Towards a Trusted Grid Architecture

Andrew Cooper

St. Cross College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2009

Title:	Towards a Trusted Grid Architecture
Name of candidate:	Andrew Cooper
College:	St. Cross
Degree:	D.Phil
Term of submission:	Hilary 2009

Abstract

The malicious host problem is challenging in distributed systems such as grids and clouds. Rival organisations may share the same physical infrastructure. Administrators might deliberately or accidentally compromise users' data.

The thesis concerns the development of a security architecture that allows users to place a high degree of trust in remote systems to process their data securely. The problem is tackled through a new security layer that ensures users' data can only be accessed within a trusted execution environment. Access to encrypted programs and data is authorised by a *key management service* using trusted computing attestation. Strong data integrity and confidentiality protection on remote hosts is provided by the *job security manager* virtual machine.

The trusted grid architecture supports the enforcement of digital rights management controls. Subgrids allow users to define a strong trusted boundary for delegated grid jobs. Recipient keys enforce a trusted return path for job results to help users create secure grid workflows. Mandatory access controls allow stakeholders to mandate the software that is available to grid users.

A key goal of the new architecture is backwards compatibility with existing grid infrastructure and data. This is achieved using a novel virtualisation architecture where the security layer is pushed down to the remote host, so it does not need to be pre-installed by the service provider. A new attestation scheme, called *origin attestation*, supports the execution of unmodified, legacy grid jobs. These features will ease the transition to a trusted grid and help make it practical for deployment on a global scale.

Acknowledgements

To Jess and my family for, if not understanding, then at least putting up with 'The PhD' *for all these years — which is in some ways more difficult than going through the process itself.*

Related Publications

Some of the material in this thesis has been published in peer-reviewed conference papers and articles, including the following:

- Andrew Cooper and Andrew Martin. Towards a Secure, Tamper-Proof Grid Platform. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, pages 373–380, May 2006. IEEE Computer Society.
- Andrew Cooper and Andrew Martin. Towards an open, trusted digital rights management platform. In *Proceedings of the Sixth ACM workshop on Digital rights management (DRM '06)*, pages 79–88, October 2006. ACM Press.
- Andrew Cooper and Andrew Martin. Trusted Delegation for Grid Computing. In *The Second Workshop on Advances in Trusted Computing (WATC '06 Fall)*, November 2006.
- Andrew Cooper and Andrew Martin. Building a Trustworthy Platform for Grid Computing. *International Transactions on Systems Science and Applications*, 3(3):193–202, October 2007.

The material included in this thesis is, except where indicated, the author's own work.

Contents

List of Figures

1 The Need for a Trusted Grid

1.1 Motivation and Introduction

Around 2001 HP began an initial attempt to promote their Utility Data Centre as a fundamentally new way of delivering IT services. The idea of grid computing (which was originally conceived decades earlier [87]) was that storage, processing and networking resources could be scaled on demand and consumed in the same way as a utility such as electricity. It was around this time that the author of this thesis became aware of the unique security challenges facing grid computing, during a joint research project with HP labs.

Among the largest HP Utility Data Centre customers were computer animated film production companies. In order to keep pace with the latest technology, in-house IT equipment would have to be completely replaced each time a new feature film was made. By outsourcing their IT to HP, the production company could avoid the significant costs associated with managing and upgrading their internal IT infrastructure.

HP had become aware that the main barrier to the wide-scale adoption of grid computing was customers' fears around the security of their data after outsourcing it to a third-party such as HP. Customers needed to be convinced that their data would be isolated and protected whilst it was processed in a data centre that might be shared by their main commercial rivals. For example the loss of a single frame of animation could do grave damage to the production company by revealing details of an unreleased film in the context of a fiercely competitive industry.

Fast forward to today and there is more interest than ever in grid-like distributed systems. Large software companies such as Microsoft, Sun, Adobe and Amazon have embraced so-called cloud computing [111], a similar concept to HP's Utility Data Centre. A large amount of research is ongoing to develop interoperable grid services that offer the opportunity to create a truly global-scale grid. This vision would allow the formation of so-called virtual organisations that offer collaboration upon huge quantities of data, compute power and specialist equipment that are currently stove-piped within separate physical adminis-

trative domains.

Grid systems are emerging as popular platforms for distributed computing, but these advances create difficult challenges for security. Grids join together multiple organisations allowing them to share their IT resources and data. This close collaboration enables improved ways of working but it requires a high degree of trust that the members will safeguard each others' information.

Traditional security measures focus on protecting systems from attack by malicious users. In the grid the reverse problem also occurs. If grid systems are malicious the confidentiality, integrity and availability of users' data can be compromised. This can happen through deliberate or accidental attacks by other users of the grid, administrators and external attackers. The main problem addressed in this thesis is how to solve the malicious host problem in a grid.

Solving the malicious host problem is challenging. Data owners cannot easily enforce security controls on remote grid systems outside their administrative domain. A robust and secure solution to this problem must be found before companies and organisations will trust the grid with their most sensitive intellectual property and scientific data. Only then can the grid realise its potential as a global-scale distributed computing platform.

The goal of this chapter is to both motivate the need for a trusted grid and to specify the thesis problem in further detail. The first section examines the common security problems faced by users in a range of popular distributed computing scenarios. These use-cases share many properties in common and might each be seen as a different specialisation of the generic grid concept. Indeed, the term *trusted grid* is used throughout this thesis to refer to a security solution applicable to many of these types of distributed systems. The analysis of these use-cases is then used to derive the detailed requirements for a trusted grid in the following section.

1.2 Distributed Systems Use-cases

1.2.1 Cloud Computing

Cloud computing allows companies to outsource their IT infrastructure to a third-party service provider (see Figure 1). A number of high-profile vendors such as Amazon [53] and Sun [59] currently offer cloud-based services. Customers can gain enhanced scalability and performance because the IT resources allocated to them can be dynamically adjusted to meet their demands. Companies no longer have to provide and manage their own in-house IT infrastructure for tasks that are moved to the cloud.

Market analysts claim that a fundamental shift is underway towards the delivery of software as a service [109]. In this approach the cloud is used to provide ubiquitous access to software services. So-called Web 2.0 technologies such as GoogleDocs offer business productivity applications delivered through a web browser [28]. One key advantage of this approach is that users around the world can collaborate on shared documents.

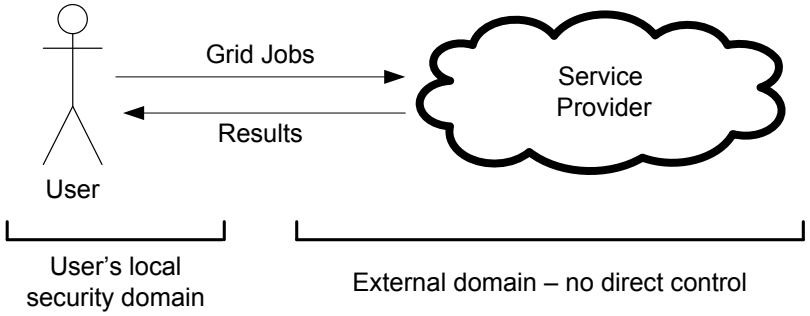

Figure 1: Cloud Computing Use Case

Many companies might be put off using the cloud (and other grid systems) because of security concerns [66, 106, 128]. A significant fear for companies is that their data is moved out of the safety of their secure network perimeter. The company is no longer in control of its information and is wholly reliant on the third-party service providers to protect their sensitive intellectual property.

Service providers might find it difficult to reassure customers that their data is secure on the cloud. The cloud services are generally based on proprietary systems [53]. Service

providers may be unable to provide suitable information to allay customers' security concerns because detailed technical information is regarded as the intellectual property of the service provider. Instead customers might have to rely on insurance terms and contractual liability to force service providers to adequately secure their data.

Once data is placed in the cloud it is at risk from severe new security threats. One key threat comes from other users of the same service provider. Take, for example, a film production company using the cloud to render animation frames. The film company might be sharing the same cloud provider, and even the same remote computer, with a rival company or other interested party. The theft of a single animation frame could do severe damage to the production company in a competitive marketplace.

Insider attack is another key threat. The cloud administrators have access to customers' information. The administrators could deliberately gain unauthorised access to sensitive data, or might be coerced into doing so (for example using social engineering). Again, the customer has to trust that the service provider has put in place suitable security measures to guard against these attacks.

It is difficult to protect grid-like systems from attack even if they are adequately secured. A typical service provider's infrastructure might consist of thousands of complex software and hardware systems. A single unpatched security vulnerability or mis-configuration of any of these systems could result in a successful attack.

There is a history of security vulnerabilities in the popular Amazon EC2 cloud service. A bug existed that caused data belonging to users to remain on the system after it was re-allocated to another customer, resulting in a loss of confidentiality [53]. The authentication mechanism used by Amazon is reported to be weak because it relies on a password that can be reset by email [53]. Attacks on the authentication system potentially allow attackers to gain unauthorised access to other customers' data.

1.2.2 Computational and Data Grids

Similar to the cloud, computational grids allow users to submit grid jobs for execution on remote systems. One of the defining aspects of a computational grid is that multiple

administrative domains participate to form a virtual organisation (see Figure 2). The members can collaborate on a joint project by sharing workloads and resources. There is a vision that grids could reach a global scale through the widespread adoption of standardised middleware software that handles all the common aspects of grid computing.

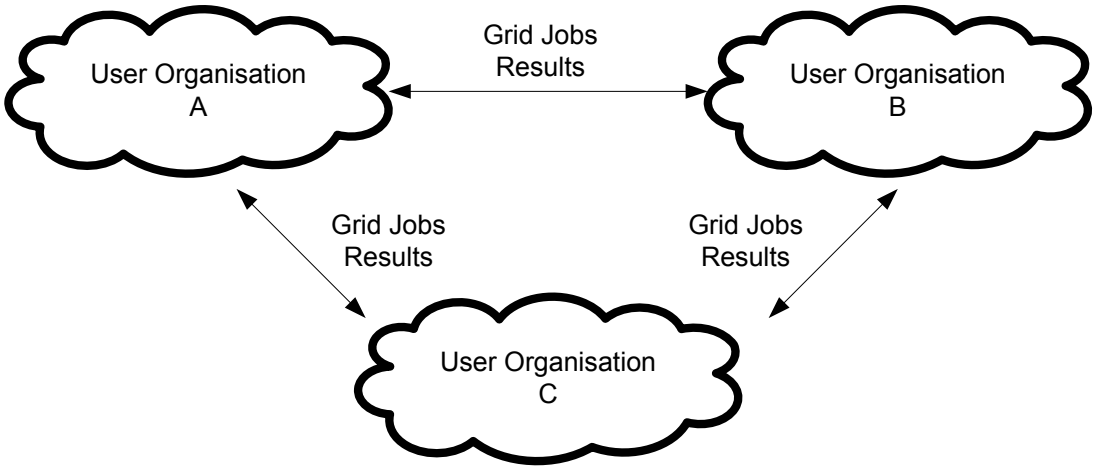

Figure 2: Computational and Data Grid Use Case

In computational grids the focus is on sharing processing resources. In many ways this is similar to the cloud use-case considered in the previous section, except each member of the virtual organisation acts as a separate service provider. Computational grids also share similar security challenges with cloud computing because there is a need to trust a third-party with potentially sensitive software and data.

The grid use-case becomes more differentiated when data grids are considered. Data grids enable collaboration on a large distributed data repository. Data that was previously held by individual organisations is made available on the grid. This is achieved by placing a standardised set of grid middleware services in front of a wide variety of incompatible storage systems.

Sharing medical data is seen as an important application for data grids [127]. Each hospital trust in the UK holds its own database of patient medical records. Medical data is currently not accessible outside the hospital trust. Resources might be put to better use if a doctor in another hospital could make a remote diagnosis. A data grid can be used to

make medical data widely available but the solution must meet the strong legal and ethical constraints that are in place to protect patients' medical records.

One key purpose of a data grid is to share data that was previously inaccessible to a large community. In a cloud, data ownership generally remains with the user. The cloud forms a logical extension of an organisation's internal IT infrastructure. In a data grid, on the other hand, data is explicitly made available to others. This introduces the problem of how to allow other members of the grid to access shared data securely.

Trust is an important factor in grids. Users must trust the other members of the grid to protect their data while it is in their possession. Trust is difficult to achieve. As the grid grows in size the likelihood that one of the members forgets to install a security patch, or mis-configures the software, increases. As data grids become larger there is a greater likelihood of security vulnerabilities.

There are underlying security problems in grid software that remain unsolved. The software that drives grids is inherently complex, leading to a high likelihood of bugs and vulnerabilities (as will be argued in Chapter 6). This difficulty is referred to in this thesis as the grid middleware problem. Higher level security services cannot be trusted because they can be bypassed by compromising the underlying grid middleware layer. Until this problem is solved a trusted grid cannot be constructed because there is no solid foundation on which it can be built.

Security vulnerabilities also arise from a lack of isolation in data grids. Typically each member will add grid systems to their existing infrastructure. Some members may fail to adequately isolate the grid middleware from their internal systems. Inadequate security on internal systems will lead to attacks on grid data belonging to other organisations.

The likelihood of security vulnerabilities in data grids mean that simple access controls are not always enough to protect sensitive information. Take the example of a hospital trust opening up patient records for medical research. The hospital trust is unlikely to have the resources to perform a security audit of a large number of research organisations to ensure medical data is appropriately protected.

Instead digital rights management (DRM) technology can be used to secure data even

after access has been authorised [131]. The advantage of DRM is that data is protected at all times wherever the object is located. If there is a deliberate insider attack by authorised users, or accidental mis-configuration of systems, the DRM controls should remain uncompromised. DRM can therefore be used to implement strong mandatory access control policies to protect owners' data throughout a large-scale grid.

Digital rights management can be used to enforce rich access control policies. In addition to controlling who can access data, DRM can be used to constrain where data is accessed from, and how it is accessed. For example a public workstation might be treated as less secure than a dedicated server in controlled facilities. DRM can also be used to restrict data access to authorised applications. For example access could be restricted to applications that display sanitised medical records to avoid leaking sensitive information [19].

