ (BAS) that allows analyses to be carried out securely via a remote server (see Figure 8): the user submits statistical queries by means of a job; analyses are carried out on the raw data collected from trustworthy sites, and only the processed results are delivered to the user.

The blind analysis server consists of the following components:

- The configuration resolver (see above).
- ‘Privacy Preserving Analysis Tool’ (PPAT) – this can be any software designed to reconcile distributed raw data and run analyses on the reconciled information; the tool enforces privacy policies on the processed results to protect the privacy of the sensitive data.
- ‘Privacy Policies’ – specify privacy rules governing the release of processed information; these are defined under strict ethics committee guidance to comply with legal and ethical undertakings made.

We imagine that an ethics committee would define the privacy policies for different types of analyses supported by the blind analysis server. This would be more practical than relying on the data owners to figure out their own sticky policies when it is not known which analyses might be performed. Moreover, it would be difficult to reconcile and make sense of such policies collected from different data sources.

The three components mentioned above form the trusted computing base of the blind analysis server. The server attests its security state through an attestation token (AT(BAS)):

$$AT(BAS) = (\ PCR\ Log\ ,\ AIK(BAS)\ ,\ \{cred(AIK)\}_{CA}\ ,\ P_K(BAS)\ ,\ \{cred(P_K)\}_{AIK}\ )$$

This attestation token contains a public key credential signed by the AIK(BAS) which identifies the private key as being sealed to the PCR value corresponding to its trusted computing base.

The rest of the section uses the healthcare grid example (see Section 2.2) to explain how the security operations have changed from the computational architecture with the blind analysis server in place.

7.3.1 Distribution of Job(s) through the Blind Analysis Server

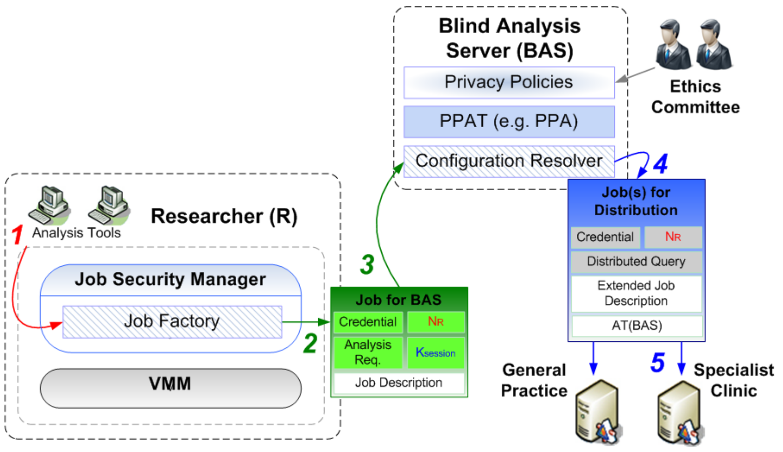

Figure 8 Operations of the Blind Analysis Server

A researcher is carrying out a study that looks at association between age (data available from a GP practice) and progression of colon cancer (data available from a specialist clinic). The researcher specifies the analysis requirements via an external analysis tool to observe how the cancer status has changed for patients aged between 45 and 55 (**1**, Figure 8). The analysis tool should provide an appropriate interface for capturing the information required to run the analysis queries.

The job factory, after receiving the analysis requirements, verifies the security state of the blind analysis server by comparing the PCR Log (obtained from the server's attestation token, AT(BAS)) against the known good configurations. It then creates a data access job (**2**, Figure 8) and encrypts the secret using the server's public key (P_K(BAS)). The analysis requirements are encrypted as part of the job secret. This job is then submitted to the configuration resolver running inside the analysis server (**3**, Figure 8).

The resolver is configured to manage the metadata of the participants' databases. Hence, by looking at the researcher's analysis requirements, the resolver is capable of selecting relevant sites and constructing distributed data access queries. The resolver selects trustworthy GP and specialist clinic systems to collect the data from and constructs a distributed query. The analysis server's sealed private key is used to sign the distributed query inside the TPM – it will only be accessible for signing if the trusted computing base has not been modified. This query as well as its signature is included in the job secret. The resolver dispatches a series of encrypted jobs to the policy enforcement points of the selected GP and specialist clinic systems (**4**, **5**, Figure 8).

The unencrypted part of the job now includes the analysis server's attestation token (AT(BAS)) which can be used by the job recipients (data owners) to verify the trustworthiness of the server before processing the jobs. The researcher's session key is omitted from the job secret since this key will only be used when the analysis server returns the final results to the researcher.

7.3.2 Operations of a Trustworthy Data Access Virtual Machine

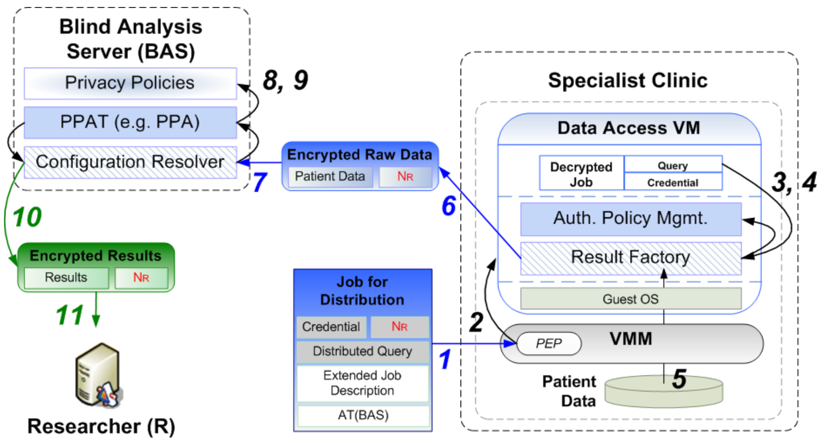

Figure 9 Operations of a Data Access Virtual Machine

Once the job arrives at the clinic, the policy enforcement point checks the security state of the analysis server using its attestation token (AT(BAS)) – this is how the job is authenticated at the clinic (**1**, Figure 9). It would detect a job dispatched from a compromised analysis server and prevent, for example, the server sending a malicious query. To simplify Figure 9, the policy enforcement point (which should be part of the monitor virtual machine) is drawn inside the virtual machine monitor.

After job authentication, the per-job virtual machine image files are checked for integrity. The middleware stack installed on this virtual machine provides a common interface for the job to access the patient data. For instance, if implemented in Java, such services would include the Java Database Connectivity, connection string and Java virtual machine. Again, the sealed private key – bound to the PCR values corresponding to both the trusted computing base and virtual machine files – is intended to guarantee that only a trustworthy virtual machine has access to the decrypted job secret to execute the query (**2**, Figure 8). The signature of the query is verified using the analysis server's public key (obtained from AT(BAS)): a valid signature proves that the query originates from a trustworthy analysis server and the encrypted secret correlates with the attestation token. The result factory checks the query for any attempt to exploit vulnerabilities in the database layer (e.g. SQL injection) before executing it (**3**, **4**, **5,** Figure 8).

A secure message containing the accessed data and the researcher's nonce (N_R) is encrypted with the data owner's symmetric session key (**6**, Figure 9). This session key, in turn, is encrypted using the analysis server's public key (obtained from the server's attestation token). Note, in contrast to the computational architecture, this result message is sent back to the analysis server and not to the researcher (**7**, Figure 9). The session key can only be decrypted if the analysis server's trusted computing base has not changed. Hence, a compromised server will not be able to steal the patient data.

7.3.3 Reconciliation of Collected Data

This result message and the encrypted session key arrive at the job distribution service of the resolver. First, the session key is decrypted using the sealed private key; the session key is then used to decrypt the result message. The returned nonce (N_R) is compared with the original to verify that the job has been processed (and the data has been accessed) through an integrity protected virtual machine.