1.2.3 Public Resource Computing

Public resource computing (PRC) [8] is part of a vision of a global grid in which every machine on the planet could potentially participate. PRC opens the grid up to include shared resources such as home computers and business workstations (see Figure 3). In practice these machines are rarely used to their full potential. PRC middleware software allows the spare CPU cycles to be used to solve challenging science problems by pooling together their

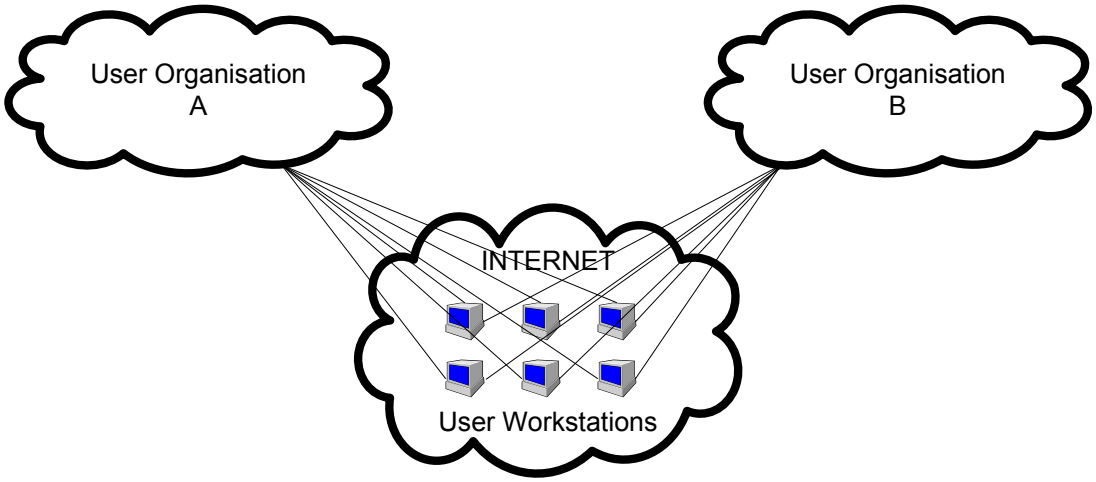

Figure 3: Public Resource Computing Use Case

resources into one giant distributed supercomputer. Public resource computing represents a vision of a huge and globally available grid.

PRC projects tend to investigate popular science themes in order to engage users. This is necessary because users are not generally paid to participate. One of the most high-profile PRC projects is Seti@home, which claims a throughput of over 27 TeraFLOPS [8]. The Seti@home project analyses radio signals in search of extra-terrestrial life. Another example is the climate*prediction*.net [159] project, which aims to model future climates to predict the effects of global warming.

A key security requirement in these distributed science projects is integrity of the results [160]. Many PRC projects provide encouragement to users through a leaderboard showing the most dedicated participants. Users might seek to return false results to raise their position on the leaderboard. Rival projects, or other motivated individuals, might also seek to discredit the results by attacking their integrity.

Public resource computing technology can also be used on a smaller scale. Companies could use the technology to harness the spare CPU cycles of business machines throughout an enterprise [94]. Security threats in this use-case exist from internal employees. For example financial and business information might be processed on the same computer as a user from another department. These employees might seek to commit corporate espionage by selling a company's intellectual property.

Science-based grid computing raises unique security challenges. The participants have little incentive to protect the grid data since it does not belong to them – the users are under no obligation to the science project. The client software runs on home computers that may contain viruses and other malicious code. The administrators of the computer – the home users – are completely untrusted and may be actively malicious. Users have physical control of the computer, and access to all the data and software stored on it, so countermeasures cannot easily be put in place.

The current solution to integrity issues in public resource computing is unsatisfactory. The same grid job is sent to many clients and the results are only accepted if they match [7]. This approach reduces performance because work is repeated.

A better way to enforce integrity is to reliably measure the grid job to confirm it is unaltered. Unfortunately these solutions (described in Chapter 5) do not work well in a public resource computing environment because the data is constantly changing as new jobs are received. Maintaining integrity in a dynamic and uncontrolled environment is a challenging problem.

Previous work by this author exposed security vulnerabilities in the grid middleware platform on which Seti@home and many other popular PRC project are based [32]. These vulnerabilities would allow an attacker to execute a Trojan horse on every participant's computer. The legitimate software is exchanged for a Trojan horse which could compromise the grid data on every client as well as users' personal information. These attacks demonstrate how easily grid functionality can be abused if the host computer has been compromised.

1.3 The Goals of a Trusted Grid

The example scenarios have demonstrated the need for a trusted grid. A trusted grid must be able to deal with the threats and vulnerabilities present in large-scale distributed systems. The goal of the thesis is to design a trusted grid architecture that helps solve the malicious host problem. The fundamental problem that must be solved is how users can establish trust in a remote computer that is outside of their control.

There are many different aspects to a trusted grid, but the security requirements that are most relevant to a broad range of use-case scenarios (described in the previous section) are as follows:

Security goals

1. **Solve the grid middleware problem** Protect grid data integrity and confidentiality against attacks by external attackers and users sharing the same service provider infrastructure (e.g. virus and Trojan horse attacks).

2. **Resist admin attack and error** Protect grid data integrity and confidentiality against accidental or deliberate attacks made by service provider administrators with

privileged access to the remote infrastructure.

3. **Digital rights management** Enforce digital rights management and mandatory access control policies to control how and where grid data can be accessed.

Data availability is an important security requirement for the grid. When users place critical software on the grid there will be an expectation that the results can be retrieved in a timely manner. Indeed the ability to cope with a hardware failure is a key advantage of computing using large distributed systems.

It is important to emphasise that availability in security terms often focuses more on malicious denial of service than benign failure. An attack that can bring down a system at the exact time of an attackers choosing can have devastating consequences, particularly for commercial companies for which financial losses can quickly mount up while systems are under attack.

Despite the importance of availability it has been decided, for a number of reasons, that the thesis will instead address the problem of data integrity and confidentiality. Firstly, availability is already improved because of the redundancy and fail-over mechanisms that are built into the grid. This makes denial of service attacks harder to perform successfully. Secondly, the kind of countermeasures that are needed to defend against availability attacks are very different, and would require a significantly more complex solution. Finally availability attacks are inherently difficult to defend against because fundamentally the grid must provide a service, and there is always a danger it could be overwhelmed with requests given an attacker with sufficient resources.

Security is not the only consideration in a grid. If it was, the best solution would be to go back and completely re-engineer existing grid middleware software to enhance the security it offers. Making such fundamental changes is unacceptable because it breaks interoperability with current systems. It is vital to maintain backwards compatibility with established solutions that have been developed over many years. It would be very difficult to gain wide support for a new security architecture that does not interoperate with existing solutions.

As grids grow, interoperability issues become progressively more important. As the number of organisations increases there will inevitably be a proliferation of different grid middleware solutions. Web services and similar technologies help overcome differences in the underlying implementation. These differences, however, are important for security. The quality of the implementation and the architectural design of the components make the difference between a robust grid infrastructure and one that contains too many vulnerabilities to be trusted.

If these interoperability problems are not solved there is a danger that the vision of a global grid will fail. If service providers are forced to choose a particular security solution it is likely the grid will become segmented into communities with different security requirements. An organisation hoping to join the grid would be forced to either use a software solution trusted by all the other members, or be isolated from the community. These interoperability problems threaten to undermine the key advantage of grid technology – encouraging collaboration between organisations on an unprecedented scale.

These problems motivate why interoperability is one of the key goals for a trusted grid architecture. The thesis explores how to build a trusted grid that is interoperable with existing middleware software, but does not share any of its security weaknesses. This flexibility would permit a large grid community to collaborate using a common, trusted security solution despite differences in the underlying middleware infrastructure. Solving this problem is essential if a large-scale trusted grid is to become feasible.

A second issue for interoperability is compatibility with existing grid jobs. As will be shown in Chapter 4, several existing solutions require users to add a security layer to their software before dispatching it onto the grid. Support for legacy grid jobs is useful because companies often have software that does not run on the latest programming and operating-system platforms. The source code for these applications may no longer be available to make changes, and adapting the application to support the trusted grid would be costly.

There is sometimes a case for security measures that are not visible to users. The idea behind the trusted grid architecture is that users can create and submit grid jobs, and gain all the benefits of the trusted grid, without needing to be aware of it. Users should be

able to make use of their existing software on the trusted grid without the need to make complex changes to the code or seek the help of security experts. As a result users do not need to be trained in security in order to create trusted software.

In summary, the interoperability requirements for a trusted grid are as follows:

Interoperability goals

1. **Support existing grid middleware** No changes to the grid middleware or other service provider systems are necessary to support the trusted grid.

2. **Support legacy grid jobs** Trusted grid support is available to unmodified grid jobs (e.g. legacy code).

1.4 Thesis structure

A wide range of security goals have been identified in the previous section. One way to approach this problem would be to design and implement a small part of the solution. However research in this area is still very much ongoing. As will be demonstrated in later chapters, the current state of the art solutions still have a long way to go before they will meet the requirements for a trusted grid.

Instead, the approach taken in the thesis is to concentrate on developing a high-level grid security architecture. This helps demonstrate the 'big picture' – how many components can work together to create a robust set of security properties. Developing the architecture first shows what might be possible in the future and motivates the need for more detailed work to develop specific components of the solution.

Chapter 2 introduces the concepts necessary to understand the rest of the thesis. Chapter 3 contains a review of existing work in the area of building trusted grid systems.

There is not one single solution to the trusted grid problem. As has already been seen in the example scenarios, there are a range of different security properties that might be required depending on the use-case scenario. There are, however, a set of related security components and design principles that may be combined to produce a security architecture. The core chapters of the thesis (Chapters 4 to 6) present the detailed concepts and

components necessary to build a trusted grid.

Chapter 4 describes how to solve the interoperability problem by decoupling the security layer from the grid middleware. Chapter 5 presents a solution to verifying software integrity in dynamic environments such as those found in public resource computing projects. Chapter 6 shows how to provide integrity and confidentiality protection for grid jobs, as well as enforce digital rights management and mandatory access controls.

The next two chapters are primarily concerned with the feasibility of the proposed new architecture. Chapter 7 considers how the trusted grid could be implemented. Since current technology is not yet at a sufficiently advanced stage, alternative implementations are considered in the final section to show how a solution could be achieved in the near future. Chapter 8 evaluates the success of the solution in terms of security and performance.

Finally the conclusion (Chapter 9) returns to the goals for the trusted grid set out in this introduction, and considers the extent to which they have been achieved, as well as considering potential future work.

2 Trusted Grid Concepts

2.1 What is The Grid?

The Grid is a term that has been recently popularised as part of a resurgence of interest in distributed computing. A grid coordinates the sharing of resources across multiple organisations to create the illusion of a giant super-computer. The distributed nature of the grid means that users can collaborate over a geographically distributed space with a high quality of service. The community that works together to provide and consume grid resources to complete a task of mutual interest is known as a *virtual organisation*.

The huge potential of harnessing computing resources across the globe has led to the development of large-scale grid systems. This is evident in the creation of nationwide grid services such as the TeraGrid project in the United States [38], the UK National Grid Service [58] and the European Union DEISA project [93]. These grids enable multi-disciplinary research organisations to share resources such as databases storing petabytes of data, scientific devices and supercomputing clusters over high-speed fibre-optic networks.

It is hoped that grids will help solve so-called grand-challenge science problems that require collaboration and the use of computing resources on an unprecedented scale. Users are drawn from many disciplines such as physics, engineering and bioinformatics. Grids have been used to model the origins of the universe [90], the blood flow in the human body [38], and to predict future climate change [159] and earthquakes [182]. The book edited by Foster and Kesselman provides numerous additional examples of grid applications [47].

Grids are typically driven by middleware software that provides a widely used set of core services (see Figure 4). Many different grid middleware implementations exist, for example Legion [161], Globus [50] and UNICORE [158]. Most middleware systems provide similar high-level services including:

- **Execution management services** that manage the allocation, execution and termination of grid jobs.
- **Information services** that provide dynamic information about the data and resources available throughout the grid. This information is used for various purposes

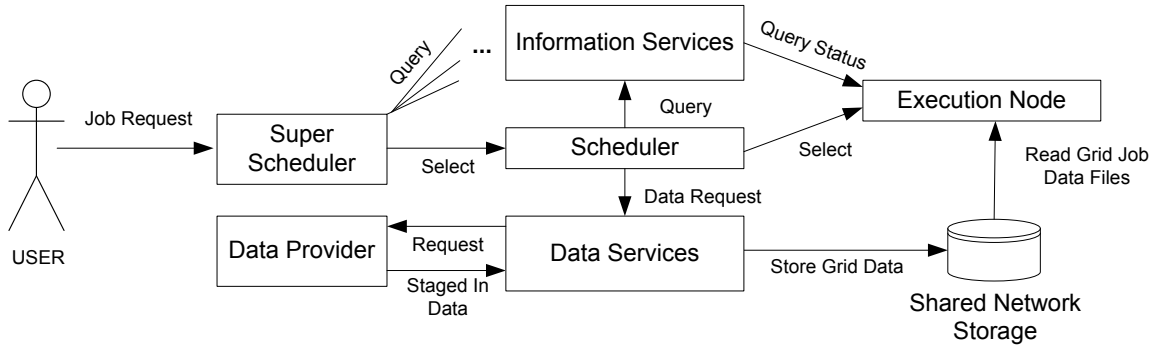

Figure 4: A Typical Grid Middleware Architecture

such as scheduling jobs and locating suitable data-sets.

- **Data management services** efficiently and reliably transfer large volumes of data from a variety of data sources for consumption by grid jobs, as well as retrieving and cleaning up the grid job results.

- **Security services** provide single sign-on, authorisation and delegation for grid users. Grid security services are described more fully in the following section.

Researchers have asked exactly what is the grid, and what sets it apart from traditional distributed systems [46]? Many architectures have similar properties including cloud computing, peer-to-peer computing, public resource computing and others. Ian Foster, a prominent advocate of the grid, answers that standardisation is its defining novel contribution [49]. The Open Grid Services Architecture [48], now developed by the Open Grid Forum consortium, defines a set of open standards for grid services. Standardisation is essential to facilitate interoperability between different middleware and service implementations in a large-scale grid.

There are many different types of grids, and equally many ways that users can interact with them. A common approach is similar to a batch system where the user dispatches a *grid job* and waits for the results to be retrieved (possibly while the user is offline). A more interactive approach would allow the user to connect to an executing job to alter its computation based on partial results. Users can interact with the grid through a *portal* service that gives them a user-interface onto the grid, for example through a web browser.

A distinction is sometimes made between grid systems. A *computational grid* focuses on giving users access to high-performance computing resources. A *data grid* focuses more on giving users access to large-scale data-sets [27]. Data grids help users to efficiently access huge quantities of data that might previously have been isolated in different legacy storage systems with no standard, central mechanism for accessing them.

Schedulers allocate grid jobs to the available resources within an organisation. *Super-schedulers* communicate with local schedulers throughout the virtual organisation to select the most appropriate location for grid job execution. The schedulers ensure high availability for grid jobs because if an execution node fails the job can be rescheduled and executed again. The same strategy can be used to *migrate* an executing grid job onto a less heavily loaded node.