The internal analysis tool (PPAT) reconciles the collected data and generates association between the patients' age and colon cancer progression (**8**, **9**, Figure 9). During this process, the privacy policies are enforced to protect privacy of the patient data. Attestation of the analysis server is sufficient to establish that these policies will be enforced correctly.

The final results are encrypted with the researcher's session key (obtained from the original job secret) and sent back to their job security manager (**10**, **11**, Figure

9). The researcher studies these anonymised results via the external analysis tool, knowing that the results are accurate and their integrity has been protected.

8 Observations

This section explains how the proposed distributed systems are responsible for meeting the security requirements identified in Section 3, and bridging the gaps identified in Section 6. Remaining performance, whitelist management, and job delegation issues are also discussed.

8.1 Satisfying the Requirements

Job submission is a two-step process. First, a job is submitted to the local configuration resolver; the job secret is encrypted using the resolver's public key. The sealed key approach ensures that only a securely configured resolver can decrypt the job secret. Second, the resolver selects a trustworthy participant suitable for running the job; the job secret is encrypted using the public key of this selected participant and dispatched through an untrusted public network. The private half is strongly protected by the participant's TPM. These features are responsible for meeting the 'secure job submission' requirement (see Requirement 1).

A combination of the sealed key mechanism and attestation is responsible for meeting the 'trustworthy execution environment', 'authorisation policy management', and 'job isolation' requirements (see Requirements 2, 3, 4). The trustworthiness of the trusted computing base and per-job virtual machine images of the participant are verified when they register with the local resolver. In this way, the resolver maintains a list of trustworthy participants.

The job is dispatched with its secret encrypted using the selected participant's public key. The private half is only accessible if neither the trusted computing base nor the virtual machine image has changed. The integrity of the virtual machine image is verified with runtime measurement of the files. These features are intended to guarantee a trustworthy execution environment that contains a securely configured authorisation policy management service. Moreover, the verification of the trusted computing base is sufficient to know that the virtual machine monitor is securely configured to provide strong isolation between the job virtual machines.

Virtual machine isolation ensures that the code is executed free from any unauthorised interference, including threats from rogue administrators to subvert the results. These results, before being sent back, are encrypted using the job owner's symmetric key that is strongly protected by the job owner's TPM. These features satisfy the 'protecting the results' requirement (see Requirement 5).

Finally, the provision of the blind analysis server aims to satisfy the 'digital rights management' and 'blind data analysis' requirements (see Requirements 6, 7). The data owners verify the security state of the blind analysis server before allowing the query to run. Two properties checked are: (1) the state of the 'privacy preserving analysis tool' installed, and (2) the integrity of the data privacy policies. The accessed data is encrypted in a way that only a securely configured server can decrypt the data. These properties provide assurance that the integrity protected policies will be enforced correctly upon data processing, and only the anonymised results will be released to the user.

8.2 Filling in the Missing Pieces

In Section 6, we identified the missing components from existing trusted virtualization approaches. This section explains how these missing components are dealt with in the proposed systems.

Many of the existing work on trusted distributed systems – likes of the Trusted Grid Architecture (Löhr, Ramasamy and Sadeghi 2007), trusted delegation for grid (Cooper and Martin 2006) and Terra (Garfinkel, et al. 2003) – expect the end user to collect the attestation tokens (or PCR quotes) from participant machines and make trust decisions. Typically, this requires the user to (1) manage an application whitelist of trustworthy system configurations, (2) discover available participant machines and download their attestation tokens, and (3) verify their security configurations and make trust decisions. We argue that this requirement is unrealistic given the dynamic nature of large-scale distributed systems.

In the proposed systems, the configuration resolver manages all of the above operations on the user's behalf, taking away the burden of performing attestation and managing application whitelists from the user. The participants' security configurations are verified when they first register with the resolver, and only those considered trustworthy are listed and used.

After identifying the local resolver, the user can submit their jobs without worrying about attestation or what the security configurations mean to them. All the user has to do is specify the job description and their security preferences, and submit their jobs to the correct resolver. The resolver will find trustworthy participants that satisfy the user's preferences and dispatch the jobs on the user's behalf. This shift in responsibility (of performing attestation) should also improve overall *usability*.