The data management services can efficiently transfer large quantities of data. Data can be replicated to improve availability and to increase throughput by accessing a nearby copy of the data. Another advantage is that throughput can be improved by downloading the same replica from multiple locations simultaneously.

Data staging services transfer grid data when the associated job is not currently executing. Grid job data can be *staged in* prior to execution to ensure resources are not tied up during the data transfer. Grid job results can also be *staged out* during or after job execution.

2.2 Grid Security

A grid security architecture must overcome some difficult challenges. Grids implement many widely accepted security solutions such as user authentication and authorisation. Some interesting changes and additional security controls are necessary, however, to deal with a large-scale and dynamic distributed environment.

Digital certificates are a common approach to user authentication on the grid [50]. Certificates allow users to authenticate themselves without giving out a long-term secret like a password to a potentially untrustworthy remote host. Instead the user keeps a long-term private key. The corresponding public key is contained in a certificate signed by a trusted

certificate authority. The user authenticates by providing their certificate and proving that they possess the corresponding private key without giving it out to the remote system.

The user must be authenticated many times in a typical grid job life-cycle. Multiple schedulers, information services and data servers might be contacted prior to grid job execution. Once the job starts running it might need to dispatch further grid jobs or access additional data. Each of these operations requires user authentication in order to authorise access.

There is a significant problem in grid authentication because the user will not typically remain online throughout the lifetime of the grid job. Instead, the grid services must authenticate on the user's behalf. To facilitate this the user *delegates* their authority to a resource. Usually chained delegation is allowed, so that a service can recursively authenticate using the identity of the user. Delegation permits *single sign-on*, allowing grid jobs to run successfully while the user is offline.

There is a danger that delegated authority might be mis-used without the user's consent. A popular solution involves the use of *proxy certificates* [50, 178]. The remote resource is given a time-limited delegated credential that can only be used for the life-time of the grid job. Proxy certificates are created by having the user digitally sign a certificate with a short expiry date using their long-term private key. A potentially complementary solution is the use of *restricted delegation* [161]. In this approach the user only permits the credential to perform a limited set of actions throughout the delegation chain.

A potential problem with certificate-based authentication is a lack of portability. The user can only authenticate from a computer that has access to the user's long-term private key. If the user is travelling with a laptop, or using an Internet cafe, for example, their private key may not be available. One solution is to use a grid *credential repository* [116]. The user can retrieve a short-term proxy credential from the repository by authenticating themselves using a user name and password. The proxy credential can then be used to access the grid.

Host-based authentication is another security control that can be useful on the grid [50]. Each host is assigned a private key and a digital certificate. During delegation the delegator

authenticates the host certificate of the delegatee, and vice versa. The virtual organisation will typically trust a limited set of certificate authorities to certify private keys belonging to legitimate hosts. Host-based authentication helps ensure that user credentials are not delegated to a rogue machine outside the virtual organisation.

Authorisation systems are used to control users' access to grid services following authentication. The access control architecture can be split into two components. The *Policy Enforcement Point* (PEP) is responsible for allowing or denying access. The PEP refers to a *Policy decision point* (PDP) which makes the access control decision. This architecture improves design modularity as well as interoperability with multiple types of devices. For example the PEP can be re-implemented on a variety of gateway devices that all connect to the same PDP.

Grid authorisation systems must meet a number of complex requirements that would not be found in many other environments. Resources are spread across administrative domains, raising issues over whether access control policies should be controlled centrally or by each member of the virtual organisation. The available resources can alter dynamically as users and services join or leave the grid. Entire virtual organisations might be created and disbanded rapidly for specialised projects. Many different authorisation and federated identity systems have been designed to meet some of these constraints including CAS [120], VOMS [3] and Shibboleth [153].

2.3 Trusted Computing

2.3.1 Trusted Computing and the Grid

Trusted computing refers to a range of technologies that seek to significantly improve the trustworthiness of mainstream computing platforms. Huge numbers of viruses, Trojan horses and other malicious software are released every year that target security vulnerabilities in widely used software [172]. Many members of the security research community believe that commodity software and operating-systems currently depended upon by millions of computer users are too complex to reliably protect valuable information [54, 99, 131, 137].

Trusted computing seeks to redress this balance, enhancing the trustworthiness of soft-

ware by underpinning it with hardware-enforced security mechanisms. Trusted computing is an attractive solution for grid security problems because the security features it provides match very well with grid security requirements.

Attestation offers a potential solution to the grid malicious host problem because it allows the user to determine that platforms beyond their physical control are in a trustworthy state. Consider a grid job that is deployed to a remote platform. When the user retrieves results the grid job can be attested to make sure that it ran free from integrity attacks using the correct code and data.

A second capability provided by trusted computing that is well-matched to grid security requirements is called *protected storage*. Protected storage allows a user to protect grid job data from software-based attacks when it is run on a remote platform by using encryption. Protected storage can also be used to create a digital rights management system where even the owner of the platform is not able to reveal the encrypted information.

Virtualisation is an important supporting technology for trusted computing. It allows a single computer to run multiple operating-systems. Each operating-system can be run in a compartment where it is strongly isolated from other software on the system. In a grid, virtualisation can be used to run each grid job in its own compartment where it is strongly protected from other users that are sharing the same platform.

2.3.2 Trusted Platform Module

The *Trusted Platform Module* (TPM) is the central component of trusted computing. The TPM is a hardware chip embedded in increasing numbers of PC platforms: nearly 150 million units were shipped in 2008 and the estimate for 2009 is 250 million units [129]. Trusted computing can offer a significant improvement over existing systems because the security critical functions are implemented in hardware where they cannot easily be affected by a software-based attack.

The functionality provided by the TPM is defined in specifications published by an industry-led consortium called the *Trusted Computing Group* (TCG) [170]. A TPM chip must be capable of performing cryptographic signing and hashing operations. To reduce

costs, support for symmetric encryption of bulk data is not required. In addition, the TPM can store a limited number of secrets and other data that are protected from software-based access.

To understand this thesis it is vital that the reader has a good understanding of the key trusted computing functions. The remainder of this section gives a broad overview. Several good resources are available that provide a more thorough description of the trusted platform module and related technologies [11, 105, 110]. A readable technical overview of virtualisation is given in the book by Smith and Nair [154].

2.3.3 Authenticated Boot

Authenticated boot aims to ensure the platform reaches a known state. Normally when we interact with a computer it is hard to know whether the software it is running can be trusted. The software may appear to be genuine when it in fact contains a Trojan horse. Trusted computing allows the process of booting a platform to be *measured* so that its current state cannot be mis-represented by malicious software. If the system has been subverted this will be evident because the state measurement will have altered from the expected value. The user can then decide whether or not they trust the measurement before continuing to interact with the platform.

The TCG specification does not define exactly what should be measured. This decision is left to the programmer. A measurement is a fixed length digest computed from a one-way hash of a potentially large amount of arbitrary data. In practice, the authenticated boot measurements often include the executable code involved in the boot process and critical data configuration files [142].

The TPM provides a number of shielded locations called *platform configuration registers* (PCRs) where the measurements can be stored without being tampered with. Each time a measurement is stored, the TPM hashes the concatenation of the old and new measurement values. This use of the hash function ensures that malicious software cannot set a PCR to a specific value, or roll back to a previous value, in order to spoof a valid measurement. The use of recursive hashing allows a single PCR to hold an arbitrary number of measurements.

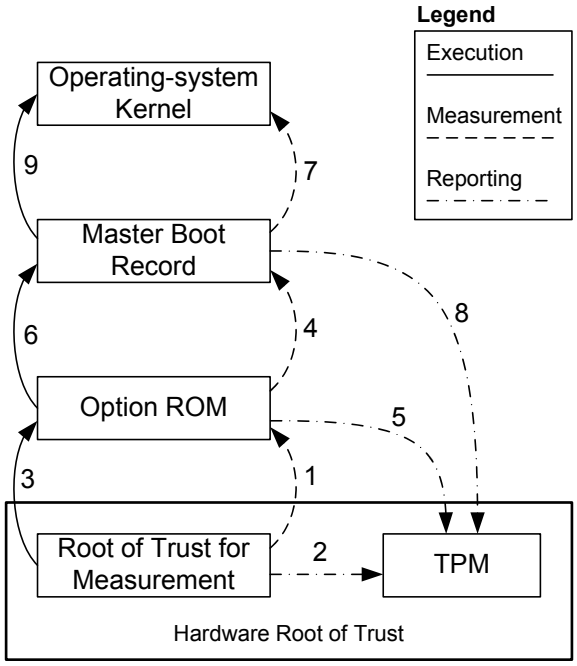

Figure 5: Trusted Computing Authenticated Boot Process

In PC platforms the boot process consists of several stages (see Figure 5). The BIOS is first to execute when a PC is powered on. Subsequent stages include the option ROMs belonging to hardware devices, the master boot record on the hard-drive, followed by the operating system kernel which then loads the rest of the operating-system. TCG specifications require that measurements of each stage in the boot process are stored in separate PCRs. After the boot process is complete, further PCRs are available to measure application software and other data.

Each stage in the boot process is responsible for measuring the software involved in the subsequent stage before it is executed. The very first measurement is performed by a trusted, immutable hardware component called the *root of trust for measurement* (typically this is part of the BIOS). This process creates a transitive trust chain proving that all the measurements are genuine. If one of the stages in the boot process is subverted with a Trojan horse, this would be revealed in the measurement performed by the previous stage.

Intel recently added support for a feature called *late launch* that eliminates the need to measure the BIOS and other firmware during the trusted computing authenticated boot

process, in order to simplify attestation measurements [62].

2.3.4 Remote Attestation

Remote attestation is a powerful security primitive that allows a party to obtain assurance in the correct operation of a remote system that is beyond their physical control. During attestation the measurements stored in the TPM are passed to the requester to prove the state of the remote platform. The requester must be convinced that the measurements were produced by a trusted platform containing a valid TPM. This proof is provided using public-key cryptography.

A unique private *endorsement key pair* is stored within each TPM. A suitable entity, such as a PC manufacturer, vouches that the corresponding platform conforms to TCG specifications. The manufacturer does this by signing a certificate containing the public half of the endorsement key. To prove that measurements were produced within a valid TPM they can be digitally signed by the private endorsement key. The requester can then verify that this private key corresponds to a valid TPM by examining the certificate.

Privacy issues are an important consideration in trusted computing. There are concerns that attesting a platform multiple times using the endorsement key could lead to the ability to track individual users [11]. Instead the measurements are signed using an *attestation identity key* (AIK). A single user can generate an unlimited number of AIKs to provide pseudonymity.

A mechanism is needed to prove that an AIK was generated on a trusted platform containing a valid TPM. A digital certificate of the AIK signed by the endorsement key cannot be presented directly during attestation without causing privacy concerns. Instead this digital certificate can be presented to a trusted third-party, known as a Privacy-CA [110]. The Privacy-CA can then issue a digital certificate for an AIK public key, which specifies details of the owning platform without uniquely identifying it. The certificate issued by the Privacy-CA can then be used to perform attestation based on a pseudonymous AIK, without any way for the involved parties to link it back to the identity of the TPM that produced it.

One disadvantage of this approach is that the Privacy-CA must still be trusted to protect users' privacy. If a Privacy-CA is compromised the attacker can discover which AIKs belong to the same, unique platform. An alternative scheme that eliminates the need for a trusted third party is known as Direct Anonymous Attestation [24].

Despite some of the complexities behind the cryptography implementing remote attestation, its aim is simple. It provides proof that a platform is executing a given set of software. The additional mechanisms prevent this proof being falsified by a malicious platform or third-party.

2.3.5 Protected Storage

Protected storage allows access to cryptographic keys to be controlled by the TPM. The keys cannot be accessed directly by software. Instead the key is loaded into the TPM, where it is decrypted. The plaintext key is therefore never revealed in memory where it might be stolen by malicious software. In this way protected storage is somewhat similar in concept to storage of a private key on a smartcard.

Authorisation to access keys is based on knowledge of an *authorisation secret*. This is an arbitrary hash value that could be based on various data such as a PIN number or password.

Protected storage keys can also be *sealed*. Sealing a key ensures it can only be accessed at a later time if the platform is in the state nominated when it was originally sealed. This is done by storing a target set of platform configuration registers (PCRs) alongside the private key. During the unseal operation the TPM will check that the current set of PCRs matches the expected values before allowing decryption to take place. This is a powerful feature because it means that access to keys can be restricted to attested applications.

The keys are stored in a tree-like structure (see Figure 6). Each child key is encrypted with the parent key. Access to the parent key must be authorised by the TPM before the child key can be decrypted. This tree structure lends itself to defining roles where users (or software processes) of different privilege levels have access to different parts of the hierarchy.

The TPM only has limited storage space for keys. Instead protected storage keys are

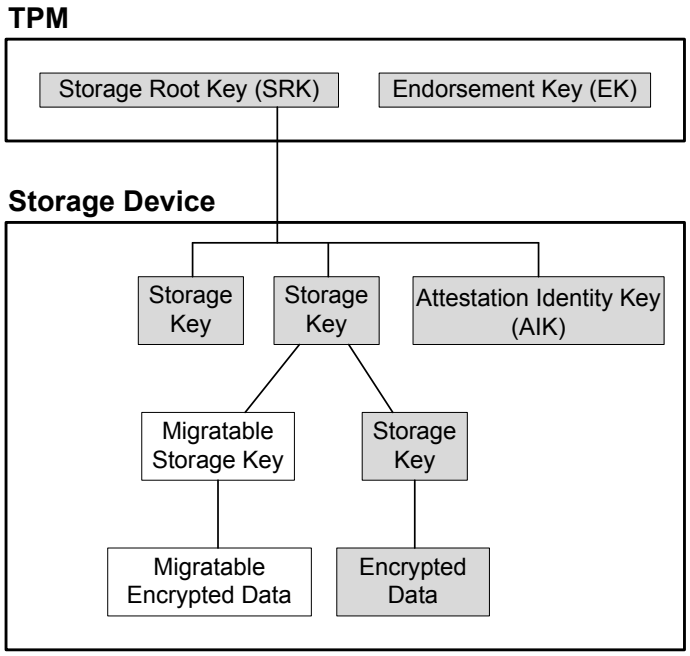

Figure 6: Example Trusted Computing Protected Storage Key Hierarchy

held in encrypted form on secondary storage. Only the root of the tree, the so-called *storage root key*, needs to be permanently kept within the TPM itself. Keys further down the hierarchy are accessed by loading each key in turn into the TPM on the path down to the chosen node.

Protected storage keys can be used for a variety of purposes. For example private keys belonging to users' grid credentials can be stored securely within the TPM where they cannot be accessed directly by Trojan horse software. Symmetric encryption is somewhat more difficult because the TPM cannot perform these operations on-chip. However asymmetric protected storage keys can be used to protect symmetric keys, that can in turn be used by software.

There is sometimes a need to access a protected storage key on another platform. For example data encrypted under the key might be transferred to another computer on the grid. For this purpose the TCG allow the creation of *migratable keys*. These special keys can be decrypted outside the TPM, where they can be directly accessed by software. Migratable keys are inherently less trustworthy than non-migratable keys because there is no guarantee

they have not been compromised.

Later versions of the TPM include support for a *monotonic counter* that is guaranteed to count upwards. This counter can be used for various purposes including trusted timestamps, and digital rights management controls to restrict the number of times data such as video and music can be accessed. Since the counter is implemented in hardware, there is no way to bypass security controls by rolling back the state of the system to a previous point in time.

2.3.6 Virtualisation

Virtualisation allows a single computer to share its CPU and other resources between multiple operating-systems [61]. Operating-systems normally expect to have dedicated access to hardware. To allow multiple operating-systems to run each is executed within a *virtual machine* where it is given the illusion of dedicated hardware access (sometimes called a *guest operating-system*).

A *virtual machine monitor* (VMM) mediates all access to the physical hardware resources. Virtual machines each run on a set of virtual hardware. The VMM intercepts all access to the virtual hardware and passes the requests onto the physical hardware.

An *isolation kernel* is a name for a common type of VMM that strives to ensure that no interaction is possible between any virtual machines on the system. Each virtual machine is given its own dedicated memory and disk space. An isolation kernel attempts to make virtual machines behave identically to physically separate computer systems. Isolated virtual machines are often referred to as *compartments*.

Normally an operating-system maintains isolation between user accounts on the system. A typical operating-system contains millions of lines of code that must be trusted to perform this separation [137]. A much stronger degree of isolation can be achieved by running grid jobs in a separate virtual machine. The only trusted software component is the virtual machine monitor, which is typically far smaller, on the order of tens of thousands of lines of code [137, 71].

One incentive to use virtualisation is the introduction of native hardware support in all

recent AMD and Intel processors [162, 62]. These improvements to the CPU architecture are designed to speed up the execution of virtual machines by minimising the overhead of virtualisation.

Recent improvements to hardware virtualisation are intended to work in partnership with trusted computing technologies. Intel is following a road-map that will see significant changes made to the PC motherboard chipset and devices. One goal is to add support for new, secure I/O devices by creating an encrypted channel between devices such as displays, keyboards and hard-disks [62, 162, 169]. Another goal is to help improve the assurance of mainstream operating-systems by eliminating the need for trusted device drivers [1]. These improvements rely on virtualisation technology to run trusted code in a virtual machine compartment where it is isolated from the rest of the system.

2.3.7 Virtual TPMs

The TPM has been designed by TCG as a chip that is bound to a host hardware platform. In a virtualised environment, however, a single host can execute many virtual machines, each of which may require the use of a TPM. A *virtual TPM* [15] (VTPM) is a virtualised abstraction of a real TPM designed to give virtual machines the illusion that they have direct, dedicated access to the physical TPM residing on the host platform.

Virtual TPMs have been implemented in Xen, a well-known open-source hypervisor that has been commercialised by Citrix [12]. In this approach the VTPM is implemented in software. Each virtual machine is given access to a fresh instance of the software VTPM. The VTPM process runs within the host platform where the virtual machine monitor protects the data it contains from direct access or modification by potentially malicious software running in a virtual machine.

Some mechanism is needed to extend the trust chain from a VTPM back to its host physical TPM. Without this link there would be no basis for trusting attestation measurements produced by the VTPM, since there would be no hardware root of trust. Berger et al. suggest a number of alternative approaches for extending the trust chain [15]. In one alternative, the attestation identity key (AIK) of the host's TPM is used to digitally sign

the endorsement key (EK) of the VTPM. This allows a challenger to verify an attestation measurement produced by a VTPM by checking there is a valid certificate chain leading up to the AIK of the host's physical TPM, which might itself be certified by a Privacy-CA (see Section 2.3.4).

An attestation measurement consisting of the state of the virtual machine alone is insufficient. The virtual machine monitor (VMM) together with the VTPM and other software running on the host platform is also part of the trusted computing base. Software running on the host platform is privileged to modify the behaviour of all running virtual machines, so it must be included in the attestation measurement. Berger et al. [15] again suggest a solution where the measurement of the VMM and host platform is copied into the platform configuration registers of the VTPM. This allows an integrity challenger to verify the state of both the virtual machine and the host platform during attestation.

2.4 Virtualisation and the Grid

Many researchers have discussed the benefits of using virtualisation for grid jobs [45, 83, 88, 147, 183]. The central idea is that each grid job is deployed as an entire operating-system, together with grid applications and data, running in a virtual machine.

Researchers have made a good case for grid computing using virtual machines, suggesting advantages including the following [45]:

- The user has complete flexibility in their choice of operating-system and applications.
- Virtual machines are portable to different underlying hardware architectures without changes, provided a suitable virtual machine monitor is available.
- The virtual machine monitor can easily control the resources made available to grid jobs, such as the amount of memory and storage space.
- Security is improved, in terms of the separation between grid jobs running on the same platform.

The Globus virtual workspaces project [84] (recently renamed Nimbus) adapts the widely used Globus middleware software to use virtual machines. Instead of transferring the

individual files making up the grid job, the data management services are used to download entire virtual machine image files. The grid execution management layer is adapted to add support for controlling virtual machines instead of software processes.

One problem with running grid jobs as virtual machines is the large size of virtual machine images. As well as the grid job itself, each virtual machine contains an entire virtual operating-system on which the grid job is executed. This adds significantly to the storage overheads for each grid job. Since the same operating-system is likely to be shared by many grid jobs, a distributed caching filesystem is an attractive option [183]. This filesystem can cache disk blocks that are shared between multiple virtual machines to reduce the storage and data transfer requirements.

Cambridge researchers described in the Xenoservers project [130] how virtualisation can be used to replicate jobs on a distributed system in order to meet performance requirements. The example given is a web server running in a virtual machine. As the load on the website increases, the number of web servers can be increased simply by starting new virtual machines to accommodate the load. Another feature of modern VMMs allows live virtual machines to be migrated onto another physical platform [29]. If one grid server becomes overloaded it would be possible to dynamically move the grid job onto another server with a lower utilisation [134].

It is an assumption made throughout this thesis that virtual machines will become the standard mechanism to deploy grid jobs. The security architecture that is developed relies on the enhanced isolation provided by running grid jobs in virtual machine compartments.

2.5 Assumptions

The thesis builds on the work performed by many other researchers. Where possible the thesis re-uses existing solutions, and focuses instead on unsolved problems. This section considers some of the underlying assumptions made throughout the following chapters.

It is assumed that grid software will be based upon re-usable middleware. Since many grids share similar requirements and functionality, it makes sense to build specialised grid software on top of re-usable core components. Although it is possible to build bespoke

services for isolated communities, this idea is not compatible with the idea of a global grid. The thesis focuses on how security could be deployed successfully on a large-scale grid consisting of many diverse members.

Virtualisation is likely to become one of the key building blocks for effective grid systems. This is because virtualisation provides many properties that are an ideal match to grid requirements, such as secure grid job isolation, as many researchers have observed (see Section 2.4). The remainder of the thesis assumes that the general problem of deploying grid jobs as virtual machines has already been solved — indeed there are already widely publicised solutions such as the Globus Nimbus project [84] that achieve this.

Combining trusted computing with virtualisation creates a number of difficulties. A key problem is that the TPM is no longer physically bound to the platform, since a virtual machine can be migrated between systems. In addition, there is a need for a means to share the single physical TPM in the platform between multiple virtual machines. This is a well known problem that has already had considerable research effort placed into it (see Section 2.3.7). For the purposes of the thesis it will be assumed that this problem is already solved.

Secure storage services are already well developed. The trusted grid solution in the thesis will rely on suitable algorithms for performing high speed disk encryption and integrity protection. This is a mainstream technology that has already been deployed in popular software such as Microsoft Windows (see Section 4.5.2 further details)

The TCG define the TPM and the mechanisms needed to support attestation. They do not, however, provide the detailed protocols and technology for performing attestation. Since this is a generic problem needed to apply trusted computing to real-world scenarios, it has already been examined in detail by the research community (see Section 3.2). As a result, the thesis assumes that a generic attestation challenge and response protocol is already available, and focuses instead on what would be measured.

2.6 Conclusion

A large research community has formed with the common goal of creating the components that go towards the construction of a global grid. A cornerstone to this work is grid mid-

dleware software, which drives the common functionality needed across many applications. There is a drive towards standardisation of middleware software to open up the grid to potentially thousands of geographically dispersed research and commercial organisations, allowing them to work together for their common benefit.

Difficult security challenges must be met before a global grid can be realised. Trusted computing appears to offer a well matched set of functionality to meet these requirements. One of its key functions, attestation, allows the software state of a remote system to be measured. This exciting security functionality could help tackle the grid malicious host problem by attesting a grid execution node prior to job execution.

The next chapter goes on to examine the state of the art in research on the use of trusted computing to solve complex security problems in distributed systems, as well as alternative solutions.

3 State of the Art

3.1 Introduction

The purpose of this chapter is to present the state of the art in research related to the topic of the thesis. It is important to understand the context in which this work has been undertaken and how it builds, as well as improves, upon existing research.

Trusted computing attestation offers a number of exciting ways to enhance the security of grid systems. Attestation is still a subject of intense academic research in order to solve various security and feasibility problems. This research is described in Section 3.2.

The following two sections describe research on the malicious code and host problem. Preventing the execution of malicious code on grids is important to protect users against attacks by other users of the system. A detailed review of existing solutions to the malicious host problem is important because this is the primary subject of the thesis. The malicious code problem is strongly related to the malicious host problem, because a Trojan horse installed in the host platform is in an excellent position to attack grid jobs belonging to users of the system.

The chapter concludes with a review of the wider research on applications of trusted computing (Section 3.5).

3.2 Attestation

3.2.1 The Time-of-use vs. Time-of-attestation Problem

Remote attestation is a powerful mechanism that introduces an exciting new security primitive. The trusted computing community has invested considerable effort in addressing concerns with both the manageability and flexibility of remote attestation, as well as enhancing the security guarantees it provides.

The idea of remote attestation pre-dates work on the trusted computing TPM. Lampson et al. are often credited with devising the concept of attestation in their work on distributed systems [91]. These early ideas were made more concrete by Arbaugh et al. who define the exact mechanism for attestation using trusted hardware [10].

Other work that pre-dates the TPM includes the IBM 4758 series of secure co-processors [157]. The IBM co-processor supports a mechanism called *outbound authentication* that is similar to attestation [156]. Unlike the TPM, the IBM 4758 runs attested code on a dedicated tamper resistant hardware module containing a co-processor and embedded firmware. The IBM 4758 includes a facility to perform remote attestation by measuring the entire contents of the firmware chip and signing that measurement with a private key analogous to the attestation identity key used by the TPM.

The IBM 4758 design side-steps a crucial problem with attestation that has been dubbed the problem with *time-of-use versus time-of-attestation* [151]. To reduce cost, the TPM relies on the computer's main memory to run attested applications. While an application is running, it is open to attack from a Trojan horse that might alter its behaviour by overwriting its program code in memory, or steal sensitive information processed by the application. More expensive solutions, such as the IBM 4758 co-processor, solve this problem by running attested applications on dedicated memory physically isolated from the host platform.

The BIND research prototype [151] attests software in memory at the point of execution, eliminating the gap between time-of-use and time-of-attestation. Attested code is executed in a security kernel that is responsible for protecting the attested code from tampering by external software processes during execution. The security kernel uses the TPM to attest a hash of the attested code and input data, and the related output data. Since the attestation measurement is produced at the point of execution, a recipient can verify the exact code that was executed to produce the output data. Tampering with the memory of the attested code can be detected because the hash will no longer match the expected value.

The design of BIND is similar to AEGIS, which takes a more hardware-oriented approach [164] involving modifications to the central processing unit. An AEGIS CPU includes a special instruction that is called by applications to initiate a switch into trusted execution mode. A trusted operating-system then measures a contiguous area of memory representing the application, and uses that value as the run-time integrity measurement.

3.2.2 Determining an Identity for Software

There are fundamental problems in determining a static notion of identity for software. Software is usually highly dynamic, with a constantly changing set of data, and a physical memory layout that is controlled by the operating-system to accommodate concurrently executing processes.

Rather than deal with this complexity, trusted computing attestation is usually designed to measure the software on disk before it is loaded into memory. This does not completely overcome the identity problem, however, because once software is deployed, configuration parameters are altered, temporary files are written to disk, and the software might be continually patched and updated with revisions. During execution, the states of most applications, and the operating-system they run on, are constantly changing, leading to problems pinning down any static notion of identity for attestation.

Several researchers have suggested solving the identity problem by attesting only the binary or relatively static parts of a platform [151, 143]. Since these components change less frequently, the attestation identity should remain consistent over time, only changing when software patches or significant configuration changes are introduced. It is easy to see the attractiveness of this solution, since it vastly improves the usability of attestation and, if designed carefully, the unattested components should have a minimal impact on security.

Assume, for example, that a bank wants to ensure customers are using a safe browser to access Internet banking. Consider the temporary cache of web pages stored by a web browser to speed up viewing. The browser cache could reasonably be omitted from the attestation measurement because it should not affect its behaviour. On the other hand, the bank would be remiss not to include the security policy of the web browser in its attestation measurements, as an incorrect policy will facilitate infection by malicious code.

Microsoft, in their proposed Next-Generation Secure Computing Base (NGSCB) platform [122], requires developers to separate software into trusted and untrusted components. The untrusted component is the host program and the (hopefully small) attested component is called an agent. Agents are compiled to run on a trusted operating-system also created by Microsoft. The host application runs in a standard Windows operating-system and

communicates with the agent via a programming interface. For example, an NGSCB agent in the context of an email application might be a trusted viewer ensuring that the content of emails cannot be stolen or tampered with by untrusted software. Unlike BIND, NGSCB does not attest agents in memory during execution, and relies on secure programming to prevent attacks.

IBM take a more pragmatic approach in their Integrity Measurement Architecture (IMA) [143] by adding incremental support for attestation on an existing Linux operating-system. In IMA, the Linux kernel is modified to automatically measure every binary executable and linked library on the system immediately prior to execution. Each measurement results in a fresh attestation measurement being stored in the Trusted Platform Module, as the hash of the loaded file. Alongside this measurement a log is kept of all files loaded on the system and the corresponding hash, allowing a user to verify the current state of the platform given an attestation measurement. The attested software is then responsible for measuring further static content, such as configuration files, by itself. Unlike BIND and NGSCB, IMA does not attempt to protect against attacks on the attested code by using a trusted operating-system, with the advantage of maintaining compatibility with existing applications.