Another missing piece is the integrity-report based job verification mechanism for the participant. In most of the existing work, the participant relies on basic PKI to identify users and authenticate job virtual machines; however, this provides no information about the security state of the job virtual machines.

In the proposed systems, the participant creates baseline virtual machine images which are allowed to be deployed on their machine, and seals the private-half of the TPM-key to the image files. This sealed key approach, together with run-time verification of the image files, guarantees the integrity of the job virtual machines. With these mechanisms in place, the participant can be assured that only those securely configured are ever used on their machine as baseline images for executing the jobs.

8.3 Performance Degradation

One of the key drivers behind the development of computational distributed systems is high performance (Foster and Kesselman 1999). However, the suggested use of virtualization and various cryptographic operations necessarily incur a performance penalty (Pradheep, et al. 2005).

Running a job inside a virtual machine requires extra information flow upon accessing the hardware. Each I/O request would go through a number of virtual device drivers before reaching the physical hardware; the same applies when receiving an I/O response. A recent study (Ruth, et al. 2005) suggests that a typical virtualized, distributed system incurs 20 percent performance penalty over native execution. With the introduction of native hardware support in all recent CPU architectures (Adams and Agesen 2006) (Strongin 2005), however, this overhead can be minimised with time to come.

Moreover, attestation involves expensive public key operations for signing the PCR values and validating the signatures. It also involves comparing the reported PCR event log against the whitelist entries and verifying the trustworthiness of a platform.

To improve performance, the attestation service (of the configuration resolver) could be configured to minimise the use of attestation. Since the trusted computing base of the participant platform is designed to be relatively static, the previous attestation results could be used again and again up to a given expiry date. A fresh attestation would be performed when the previous results expire, avoiding the need to attest every time a job is submitted. If the trusted computing base changes at a time before the expiry date, the sealed key mechanism would detect it and inform the resolver. The resolver would then request for the latest configuration token to perform a fresh attestation.

8.4 Whitelist Management

In systems spanning multiple administrative domains, different domains will likely have different software requirements and whitelist of acceptable configurations. While the administrators for one domain will be competent with the required list of software and their acceptable configurations for the local users, they will not know about all the software requirements in other domains. In consequence, multiple configuration resolvers could introduce availability issues depending on the level of inconsistency between their whitelists.

For example, if configuration resolver A is more active in inspecting software vulnerabilities and updating the whitelist entries than other domains, configuration tokens collected from configuration resolvers B, C, and D are likely to be classified as untrustworthy by resolver A, and their services will not be advertised to the users in Domain A. In order to minimise the level of inconsistency, the whitelist manager (in the resolver) needs to support functions that would enable efficient discovery and sharing of whitelist updates. The authors participated in research (Huh, et al. 2009) that explores these issues in detail, and suggests what the content of whitelist entries should be and how entry update messages should be shared.

8.5 Job Delegation

In practice, the job recipient might delegate some parts of the job on to other participants – this is known as *job delegation*. In the Trusted Grid Architecture (Löhr, Ramasamy and Sadeghi 2007), the user is capable of verifying the service providers' platform configurations against a set of known good values (good_U). Using its job submission protocol, the user may also check to see if the service provider's list of known-good values (good_P) – which specifies all the acceptable configurations of possible job delegatees – satisfy the condition $\mathsf{good}_P \subseteq \mathsf{good}_U$. If this condition is satisfied, the user submits the job to the provider knowing that the job will only be delegated to other service providers whose platform configurations also satisfy good_U. However, the main concern with this type of approach is that the burden of managing the whitelists (good_U, good_P) rests on the users and the service providers.

Although job delegation has not been considered in the proposed systems, the configuration resolver could be configured to verify the configurations of possible job delegatees before dispatching the job. Since the resolver already has access to all the trustworthy participants' platform configurations (configuration tokens), it could exchange several messages with the potential job recipient to determine whether all the possible job delegatees are also trustworthy. This would involve the job recipient sending a list of identities of the possible delegatees to the resolver, and the resolver checking to see if all of the possible delegatees are registered. The job would only be dispatched if all of the delegatees are also trustworthy.