Terra [54] implements a solution for attestation where virtual machines are transparently attested by a trusted component that exists outside of the virtual machine itself. In Terra each application is deployed along with its own operating-system in a self-contained virtual machine that is controlled by a trusted virtual machine monitor (VMM). Applications can access their attestation measurement by performing a special system call that is intercepted by the VMM. Terra allows great flexibility because it does not require any modification to the operating-system unlike the IBM integrity measurement architecture. It is compatible with any operating-system, whether open- or closed-source.

Like other approaches, Terra allows applications to be split into a trusted, attested component that is relatively static, and a more changeable, unattested part. Terra takes advantage of the fact that virtual machines are typically stored in a series of large files, each of which holds an entire virtual hard drive partition. The virtual machine monitor

implements attestation by attesting hashes of virtual machine image files. Applications must be designed so that all attested files exist in the same physical drive partition, and VM disk files are marked either attested or unattested. The unattested virtual machine storage partitions are used to store dynamic data.

Researchers from Dartmouth College [103] suggest dividing software into three levels of classification for the purposes of attestation. Long-lived data can be attested in its entirety because it is relatively static. Examples of long-lived software include the operating-system kernel and the system BIOS firmware. Medium-term data includes application software. The enforcer, a long-lived component, is responsible for measuring and attesting medium-term data. Finally, short-lived data including user content, such as web pages or emails, are unattested. All secrets in the system are sealed to the long-lived components, avoiding the need to re-seal data when applications are patched or updated

Haldar et al. makes the important point that attestation proves the identity of a system, but not its behaviour [63]. What is often sought instead of identity is a guarantee that a system will abide by a particular security policy. Identity-based attestation leaves the user to infer the behaviour of the system, which itself relies on trust in the developers of the attested software. This has led researchers to develop an approach where attestation deals with properties, or the behaviour of the system, rather than its identity [63, 136].

Haldar et al. [63] have devised a new approach they term *semantic remote attestation*. Instead of identity, attestation is used to measure a security policy that proves aspects of the operation of software running on the system. The implementation takes advantage of the fact that the Java Virtual Machine imposes a security policy on Java applications [154]. The virtual machine can control access to program variables and enforce type safety. The Java Virtual Machine attests security policy meta-data that is encoded in the Java applications. Unlike other approaches such as BIND [151] and the IBM IMA [143], semantic remote attestation imposes behavioural constraints throughout the lifetime of a dynamic, measured application.

3.3 Malicious Code and the Grid

Grids by their very nature offer a platform for users to run their own software. This makes grids particularly susceptible to the introduction of malicious code. Since many users execute code on a shared infrastructure, mechanisms are needed to separate users from each other to ensure they cannot compromise each others' data. In a public-resource computing scenario a malicious grid job might be used to attack data belonging to the owner of the host computer.

Users on the grid are authenticated so there is a level of trust in their good behaviour. However as the grid grows in size and becomes used increasingly for commercial purposes it will no longer be feasible to trust all the users on the grid. This has led to research into a variety of ways to sandbox malicious code, or build trusted operating-systems that control access to sensitive data.

3.3.1 Mobile Agents

Research on mobile agents has addressed the problem of both malicious code and malicious hosts (considered in Section 3.4). A mobile agent is an autonomous piece of computer software than can move between computer systems in order to perform its tasks [77]. A useful example is given by Yee [181], where he describes mobile agents used to collect airline fares that must deal with a dual security problem: a malicious host could subvert the fares quoted by rival airlines, and a malicious agent could compromise an airline's booking system.

Proof-carrying code [112] is a cryptographic solution to malicious code prompted by research into mobile agents. This approach uses a theorem-prover to determine with certainty whether code can be safely executed. Program code is accompanied with a formal proof that certain safety conditions hold, such as memory protection and type-safety. Prior to executing the code the consumer checks that the proof is valid and that the safety policy is acceptable. This approach requires the programmer to add annotations to the source code to aid production of the proof. The language semantics must be analysed so the code can be mapped onto the proof, so work is needed to add support for target programming languages.

3.3.2 Sandboxing

Operating systems already provide security functions to separate data belonging to different users on the same system. Grid applications can be run within a sandbox where they cannot affect data belonging to other accounts or privileged data belonging to the host system [37]. A key problem with this approach is that mainstream operating-systems contain too many vulnerabilities to prevent access by malicious code [137]. Successful exploitation of a security vulnerability will often result in the operating-system protection mechanisms being bypassed [131].

Web services already have a mature security model for dealing with isolated application containers. Users can take advantage of this protection by deploying grid services using languages such as Java and .NET [155]. For example the Java virtual machine monitor enforces access controls on variables and code. Separation is provided by an application-level virtual machine monitor that runs on top of an existing operating-system. One downside of this approach is that the grid job must be entirely written in the target language, as the protection is not available to legacy code.

The Entropia and Gridbox grid platforms [20, 37] create a sandbox for safe execution of grid applications written in any programming language. Operating-system calls can be intercepted and blocked to control the functions the grid job can perform. It is possible to control and limit access to the filesystem, registry and network. Access to the graphical interface, mouse, keyboard, and other I/O devices can be prevented to ensure the grid job cannot interfere with the user of the machine in public-resource computing scenarios.

Condor [94] is a distributed computing application designed to scavenge spare CPU cycles on a group of computers. Condor has been used as a scheduler to support grid applications in a system called Condor-G [51]. Condor uses a form of application sandboxing called system-call redirection [94] that allows it to support remote filesystem access. Requests made by remote grid jobs to access storage devices are sent back to the local computing environment for execution. This means that large quantities of user data do not need to be made available on the remote system where it might be stolen, or not safely deleted, after use.

3.3.3 Trusted Operating-systems

The assurance of existing operating-systems can be enhanced using mandatory access controls. For example the NSA's SELinux mandatory access control solution [98] has become part of the Linux kernel. In this approach, there is no single superuser on the system that has access to all the local resources. Instead a security policy is enforced by the kernel and each process runs in a user context with the least privileges it needs to execute. Attacks that exploit vulnerabilities in applications with administrative privileges, a common attack scenario, cannot bypass a mandatory access control security policy. This makes it harder for malicious code to perform a privilege escalation attack in order to gain complete control over the host machine.

Mandatory access controls have been used to secure access to medical data in grids [68]. Privileged software is responsible for anonymising the patient records. Untrusted grid applications can be executed subject to a mandatory access control policy that permits access only to the anonymised patient data. Trusted computing attestation can be used to ensure the policy is correctly configured before patient records are transferred onto the system.

One disadvantage of using mandatory access controls is that the policies tend to be complex and hard to write. Henricksen et al. observe that the average policy configuration is around 50,000 lines [68].

A more heavyweight solution to the malicious code problem is to create an entirely new and more secure operating-system. The idea is to carefully engineer the system from the beginning to provide a highly robust level of isolation between software, and to minimise the likelihood of vulnerabilities in the trusted computing base. Many attempts have been made to do this in the past (e.g. VAX [81]), as well as more modern systems such as Microsoft's NGSCB [122] and EROS [150]. One of the difficulties with this approach is the lack of support for the large quantity of existing software written for mainstream operating-systems.

A number of researchers have worked on trusted operating-systems that maintain backwards compatibility with existing software using virtualisation [64, 124]. The general con-

cept is to have the virtual machine monitor enforce a security policy to isolate malicious code. This allows legacy operating-systems to run in a virtual machine so that existing software can be supported without changes. Nizza [64] and Perseus [124] use an isolation kernel to run applications in their own compartment.

Virtualisation is not always compatible with unmodified legacy operating-systems. Nizza and Perseus both require the mainstream operating-system running in virtual machines to have its kernel adapted to work with the virtualisation layer [65]. Other virtualisation systems such as VMWare Workstation [163] support unmodified operating-systems.

Mandatory access controls can be used within the virtualisation layer to further enhance the separation between virtual machines. This has been demonstrated by an implementation of sHype mandatory access control architecture [141] on the Xen virtual machine monitor [139]. The access control policy can constrain virtual machine access to resources and shared communications channels on the host platform. IBM has used sHype to implement a Chinese wall policy that prevents virtual machines from different trust domains executing on the same host [139].

3.4 The Malicious Host Problem

3.4.1 Non-Trusted Computing Solutions

Although this thesis focuses on trusted computing, there are a number of alternative approaches to solving the malicious host problem.

Job replication is a valid strategy when data integrity is the primary concern [107]. The idea is to submit the same job to multiple hosts and check that all the results match. If the results differ it is concluded that at least one of the hosts is malicious and the job is re-submitted. Replication is used in the BOINC public-resource computing middleware to detect cheating users in projects such as Seti@home [7]. A major disadvantage of this approach is the performance overhead introduced by duplicating jobs.

The Globe grid middleware architecture [126] uses data replication to secure large distributed data sets. Globe uses digital certificates to keep track of authoritative and cached copies of data replicas. Changes to the cached replicas are discarded, whereas write access

to the authoritative copies is restricted to a small number of trusted hosts.

Reputation is another way to identify and avoid malicious hosts [95]. Ratings for the trustworthiness of hosts can be generated in order to derive an aggregate trust value. Trust ratings can be obtained from automated security tools such as intrusion detection systems. Problems arise in this approach, such as how to deal with hosts that become malicious after operating correctly for a long period of time.

Obfuscation can be used to make it difficult to tamper with software [30]. These techniques are similar to those used to copy-protect software. The software and associated data can be encrypted, and the routines and key used to perform the decryption are obfuscated. Of course there are limits to this approach because with sufficient reverse-engineering it will be possible to recover the plaintext data.

Research on mobile agents have also prompted research into the malicious host problem. State appraisal has been suggested as a way to protect mobile agents from malicious modification on remote systems [43]. In this approach the agent is sent along with an appraisal function that tests the value of certain invariants before determining that it is safe to execute. In this way the agent can detect malicious modifications to its code or data. However this approach will not resist attacks that involve tampering with the appraisal mechanism itself.

Encrypted computation offers a way to use cryptography to hide data and software executed on remote platforms [36]. A one-time pad is used to encrypt the data input, output and the function applied to them. The function is specially chosen so that the encryption can be reversed on the client in order to recover the correct output. However, this approach is only suitable for functionality that can be expressed as a simple Boolean logic circuit at present.

Software traces allow malicious attacks to be detected using a detailed log of execution behaviour [175]. Remote host environments run an interpreter that is certified to operate correctly (so part of the host environment must already be trusted). The interpreter is adapted to produce a detailed trace of the execution steps. The trace is digitally signed and sent along with the results so that the sender can verify correct execution.

3.4.2 Trusted Computing Solutions

Trusted computing has been used as the basis for several systems implementing digital rights management. These systems try to prevent the user bypassing the protection policy even if they have physical and privileged access to the host platform.

Gallery et al. [52] explain how attestation can be used to verify that the host contains a trusted viewer for copyright-protected broadcasts before authorising their release. A similar approach can be used to control collaboration on business documents in a distributed system [144].

Sandhu et al. have designed an elaborate architecture that ensures a consistent policy is applied when protected objects are migrated between platforms [145]. Other research has shown how the monotonic counters in the TPM can be used to prevent data being viewed more than a pre-determined number of times, even if the entire state of the platform is rolled back in time [148].

Secure storage is useful to prevent the owner of the platform from modifying or stealing protected data belonging to other users. A number of researchers have shown how to modify the Xen virtual machine monitor to add support for encrypted and integrity protected storage [22, 76, 79]. All of these approaches work by using trusted computing authenticated boot to launch the virtual machine monitor (VMM). The private keys are then sealed so they can only be accessed by a valid VMM, and access to the keys by administrators is prevented. These keys are then used to encrypt and integrity-protect the virtual filesystems allocated to virtual machines.

One potential loophole in trusted storage comes from the management of virtual memory. Memory would normally be paged to the disk unencrypted, but it might contain sensitive data from a protected virtual machine. To overcome this problem the CHAOS architecture [22, 23] adds support for encrypted page files.

Löhr et al. describe how to adapt the Perseus virtualisation architecture [124] to create a secure grid platform using trusted computing attestation [96]. The virtualisation layer is expanded to include grid-specific services to support secure grid job transfer and execution. An attestation service is provided to allow the trusted software layer to be attested to

clients prior to transferring the grid job data. Grid jobs are received over an encrypted, integrity-protected communications channel established with the grid middleware software. Grid job data is written to disk using a secure storage service that applies encryption and integrity protection to prevent access by untrusted software, or physical attacks on the system.

Terra [54] is another virtualisation architecture that the authors suggest could be used as a secure grid platform. Terra is based on the VMWare virtualisation platform [163] with modifications to support encrypted and integrity protected disks. The virtual machine monitor intercepts disk I/O requests and performs encryption and integrity checking of disk sectors.

IBM have designed a distributed mandatory access control (MAC) architecture using trusted computing attestation and virtualisation called Shamon [108]. The MAC policy is enforced at the level of the virtual machine monitor, rather than the guest operating-system, which considerably simplifies the complexity of the policy, since it applies on a less granular level [141]. IBM suggest the BOINC grid platform as an exemplar use-case for Shamon, where the MAC policy ensures that only trusted clients can communicate with the BOINC server.

The Shamon architecture consists of a MAC enforcement virtual machine running on each trusted platform. It ensures that all members of a coalition of virtual machines share a consistent MAC policy. When two potentially remote, distributed virtual machines are joined in a coalition they each attest themselves to the other to determine that the trusted MAC enforcement is in place and that the policy is consistent and agreeable. The MAC policy specifies the attestation identity of trusted virtual machines.

Trusted computing can be used to protect stored grid credentials. Credential repositories provide a convenient centralised storage location for many users' grid credentials, but the security impact of the repository being compromised is very high. Hardened MyProxy [97] and Shemp [104] both use protected storage to ensure that grid credentials are secured using the trusted computing TPM. The authorisation secret for the grid credentials is set to the users' access password for the repository.

The Daonity architecture extends the use of protected storage to secure a user's credentials on every delegated grid host [102]. Trusted computing migratable keys are used to allow the delegation of their credential between platforms. Since the private key is secured by the TPM, it is no longer necessary to enforce a limited lifetime using proxy certificates. An online certificate revocation authority can be used to revoke access to the credential from specific hosts. This mechanism allows malicious hosts to be immediately excluded from the virtual organisation in the event that a successful attack is discovered.