The advantage of this approach is that the users and service providers would not have to worry about maintaining up-to-date whitelists, or attesting and verifying the trustworthiness of the possible job delegatees.

9 Example Integration with the UK National Grid Service

This section demonstrates how the proposed security components ‘could’ be integrated with the UK National Grid Service (Geddes 2006). Some of the important practicality and interoperability issues are also uncovered.

9.1The National Grid Service Overview

The National Grid Service is a UK academic research grid, intended for production use of computational and data grid resources spanning multiple institutions across the country. The aim of the National Grid Service is to provide a reliable and trusted service using open, standards-based access to the distributed resources.

The grid consists of four core sites at Oxford, Manchester, Leeds, and STFC-AL, as well as five partner sites at Cardiff, Bristol, Lancaster, Westminister and Queens. Each site contributes to the provision of computational or data nodes. The nodes sitting on the core sites provide transparent access to the resources by using an identical middleware stack and similar filesystems, whereas the partner sites provide a more heterogeneous environment.

Each site consists of several Computing Elements (CEs) which are the front ends to a number of worker nodes (resource providers). The CEs provide gatekeeper and job-manager functionality. A gatekeeper receives a job from a resource broker and calls a job-manager to submit the job to a worker node through the Portable Batch System. Each CE uses its own information service, known as the Grid Resource Information Service (GRIS), to publish static and dynamic information about the resource availability. The GLUE Information Schema (Andreozzi, et al. 2009) is used for publishing such information. At each site, a LDAP directory called the Grid Index Information Service (GIIS) is used to collate information from many GRISs. Information from all the sites is collected and aggregated by the Berkeley Database Information Index system (Berkeley database information index v5 2009) (a central information repository), which holds information about all services and resources available in the grid. It queries the GIIS at each site to collect this information. The LDAP is used for making the aggregated information available to the users.

9.2 Integration

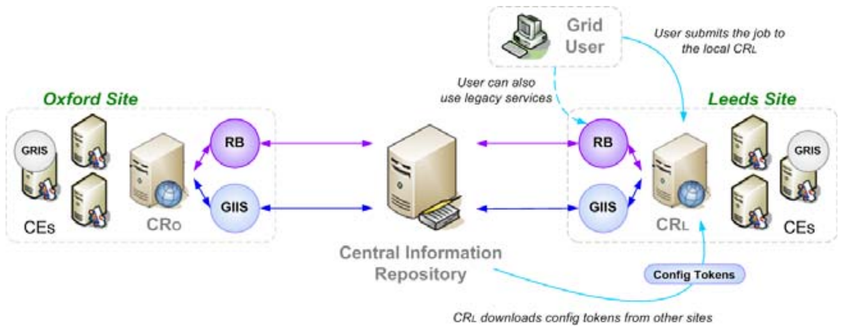
Grid User
User submits the job to the local CRL
User can also use legacy services
Oxford Site
Leeds Site
GRIS
RB
GIIS
CEs
CRO
CRL
Central Information Repository
Config Tokens
CRL downloads config tokens from other sites

Figure 10 Example Integration with the UK National Grid Service

As the first step of integration, the configuration resolver would be deployed at each site to publish a filtered list of configuration tokens (representing trustworthy participants) through the GIIS (see Figure 10). These tokens would have to be signed by the configuration resolver for authenticity and to indicate that these represent trustworthy participants.

The central information repository would then query the GIIS at each site to collect these tokens and make them available to all the configuration resolvers. In this scenario, the GIIS would merely act as a proxy between the resolver and the central information repository. The signatures of the tokens would be validated by the central information repository before aggregating them. The resolvers would have to be authenticated at the central information repository before being granted access to the aggregated tokens; verification of the resolvers' digital certificates and attestation tokens would be sufficient for this purpose. This integration would allow each site, through their own configuration resolver, to discover all the trustworthy nodes available across the entire grid.