3.5 Other Applications of Trusted Computing

Besides grid security there are numerous other applications of trusted computing, including the following (although this is by no means an exhaustive list):

- Digitally signing email using a trusted viewer [64] and delegating certificate signing on behalf of a certificate authority [34];
- Secure auctions [123] and electronic voting [146];
- Single sign-on using digital certificates [119] and biometric authentication [25];
- Controlling privacy in online user profiling of the Web [121];
- Remote access to networks by profiling the trustworthiness of connecting systems using attestation [140] and establishing an encrypted connection using IPSec [71];
- Secure services such as distributed firewalls [55] and secure logging [2];
- Preventing denial of service attacks using rate-limiting [55];
- Protecting and authenticating mobile phones and devices [33].

3.6 Conclusion

There are many competing approaches to solving security problems in distributed systems. However trusted computing is a compelling solution because of its unique ability to extend trust into remote systems using attestation. In addition the trusted computing TPM chip

offers an opportunity to harness inexpensive and widely available hardware security solutions. These security mechanisms cannot easily be compromised, even by attackers with full, privileged access to software running on the platform.

Attestation, however, is not a straightforward solution, as is demonstrated by the large quantity of research on its correct application. There are two principal problems. The first is in knowing what to measure. The second is how to know whether a system is secure based on this measurement. For example, attestation might be used to prove that a client is running a mainstream operating-system installation along with many software applications, but all this achieves is to prove that the system being attested is insecure.

The following chapter explores how to build a highly interoperable software security solution, suitable for use on a global grid, that is also small and simple enough to be meaningfully attested.

4 A Trusted Execution Environment for the Grid

4.1 Introduction

The malicious host problem is a key difficulty in the grid. When a grid job runs on a remote system the user has essentially lost all control over it. An attacker may replace the job with a Trojan horse that steals or tampers with the results. Without additional security measures the user may be unaware these attacks have occurred. The malicious host problem is challenging in the grid context because it is difficult to know whether a remote system can be trusted.

Protection against the malicious host problem using host-based authentication [50], as discussed in Section 2.2, is significantly weakened in a large, distributed grid. There is a possibility that a certified member organisation has in fact gone rogue, and is actively malicious, but that can be discounted as only a very slight possibility.

More significant is the possibility that the organisation contains one or more malicious individuals who seek to access grid jobs without authorisation. The level of motivation for this kind of attack may be high, especially if the data contains valuable or proprietary information that can be resold to a competitor, for example.

Most significant of all, however, is the likelihood that at least some members of a large grid will have failed to adequately protect their systems from external attackers. A single system that remains unpatched from a security vulnerability creates a window of opportunity in which an attacker can install Trojan horse software that can attack grid jobs.

A *trusted execution environment* is a security mechanism that allows the user to determine that grid jobs will run safely on remote systems. Grid jobs running in this environment are protected against attacks on data integrity and confidentiality throughout their execution life-cycle. This includes attacks originating from other users of the grid, as well as privileged users such as system owners and administrators, or external attackers who have compromised the grid infrastructure.

The remainder of this chapter is structured as follows. The next section explains the

required functionality of a trusted execution environment. Section 4.3 presents an argument that existing solutions are unsuitable for use on the grid. Section 4.4 describes a novel architecture that enables the use of trusted execution environments on a global scale grid. Sections 4.5 and 4.6 discuss the two key sub-components in that architecture. The *secure storage service* (Section 4.5) is responsible for encrypting and integrity-protecting the grid job data storage. The *job attestation service* (Section 4.6) allows a user to determine that a grid job has not been tampered with when retrieving results over the network.

4.2 Trusted Execution Environments

A trusted grid needs to provide a secure, isolated environment where grid jobs can execute free from unauthorised inspection and interference. A trusted execution environment protects grid jobs while they are executing on remote, and potentially malicious, grid infrastructure. Confidentiality protection ensures the data cannot be accessed from outside the trusted execution environment. Integrity protection provides tamper-evidence for the data so that the user can detect unauthorised modifications when the results are retrieved.

A trusted execution environment is the fundamental building block in solving the grid malicious host problem. The user begins by dispatching the grid job securely onto an authenticated platform that is known to provide a trusted execution environment. The grid job data is sealed using trusted computing so that it can only continue to execute while the platform remains in a safe state. At a later time the user securely retrieves the results, after verifying the integrity and authenticity of the grid job and the trusted execution environment.

There are many different ways of implementing these security requirements. A key difference in the possible approaches is the level of trust the user can place in the enforcement mechanisms. The question of trust decomposes into two separate factors relating to identity and behaviour:

- The degree of certainty that the remote platform offers a trusted execution environment;
- The level of protection that the trusted execution environment provides.

The first factor is a question of identity, and is non-trivial for remote systems over which the user has no administrative control. The second factor relates to the strength of the enforcement mechanisms that protect the grid job from attack.

One naturally looks towards approaches such as trusted computing to provide a high-assurance solution to this problem. The user can use attestation to verify that a remote grid platform provides a trusted execution environment. Assuming the trusted computing protection mechanisms have not been broken, the user can detect the presence of Trojan horses directly through attestation, since the software will no longer match its expected measurement. Rogue organisations will be discovered, as well as machines that have been compromised by attackers who install malicious software.

As trusted computing allows users to measure the identity of remote software directly, they no longer have to trust grid providers to secure their systems. The user can determine security compliance directly using attestation.

Successfully attesting a trusted execution environment is not enough. The level of protection it offers must be sufficient to protect users' grid jobs from attack. One strategy is for users to run their grid jobs under their own unique user account. The operating-system is then trusted to maintain separation. This approach does not achieve high assurance because a modern operating-system contains too much trusted code to enforce a strong degree of separation [99, 137].

Virtualisation can be an effective solution to enhance the security of mainstream operating-systems. By running each grid job in a virtual machine the level of isolation is significantly improved. The separation is enforced by a virtual machine monitor that is much simpler to design and build than an operating-system[1] [137, 12].

Trusted computing attestation and virtualisation form excellent partners in a trusted grid, but they do not completely solve the problem of building a trusted execution environment. Two key problems remain. The first problem is that grid administrators can subvert the protection because they are authorised to access all data on the platform. The second is that remote attackers can compromise the grid job through the grid provider's

[1]However due to implementation challenges secure virtualisation systems have yet to become widely available in commercial solutions, see Section 7.3 for further details.

infrastructure. For example, a vulnerability might allow an attacker access to networked storage, where they can directly steal or modify grid data.

To complete the trusted execution environment the grid job data must be protected when it is stored on the grid provider's network, so that it remains safe even if an attacker gains access via a compromised system. One approach would be to use segmentation and other countermeasures in the design of the network. There is, however, no easy way to use attestation to determine whether an entire network is secure. Attestation is normally only used to measure software running on an individual system.

The most effective remaining solution is to store the grid job data securely so that an intruder or administrator is unable to compromise it. Encryption can be used to protect data confidentiality. Attacks on integrity can be detected by comparing the hashes of read data to the previously written value. The encryption keys and hashes can be sealed using trusted computing protected storage to protect them from theft and tampering. This approach can prevent even privileged users, such as system administrators, from attacking users' data.

To summarise, a trusted execution environment requires three protection mechanisms:

1. Protection of the grid job in memory during execution. Virtualisation has been suggested as an appropriate solution to achieve strong memory isolation [137].

2. Protection of the grid job while it resides on secondary storage, which can be achieved by encrypting and hashing the data for confidentiality and integrity respectively.

3. The ability to determine remotely that these mechanisms are in place, which can be achieved using trusted computing attestation.

It is important to understand the attacks these security mechanisms are *not* designed to prevent. Firstly it is assumed that the virtual machine monitor, and the other parts of the trusted computing base, are secure from attack. Secondly, they do not protect the user against attacks that originate from within the trusted execution environment. If an attacker is able to compromise the grid job they will have full access to its data without any integrity or confidentiality protection. Compromise may be possible either because the

grid job performs an insecure function, such as sending sensitive data over the network using an insecure communications channel, or because the grid job software itself contains a vulnerability, such as a software bug, that can be exploited by an attacker.

Of course, this is no different to the problems inherent in running software on any computer. The aim is to make the security risks to software running on remote systems the same as if it was run on a local, trusted machine. Trusted execution environments are useful for grid providers because they ensure that any attacks are likely to be the fault of the user, since the grid job is strongly isolated from other parts of the system. Such an outcome is important for commercial grid providers because of the threat of litigation for theft or mis-use of valuable corporate data processed on their systems.

4.3 Research in Trusted Execution Environments

4.3.1 Existing Approaches

Trusted operating-systems have been suggested for use as a secure platform on which to run grid jobs [55, 108, 151]. In Microsoft's NGSCB operating-system [122], the Nexus security kernel runs on top of a streamlined set of software libraries that are invoked by applications using an XML-based language [114]. NGSCB requires all applications to be specifically targeted for development on the Nexus platform using a supported compiler.

BIND [151] allows attested software to run on any mainstream operating-system. When attestation is required a trusted security kernel takes control of execution. It is passed the memory address of the beginning and end of the measured process so it can read and hash the data.

The difficulty with BIND comes with the use of dynamically-loaded shared libraries and other memory regions which are non-contiguous. As these cannot be measured directly by the trusted OS, it is necessary, as described by Suh et al. [164], to adapt a stub in the user application itself to measure these memory segments. The stub is measured by the trusted OS to achieve a complete trust chain. As a result, applications must be specifically designed and compiled for BIND much like NGSCB.

IBM's Shamon architecture [108] implements virtualisation to provide support for grid

jobs that run on existing mainstream operating-systems. Shamon implements two of the three key requirements for a trusted execution environment, namely virtualisation and attestation. It omits to provide protected storage for grid jobs so it is not suitable for users that require confidentiality protection, since the data is stored unencrypted.

The Terra [54] and Trusted Grid Architecture (TGA) [96] platforms implement most of the important aspects of a trusted execution environment. TGA is specifically designed for use on the grid, whereas Terra is designed to support any application running on a potentially untrusted system, such as a home user's computer. These two approaches are compared and examined in detail in this section in order to draw conclusions about how appropriate they are for use in the grid (see Figures 7 and 8).

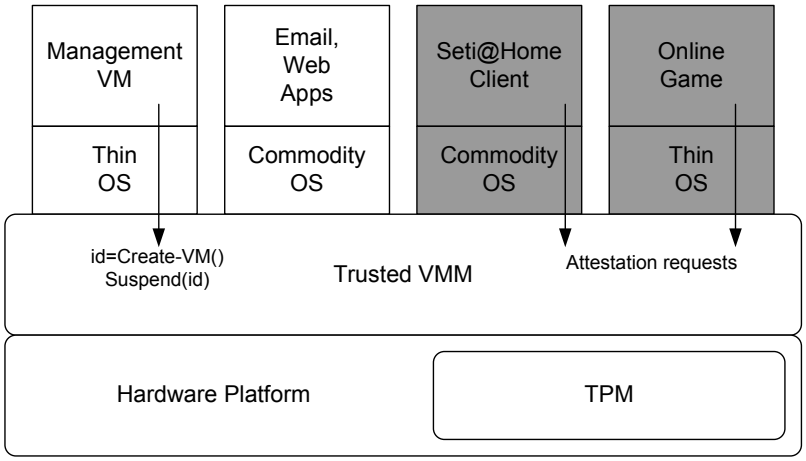

Figure 7: The Terra Architecture [54] ©2003 Association for Computing Machinery, Inc. Reprinted by permission.

Terra implements the commercial VMWare virtualisation platform [163] that supports a wide variety of mainstream operating-systems within guest virtual machines. The TGA is based on the open-source Fiasco micro-kernel developed at the Dresden University of Technology [70]. In a micro-kernel architecture, the operating-system kernel is re-implemented to support a highly efficient paradigm based on a message passing interface. As a consequence of the code changes required, operating-system support for virtual machines is currently far more limited, although Linux is currently supported under the L4Linux project [65].

Both Terra and TGA support integrity-protected and encrypted disks. In Terra, disk

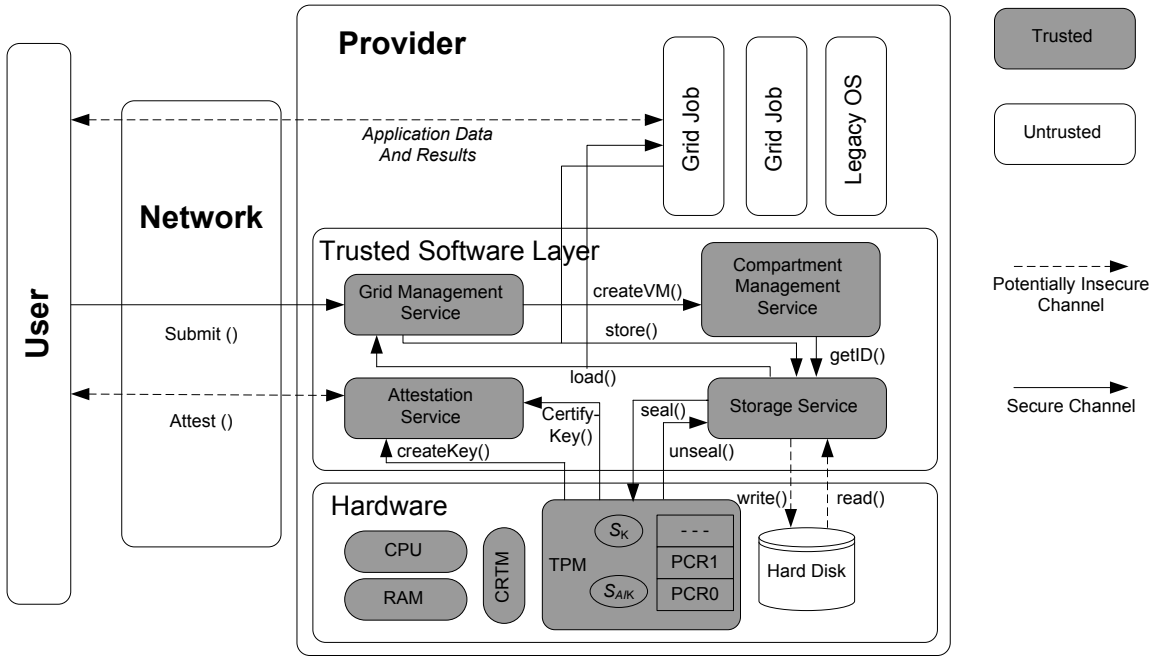

Figure 8: The Trusted Grid Architecture (TGA) [96] with kind permission from Springer Science+Business Media. ©Springer-Verlag Berlin Heidelberg 2007.

I/O requests are intercepted by the VMM and protected storage is provided using a custom library that interposes on the communication with the VMWare drivers. In TGA, protected storage is implemented by adding support directly into the native drivers in the microkernel.