Consider a job submission scenario. A researcher, who wishes to run their job in the National Grid Service, submits a job to the local configuration resolver. First, the resolver communicates with the local resource broker to discover resource availability through the central information repository. Note, this is a 'push architecture' where the resource broker polls all CEs (through the central information repository) to find out about the availability of the worker nodes. The resolver, using the LDAP, downloads the configuration tokens for those with available resources from the central information repository. Tokens that match the job requirements are returned, representing trustworthy, relevant participants available in other sites. The resolver then iterates through each token and verifies the trustworthiness of the reported configurations by comparing the PCR values against the local whitelist. Only those with acceptable configurations will be selected and merged with the tokens from the local site. Finally, the resolver selects the most suitable participant, encrypts the job secret with the selected participant's public key, and dispatches it.

The encrypted job secret and session key as well as the extended job requirements are sent to the CE to which the selected worker node is connected. The job-manager of this CE reads the unencrypted job requirements and schedules the job to the selected worker node. The middleware services as such will not be able to read the encrypted job secret and session key.

When these arrive at the worker node, its policy enforcement point first attempts to decrypt the job secret. The sealed key approach ensures that the private key is only accessible if the node is still running with trustworthy configurations. Then an integrity protected virtual machine is launched to provide an isolated, protected job execution environment. The sensitive models run within this virtual machine, and the generated results are encrypted using the session key and returned to the researcher.

9.3 Observations

There would be a significant overhead involved in upgrading the participant systems to support trusted computing and virtualization. Various security virtual machines will have to be installed and the virtual machine monitor will have to be configured to manage these securely. Although this is a large change, the advantage of the discussed approach is that legacy components like the GIIS and the central information repository can be used with only small modification.

Moreover, many existing cloud systems (Nurmi, et al. 2009) (Amazon Elastic Compute Cloud n.d.) (Enomaly - Product Overview n.d.) already support virtualization and submission of job virtual machines. With the recent introduction of hardware support for virtual machine execution (see Section 4.4), it seems likely that future developments will also make use of virtualization. The administrative tasks involved in upgrading such systems would be much smaller.

Despite the security enhancements, the use of the configuration resolver will increase the number of messages being exchanged upon job submission. The user submits a job to the local configuration resolver rather than to the Resource Broker. The resolver requests configuration tokens from the central information repository and filters the trustworthy participants. Once these checks are done, it encrypts the job with the selected participant's public key and submits the job on the user's behalf.

These extra messages and cryptographic operations will affect the overall performance of job submission. However, for those wanting to submit performance critical jobs, the legacy services are still available for use. Such jobs can be submitted directly to the local Resource Broker and skip all the trusted computing operations (see Figure 10). Usability will not be affected as much since the user relies on the resolver to carry out attestation and job submission.

10 Conclusion

A wide range of research is conducted, archived, and reported in the digital economy. Different types of distributed systems have been deployed over the years to facilitate the collection and modeling of the dispersed data, or the sharing of the computational resources. A problem arises, however, when the models or data have commercial value. They then become lucrative targets for attack, and may be copied or modified by adversaries. Despite ongoing research in the area of distributed system security, there remains a ‘trust gap’ between the users’ requirements and current technological capabilities.

To bridge this ‘trust gap’, we proposed two different types of distributed systems, one applicable for a computational system and the other for a distributed data system. Central to these systems is the *configuration resolver*, which maintains a list of trustworthy participants available in the virtual organisation. Users submit their jobs to the configuration resolver, knowing that their jobs will be dispatched to trustworthy participants and executed in protected environments. As a form of evaluation, we suggested how these ideas could be integrated with the UK National Grid Service, and highlighted the potential security enhancements.

As high performance is one of the key drivers behind the development of computational distributed systems, the proposed security mechanisms sit uneasily with these aspirations. Despite several suggestions for improving performance (see Section 8.2), a more accurate assessment would be necessary to analyse the performance implications and devise enhancement strategies. Hence, future work should consider constructing a prototype implementation of the proposed components, integrating them with existing systems like the National Grid Service, and measuring the performance overhead. This work will also help uncover other interoperability and usability issues.

Acknowledgments The work described is supported by a studentship from QinetiQ. David Power, Mark Slaymaker, and Peter Lee provided help with the healthcare grid example. David Wallom, Steven Young, and Matteo Turilli provided insights on the National Grid Service.

References

2009. http://www.climateprediction.net/ (accessed February 08, 2010).

Adams, K, and O Agesen. "A comparison of software and hardware techniques for x86 virtualization." *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems.* ACM, 2006. 2-13.

"Amazon Elastic Compute Cloud (Amazon EC2)." *Amazon Web Services.* http://aws.amazon.com/ec2/ (accessed February 17, 2010).

Andreozzi, S, et al. "GLUE Specification v. 2.0." February 2009. http://forge.gridforum.org/sf/docman/do/downloadDocument/projects.glue-wg/docman.root.drafts.archive/doc15023.

Barham, P, et al. "Xen and the art of virtualization." *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles.* ACM, 2003. 164-177.

"Berkeley database information index v5." *EGEE Web.* November 2009. https://twiki.cern.ch/twiki//bin/view/EGEE/BDII.

Cooper, A, and A Martin. "Trusted Delegation for Grid Computing." *The Second Workshop on Advances in Trusted Computing.* 2006.

Duncan, G T, and R W Pearson. "Enhancing Access to Microdata While Protecting Confidentiality." *Statistical Science*, 1991: 6(3):219-232.

"Enomaly - Product Overview." *Enomaly.* http://www.enomaly.com/Product-Overview.419.0.html (accessed February 17, 2010).

Figueiredo, R J, P A Dinda, and J A Fortes. "A case for grid computing on virtual machines." *23rd IEEE International Conference on Distributed Computing Systems (ICDCS'03).* IEEE Computer Society, 2003.

Foster, I, and C Kesselman. "The Grid: Blueprint for a New Computing Infrastructure." Chapter 2: Computational Grids. Morgan-Kaufman, 1999.

Foster, I, C Kesselman, G Tsudik, and S Tuecke. "A security architecture for computational grids." *Proceedings of the 5th ACM conference on computer and communications security.* ACM, 1998. 83-92.

Freeman, R. "Medical records and public policy: the discursive (re)construction of the patient in Europe." *Workshop 9: 'Policy, Discourse and Institutional Reform.* ECPR Joint Sessions of Workshops, 2001.

Garfinkel, T, B Pfaff, M Rosenblum, and D Boneh. "Terra: A Virtual Machine-Based Platform for Trusted Computing." *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03).* ACM, 2003. 193-206.

Geddes, N. "The National Grid Service of the UK." *e-Science and Grid Computing, International Conference on*, 2006: 94.

Grawrock, D. "The Intel Safer Computing Initiative." 119-142. Intel Press, 2006.

Hohmuth, M, M Peter, H Hartig, and J S Shapiro. "Reducing TCB size by using untrusted components: small kernels versus virtual-machine monitors." *EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop.* ACM, 2004. 22.

Huh, J H, J Lyle, C Namiluko, and A Martin. "Application Whitelists in Virtual Organisations." *Future Generation Computer Systems*, 2009: (Under Revision).

Keahey, K, K Doering, and I Foster. "From sandbox to playground: Dynamic virtual environments in the grid." *5th International Conference on Grid Computing (Grid 2004).* IEEE Computer Society, 2004.

Löhr, H, H V Ramasamy, and A R Sadeghi. "Enhancing Grid Security Using Trusted Virtualization." *Autonomic and Trusted Computing.* 372-384: Lecture Notes in Computer Science, 2007. 372-384.

Luna, J, M D Dikaiakos, T Kyprianou, A Bilas, and M Marazakis. "Data Privacy considerations in Intensive Care Grids." *Global Healthgrid: e-Science Meets Biomedical Informatics.* IOS press, 2008. 178-187.

Maguire, T, and D Snelling. "Web Services Service Group 1.2 (WS-ServiceGroup)." OASIS Open, 2004.