Attestation in TGA is performed at the point of grid job submission. The measurement includes the grid middleware software, as well as the underlying microkernel and other parts of the trusted computing base. A data staging service, also included in the measurement, is responsible for setting up an integrity and confidentiality-protected channel over which the user transfers the grid job data prior to execution.

TGA does not directly support grid job attestation, and this makes it difficult to securely retrieve the results. Assume that at a later time a user re-connects to the host to which the grid job was dispatched. The attestation measurement will identify the host platform but not the grid job itself (since the grid job data is not included in the measurement). As a result the user will have no way to know whether the results were retrieved from the same grid job they dispatched, or a Trojan horse that is impersonating their job.

The developers of TGA point out that it is possible to adapt the grid job to overcome the lack of support for grid job attestation [96]. For example the user can embed a private key in their grid job to use for authentication when collecting results. The authentication key can be used to determine that they are communicating with the same grid job virtual machine that was dispatched.

Terra supports grid-job attestation based on a measurement of the virtual machine image file. The virtual machine monitor exposes a special interface to each virtual machine that allows the hash measurement to be obtained by executing a system call that is intercepted by Terra.

The authors of Terra give the example of Quake, a network-based computer game, which they have modified to support attestation [54]. Each player connects to a computer that is running the server software. The game client is modified to verify the attestation measurement of the server prior to connection, and clients are attested to the server before allowing them to join the game. Attestation is implemented by modifying the Quake source code and network protocol to add a request for the measurement as part of the connection handshake.

The trusted computing base of both TGA and Terra is fairly large. In Terra, the user must trust the device drivers used by the VMM, in addition to a privileged management VM that is used by administrators to control virtual machines. In TGA, this same functionality is contained within services layered on top of the microkernel to support VM management and physical device access. Each time these components are upgraded the attestation measurement will change, and this must be communicated back to the user. In addition, a vulnerability in these components will break the protection provided by the trusted execution environment.

4.3.2 Limitations of Existing Approaches

A significant amount of research has been done in the area of building trusted execution environments for grid systems (see the previous subsection for further details). There has been relatively little analysis, however, of the feasibility of these solutions in a large scale

production grid environment. The remainder of this section critically examines existing approaches, focusing in particular on issues concerning the practicality of their deployment on a grid.

Problem 1: Support for existing grid jobs

Approaches such as BIND, NGSCB and Terra require users to modify their grid jobs to support attestation. As will be argued more fully in the next chapter, it is impractical to imagine that all grid jobs could be adapted to support attestation in this way. The changes require access to, and detailed understanding of, the original source code, together with changes to the network protocol used by applications. Significant expertise is required to implement attestation in an appropriate way without introducing security vulnerabilities. This is not a task that could easily be performed by the majority of grid users themselves.

Adding support for attestation in the TGA introduces significant complications. Firstly, the grid job virtual machine must be modified to add support for authentication when retrieving the results. For example, VPN software could be used to establish an authenticated channel with the user based on a private key distributed within the grid job itself. Secondly, only the original submitter can verify the attestation measurement of the grid job. An unassociated third-party would have no way to verify the authentication key. A public-key infrastructure could be used, but this introduces a considerable management overhead.

The lack of support for existing grid jobs is a major hurdle in the wide-scale adoption of a trusted grid. The time and expense required to convert existing applications places a heavy burden on end-users. There is a danger that the trusted grid will become a niche service demanded only by users with unusually high security requirements. This is at odds with the vision of a widely available secure service that can be taken advantage of by the entire user community.

Problem 2: Interoperability

In trying to build a trusted execution environment for grid jobs it is necessary to focus on the key requirement of interoperability. The power of the grid comes principally from the ability to combine resources across potentially thousands of administrative domains. Standardisation of the mechanisms underlying the grid enables jobs to be executed seamlessly

from any location to deliver results with the optimum efficiency.

Grid jobs built for use in the Terra environment are not portable to other architectures. The grid jobs become tightly coupled to Terra's virtual machine monitor because specific system-calls are added to the binary code to support attestation. These system-calls will only execute correctly on systems running Terra.

TGA suffers from similar interoperability problems. In TGA the grid jobs are decoupled from the underlying microkernel and would potentially be portable to other architectures. The microkernel itself, however, requires specific changes to the operating-system running the grid job, essentially locking the user into grid platforms compatible with the Fiasco microkernel.

Both TGA and Terra require the support of a number of security components that must be installed within every grid execution host. For example both approaches require a storage service component to provide integrity-protected and encrypted storage for grid jobs. TGA requires a compartment and attestation service that integrates tightly with the rest of the grid infrastructure.

At present these are still active research ideas and there are no published standards for grid trusted execution environments. Each implementation offers its own unique set of security functionality including network I/O protection, secure storage, and different attestation schemes. Even within these functions there are differences such as the cryptographic algorithms used and the type of security protocols that are supported.

The current solutions for trusted execution environments cannot support a wide-scale, interoperable grid. Once grid jobs are developed for a particular trusted execution environment, they become locked-in. They cannot be used on another part of the grid if the service provider uses a different implementation. From the service providers' point of view their choice of solution severely limits the range of customers they can serve. From the users' point of view they remain restricted to using the subset of the grid that provides the specific security solution they have adopted.

In their current form, it is unlikely that service providers will adopt the trusted grid on a large-scale in the near future. Service provides must make vast and complex changes

to their existing infrastructure if they want to offer a trusted grid service. Worse, these changes must be coordinated over geographically distributed virtual organisations spanning many organisations, all of which must agree upon and implement a compatible solution. The expense and difficulty of this task means that the trusted grid might see a very low adoption rate in the initial stages, and there is a realistic possibility that it will never become available on a global scale.

Problem 3: Attestation

Attestation creates an interesting dilemma for interoperability. Ideally the details of the underlying security system and its implementation can be abstracted away using standard interfaces and protocols. Interoperability is a key driver for the application of web services in the provision of grid services. However when using trusted computing attestation the details of the implementation are a principal concern because the code itself is directly measured.

Attestation sits uneasily with grid standardisation. In principle attestation undermines the benefits for interoperability that standardisation can bring. Measuring a remote platform requires the user to be aware of, and furthermore to trust, a potentially large variety of trusted grid implementations. This is especially difficult because many commercial service providers are likely to use proprietary, closed-source solutions.

There are many management difficulties involved with attestation in a trusted grid. The existing trusted execution environments include software components that are owned and managed by the grid service provider, like device drivers and system management tools. The attestation measurement is therefore likely to change frequently whenever the service provider re-configures or updates their back-end infrastructure. The user community must be kept continually up-to-date with the details of these changes even though they might not be directly relevant to security.

Summary

In summary, the key problems with existing trusted execution environments include the following:

- Users must re-implement grid jobs to support attestation at great difficulty and ex-

pense;

- Users become locked-in to their chosen trusted execution environment and cannot make use of other parts of the grid or join non-compliant virtual organisations;
- The complex changes to existing grid infrastructure needed to support the trusted grid mean that it is likely to be adopted very slowly, if at all;
- The user is forced to verify details of the service provider's back-end grid infrastructure that are subject to frequent change, because they are included in the attestation measurement.

4.4 The Job Security Manager

The rest of this chapter develops a novel approach to deploying a trusted grid. The trusted execution environment is pushed down to the grid node alongside the grid job. The user packages together security mechanisms with the grid job so they can be enforced wherever the grid job executes.

A key benefit of the new approach is that grid providers no longer need to pre-install complex software. A single grid execution node can support multiple implementations of a trusted execution environment. Users are no longer tied into using a single solution adopted by their grid provider. The user can decide on the most appropriate solution to package with the grid job depending on their security requirements.

No modifications are required to the grid job to support the security architecture. The grid job is protected transparently so that it need not be aware it is running within a secure environment. Grid jobs containing legacy code are fully supported. Time and expertise is no longer necessary to adapt the grid jobs to run on the security architecture. These improvements help make the benefits of a trusted execution environment widely available to all users on the grid.

The new architecture works by distributing the grid job along with a component called the *job security manager*, which helps enforce the trusted execution environment (see Figure 9). Normally grid jobs would be submitted as stand-alone virtual machines. Under this

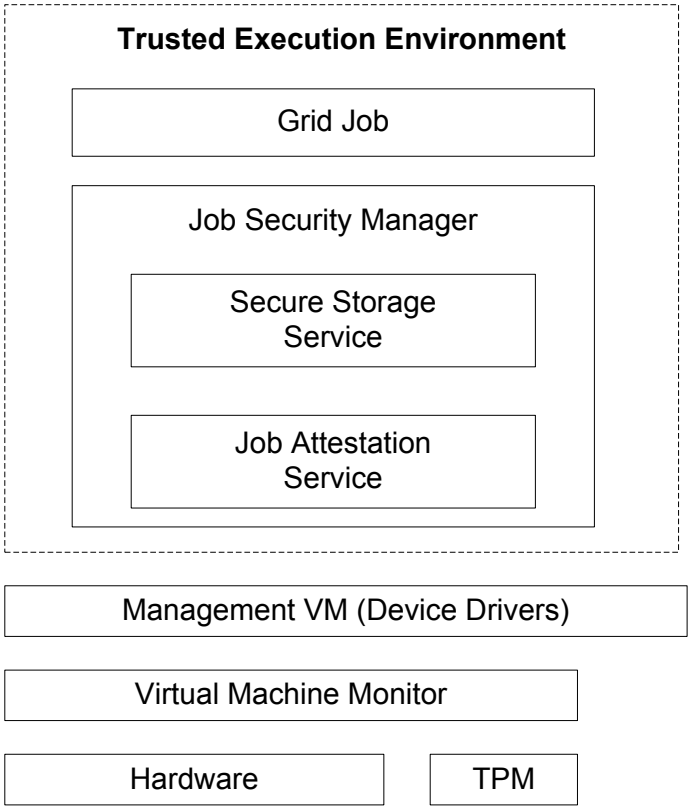

Figure 9: The Job Security Manager Architecture

new approach, each grid job is composed of two virtual machines, the first running the grid job and the second virtual machine running the job security manager.

The majority of the software involved in enforcing the trusted execution environment is packaged into sub-components of the job security manager. The *secure storage service* is responsible for encrypting and integrity-protecting grid data storage. The *job attestation service* uses trusted computing attestation to verify the integrity of the grid job on behalf of remote clients.

To ensure the architecture can be as widely available as possible very few requirements are placed on the host computer. Grid jobs can be deployed on any grid node that meets some basic hardware and software requirements. Memory protection, the remaining key requirement for a trusted execution environment, is provided by a virtual machine monitor that must be running on the host computer. Virtual machine monitors are available for

a wide range of computer hardware so this requirement can be met by many grid nodes. Secondly, a TPM is necessary for attestation and cryptographic key storage.

Virtualisation allows the job security manager to protect legacy, unmodified grid jobs. The security mechanisms are enforced transparently. No direct communications channel is opened to the job security manager because this would require the grid job to be directly involved. Instead the virtual machine monitor transfers control to the job security manager when the grid job accesses virtualised resources such as storage. The job security manager runs in a separate virtual machine, so it can apply protection outside the grid job itself.

The secure storage service works by virtualising access to the hard disk. Whereas the grid job believes it has access to a normal data partition, the underlying storage is encrypted and integrity protected. This is similar to the way the protection works in Terra [54] and similar architectures, except the software is implemented in the portable job security manager rather than the host computer.

Supporting attestation independently of the grid job is a novel concept. The existing approaches either do not attest to the grid job data at all, or require the grid job to be modified to support attestation.

The job attestation service works by virtualising access to the network. When the grid job makes a network connection the job attestation service takes over, measures the data and exchanges the attestation measurements. The grid job then continues execution as normal. The grid job is never directly involved, so it does not need to be modified to add support for attestation.

The job security manager architecture helps prevent attacks by administrators and other users with physical access to the grid node. Administrators are not permitted access to the cryptographic keys used to protect users' grid data. The job security manager is designed to prevent any tampering that would break this protection. The user verifies the attestation measurement of the job security manager when the grid job is first deployed to ensure that this protection is in place. The cryptographic keys are then sealed by the trusted computing TPM to ensure they are protected throughout the life-time of the grid job.

Grid platforms can be attested to users more easily using the job security manager

than they could with previous solutions. In TGA and Terra the user is forced to verify an attestation measurement for various system components in addition to their own, including device drivers and system management tools. This makes attestation more complex because the user must be aware of administrative changes made to the system (such as a driver upgrade) because these will change the attestation measurements.

In the job security manager architecture the user only needs to verify the attestation measurement of a small amount of trusted code, most of which resides in the job security manager that the user has themselves provided alongside the grid job. Such an approach is much easier than managing attestation for software maintained by the grid providers, that might change without notice. System management tools and device drivers are not part of the trusted computing base, so they can be modified without affecting the attestation measurement.

One of the key principles of the design is a clear separation of duty. Those parts of the system that are under the control of administrators can be used to implement security on behalf of the grid provider. The job security manager is wholly responsible for implementing security on behalf of the job owner. The division of job- and resource-owner security systems means the user only needs to verify the minimal amount of code needed for the protection of their data.

Another advantage of a clear separation of duties is that administrators can manage the system despite being partially locked out of it. In previous solutions administrators must be given access to the host environment in order to manage the system. Since the host environment also implements protected storage, administrators are able to gain access to the cryptographic keys used to protect users' grid jobs.

The job security manager shifts the provision of protected storage into a separate compartment where the cryptographic keys cannot be accessed. As a result administrators can be completely locked out of the job security manager compartment whilst still being able to perform their job.

The decoupling of the user-focused grid security functions from the host computer permits the user much greater flexibility. By choosing the job security manager that is pushed

down to the grid node the user is given the opportunity to refine and specialise the security functions for their own particular requirements. For example the user may want to add network proxy or firewall functionality to the job security manager, or change a security policy. Users more concerned about security might place more emphasis on these areas, whereas other users may choose to loosen security enforcement to gain performance, for example by selecting different cryptographic algorithms and key lengths.

To summarise, the key benefits of the job security manager architecture over existing proposals include:

- **Portability**: Trusted grid jobs can be supported by any grid execution node containing a TPM and running a suitable virtualisation platform, ensuring that the trusted grid can be made widely available.
- **Compatibility**: The transparency of the operation of the job security manager means it supports all grid jobs, including legacy code, without modification.
- **Consistency**: The security architecture is bundled with the grid job, ensuring that security functions are applied consistently across the grid despite differences in the underlying infrastructure.
- **Flexibility**: The user can configure the job security manager to support a range of security functions meeting their particular requirements for functionality, performance and so on.
- **Simplicity**: By providing the job security manager as part of the grid job the user no longer needs to trust and verify the attestation measurement of large amounts of software managed by the service provider.