Mao, W, F Yan, and C Chen. "Daonity: grid security with behaviour conformity from trusted computing." *STC.* ACM, 2006. 43-46.

Nagarajan, A, V Varadharajan, and M Hitchens. "Trust management for trusted computing platforms in web services." *STC '07: Proceedings of the 2007 ACM workshop on Scalable trusted computing.* ACM, 2007. 58-62.

Nurmi, D, et al. "The Eucalyptus Open-Source Cloud-Computing System." *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid.* IEEE Computer Society, 2009. 124-131.

O'Keefe, C M. "Privacy and the Use of Health Data - Reducing Disclosure Risk." *Health Informatics*, 2008: 3(1).

Power, D J, E A Politou, M A Slaymaker, and A C Simpson. "Towards secure grid-enabled healthcare." *Software Practice and Experience*, 2002.

Pradheep, S S, S Santhanam, P Elango, A Arpaci-dusseau, and M Livny. "Deploying Virtual Machines as Sandboxes for the Grid." *In Second Workshop on Real, Large Distributed Systems (WORLDS 2005).* 2005. 712.

Ruth, P, x Jiang, D Xu, and S Goasguen. "Virtual Distributed Environments in a Shared Infrastructure." *Computer*, 2005: 38(5):63-69.

Sadeghi, A R, and C Stuble. "Property-based Attestation for Computing Platforms." *NSPW '04: Proceedings of the 2004 workshop on New security paradigms.* ACM, 2004. 67-77.

Sadeghi, A R, and C Stüble. "Taming "Trusted Platforms" by Operating System Design." *Information Security Applications.* Lecture Notes in Computer Science, 2004. 2908:1787-1801.

Sailer, R, T Jaeger, X Zhang, and L V Doorn. "Attestation-based policy enforcement for remote access." *CCS '04: Proceedings of the 11th ACM Conference on Computer and Communications Security.* ACM, 2004. 308-317.

Simpson, A C, D J Power, M A Slaymaker, and E A Politou. "GIMI: Generic Infrastructure for Medical Informatics." *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems.* 2005. 564-566.

Strongin, G. "Trusted computing using AMD "Pacifica" and "Presidio" secure virtual machine technology." *Information Security Technical Report*, 2005: 10(2):120-132.

Stumpf, F, M Benz, M Hermanowski, and C Eckert. "An Approach to a Trustworthy System Architecture Using Virtualization." *Autonomic and Trusted Computing.* Lecture Notes in Computer Science, 2007. 191-202.

Sugerman, J, G Venkitachalam, and B Lim. "Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor." *Proceedings of the General Track: 2002 USENIX Annual Technical Conference.* USENIX, 2001. 1-14.

TCG. "TCG Infrastructure Working Group Architecture Part II - Integrity Management." November 2006. http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_architecture_part_ii__integrity_management_version_10.

"TPM Main Specification Version 1.2." *TCG Workgroup.* 2003. http://www.trustedcomputinggroup.org/resources/tpm_main_specification.

Trusted Computing Group Backgrounder. 2006. https://www.trustedcomputinggroup.org (accessed February 09, 2010).

Vejda, T, R Toegl, M Pirker, and T Winkler. "Towards Trust Services for Language-Based Virtual Machines for Grid Computing." *TRUST.* Lecture Notes in Computer Science, 2008. 48-59.

Wallom, D C, and A E Trefethen. "OxGrid, a campus grid for the University of Oxford." *UK e-Science All Hands Meeting.* 2006.

Wang, D, and A Wang. "Trust Maintenance Toward Virtual Computing Environment in the Grid Service." *APWeb.* Lecture Notes in Computer Science, 2008. 166-177.

Xen. "Xen: Enterprise Grade Open Source Virtualization A XenSource White Paper." 2005. http://xen.xensource.com/files/xensource_wp2.pdf.

Yau, P W, A Tomlinson, S Balfe, and E Gallery. "Securing Grid Workflows with Trusted Computing." *ECCS (3).* Lecture Notes in Computer Science, 2008. 510-519.