4.5 Secure Storage Service

4.5.1 Transparent Storage Protection

The secure storage service is a component within the job security manager. It provides data storage confidentiality and integrity for jobs executing on remote grid platforms. The virtual

machine monitor works together with the secure storage service to protect against attacks from a potentially malicious host. Strong memory and storage protection are combined to prevent tampering or theft of grid data by attackers with access to the underlying grid infrastructure.

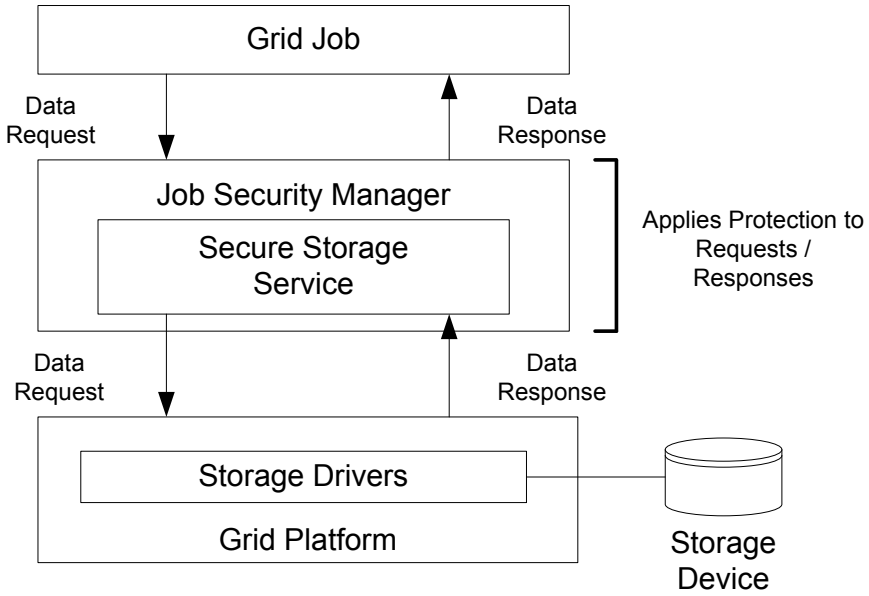

Figure 10: Secure Storage Service

The secure storage service (see Figure 10) sits in between the grid job and the storage device, protecting all data read and write requests. Read requests made by the grid job are forwarded on to the grid platform, where the data is read from physical storage. The secure storage service decrypts and integrity-checks the read data before returning the response to the grid job. Write requests are integrity protected and encrypted before being passed on to the grid platform where the changes are made to the physical storage device.

Virtualisation is used so the grid job does not need to be aware that the storage protection is taking place. The grid job accesses unprotected storage as usual. The virtual machine monitor configures the secure storage service to virtualise the grid job's storage. The responses to the data read and write requests made by the grid job are identical to those it would receive if it had direct access to unprotected physical storage. No modifications need to be made to the grid job to support the data protection applied by the secure

storage service.

Virtualisation is also used so the storage drivers in the host platform do not need to be modified. The secure storage service has its disk storage virtualised by the grid platform. When the secure storage service encrypts a write request, the encrypted data is passed on to the device drivers on the grid platform to be written to physical storage. The device drivers write arbitrary binary data to disk, so they do not need to be aware this data is encrypted.

Placing the secure storage service within the job security manager minimises the amount of code that must be trusted. Since the storage drivers can only access encrypted and integrity protected data, they are not part of the trusted computing base and do not need to be included in the attestation measurement of the grid platform. The grid provider is free to manage the disk device drivers without affecting users. The grid provider has great flexibility and can choose to implement a variety of local and networked storage solutions without affecting the attestation measurement.

The job security manager requires a different virtualisation configuration than that used in classical systems. Normally the virtualisation layer hosts device drivers for physical devices such as the hard disk. When a guest virtual machine accesses a privileged resource the virtual machine monitor transfers control to the device drivers. The job security manager architecture uses an additional layer of indirection. When the grid job accesses a virtualised resource, control is first transferred to the job security manager before being handed over to the the device drivers that handle access to the physical device.

A rogue virtual machine situated between the grid job and the secure storage service can break the storage protection (see Figure 11). The read and write requests made by the grid job are not protected until they reach the secure storage service. A rogue virtual machine positioned between the grid job and the secure storage service could steal or tamper with the requests without detection.

If the data is modified or stolen before it reaches the secure storage service then it will be too late to protect the data even if the rest of the architecture functions correctly. The job security manager architecture requires an exclusive and secure communications channel

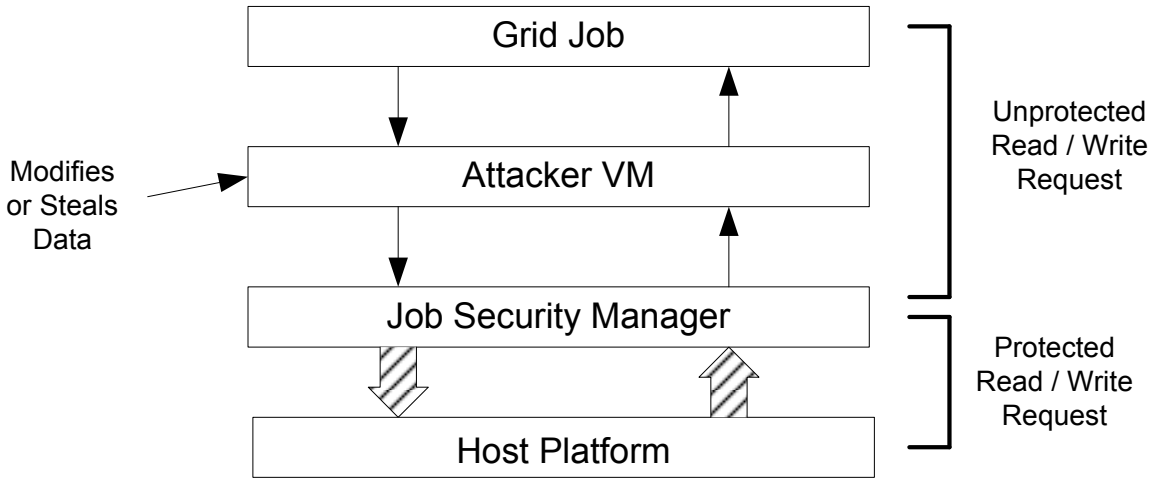

Figure 11: Attacking the Job Security Manager

to be established with the grid job to prevent these attacks.

It is the virtual machine monitor's (VMMs) responsibility to ensure that a secure channel is established with the job security manager. A VMM will typically set up a communications channel between two virtual machines by creating an exclusive shared memory region. The two virtual machines communicate with each other by reading and writing to the shared memory. The virtual machine monitor ensures that no other virtual machine on the platform has access to the shared memory region. Many VMMs already use these communication channels extensively to implement efficient hardware device virtualisation [12].

Provided the VMM virtualises the grid job with the correct secure storage service VM, and not a rogue VM, the attacks on unprotected data described above will not be possible. The VMM must attest itself to users to allow them to verify the shared memory channel is correctly configured. The changes needed to VMMs to support this process are described in detail in Chapter 7.

4.5.2 Tamper-evident and Encrypted Storage

The secure storage service allows the grid job to run with its data encrypted and integrity-protected on the physical storage device. Confidentiality is achieved by encrypting data

using a symmetric encryption algorithm as it is written to the disk.

Integrity is implemented by storing a secure hash of the data on every write operation, and checking that the hash matches when the data is read back. This approach does not prevent an adversary from modifying the grid job data, but it does ensure that all changes can be detected (this might be termed tamper-evidence).

An adversary must not be able to gain access to sensitive grid job data by tampering with the job security manager. To protect against these attacks the job security manager needs access to protected storage provided by the trusted computing TPM. Sensitive data is stored in protected storage, including the encryption keys and the hashes used for data integrity.

The job security manager must perform an authenticated boot process in order to protect the keys. Data hashes and keys are then sealed in protected storage where they cannot be accessed if the job security manager is tampered with. The current platform state must match the measurement at the time the data was sealed for access to be authorised by the TPM. Since an attacker can neither alter the integrity hashes, nor decrypt the encryption keys when they are sealed in protected storage, the grid job is protected from attacks involving tampering with the job security manager.

There may be many job security managers running on a single grid platform, but there can only be a single hardware TPM. IBM's virtual TPM (VTPM) architecture [15] allows a single physical TPM to be shared by many virtual machines. The physical TPM is used to protect the storage root key of the virtual TPM, so there is no significant loss of security. Using a VTPM is an effective solution to make the job security manager available to many grid jobs running on the same platform.

Some ways of implementing job data integrity are much more efficient than others. Digitally signing a hash of data written to disk (as suggested by the creators of TGA [96]) would achieve security guarantees, but with a large performance penalty due to the expensive public key operations. A more efficient solution involves using a message authentication code algorithm based on a private key protected by the TPM. Other researchers have suggested effective ways of implementing integrity-protected storage [85, 100, 173].

In terms of data integrity special care must be taken to prevent roll-back attacks, where an attacker replaces an entire encrypted image with one from a previous point in time without access to the decryption key [100].

Confidentiality can be implemented using a cryptographic key held in protected storage. The key is used to encrypt data written to disk and decrypt the data when it is read back. Other researchers have suggested suitable designs and approaches for encrypted filesystems [85].

Confidentiality protection is optional. If the user only requires integrity protection, it is sufficient to verify integrity by attesting the grid job when the results are retrieved. Any tampering with the grid job can be discovered at this stage because the attestation measurement will no longer match the expected value, and the results can be discarded.

Achieving confidentiality protection for grid jobs is significantly more challenging than integrity protection. If confidentiality is required for any part of the grid job data then integrity must be enforced at all times. Otherwise an attacker could tamper with the encrypted grid job and cause sensitive data to be leaked by altering its behaviour.

It may at first seem unlikely that an attacker could successfully tamper with the encrypted grid job to cause data leakage without access to the corresponding plaintext. A straightforward attack is possible when the grid job is only partially encrypted. Imagine that the application software is integrity protected, and only the input data is encrypted. This might be a common situation where the application software is public, but the data being processed contains proprietary or personal data. In this example an attacker can steal the input data by replacing the application software with a Trojan horse.

Even if the entire grid job is encrypted, integrity attacks are still possible. The attacker can attempt to tamper with encrypted blocks using data manipulation attacks that result in data leakage. These attacks are described by Niels Ferguson [44] and were overcome by building integrity enforcement into the design for Microsoft's Bitlocker encrypted filesystem. In any instance where grid job confidentiality is required, integrity must be enforced to avoid data manipulation attacks.

Confidentiality attacks that are achieved by tampering with the grid job can be pre-

vented with a small change to the job security manager. Tampering with the grid job will be detected by the job security manager when it checks the data against the stored hashes, since they will no longer match the hash of the stored data. At this point, the secure storage service can discard the invalid data and refuse to return it to the grid job. The result is that it is impossible to execute tampered code, and an attacker cannot introduce a Trojan horse into the grid job by tampering with data storage.

4.6 Job Attestation Service

4.6.1 Attesting the Platform

A problem still facing the user is how to securely transmit an encrypted job onto a grid node. So far this chapter has concentrated on how a job can execute in a trusted execution environment. It remains to be explained how a job can be securely transferred onto the platform.

It is not possible for a user to distribute a grid job ahead of time that can be accessed by any valid job security manager. The user could try submitting the job encrypted with a public key, with the equivalent private key sealed under the attestation measurement of the job security manager. However, the user would have to negotiate this public key with the destination platform prior to dispatching the grid job. Only non-migratable keys can be sealed, so there is no way to encrypt the data to a private key that is then shared by all possible destination platforms.

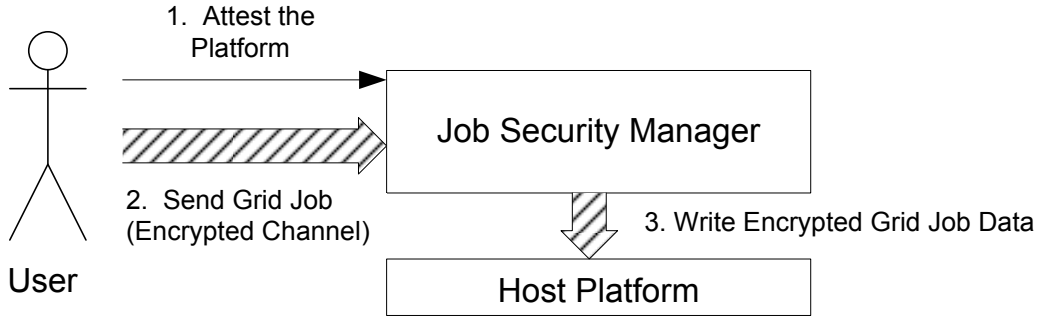

Figure 12: Securely Dispatching a Grid Job

One solution to securely transfer data into a trusted execution environment is to have

the user actively involved. A suitable approach is shown in Figure 12. The user begins by requesting the attestation measurement of the job security manager. The user checks the measurement is valid to ensure the job security manager has not been tampered with and can be trusted to safeguard the encrypted job. The user then transmits the grid job data over a confidentiality- and integrity-protected communications channel (using a secure protocol such as TLS [35]).

The job security manager receives the grid job data and encrypts the data as it is written to disk using the secure storage service. This approach ensures the grid job is securely negotiated onto the grid platform, where the grid job will be protected throughout its lifetime by the trusted execution environment.

This solution is not compatible with many grid systems. Normally the user would be offline at the time when the grid job is scheduled for execution. Data staging services would be used to upload the grid job data on the user's behalf. Chapter 6 returns to this issue and explains how to develop a more sophisticated version of this architecture that is compatible with grid middleware services.

4.6.2 Transparent Grid Job Attestation

One of the major problems with existing solutions is that grid jobs have to be modified to support attestation. The job attestation service can perform attestation without the direct involvement of the grid job. Any grid job can immediately gain the benefits of attestation when it is run in this architecture. Grid developers and users can begin using trusted computing without the need for expensive modifications to existing grid jobs.

The job attestation service virtualises the grid job's network access. When a user connects to the grid job to collect results the job attestation service takes over and performs a remote attestation exchange. During this process a measurement is sent to the user allowing them to verify that the grid job ran without tampering. If the user is satisfied that the grid job ran correctly, they can continue to communicate directly with the grid job to obtain the results. The grid job is completely unaware that the attestation exchange has taken place.

The job attestation service works by acting as a network proxy for the grid job. All network connections going into or out of the virtual machine are intercepted. When the job attestation service receives an incoming network connection from the user, it performs an attestation handshake with the client. The virtual machine monitor uses memory protection to ensure that the network traffic exchanged between the proxy and the grid job cannot be attacked by untrusted code running on the remote grid platform.

Attacks on the job attestation service are possible if it is not designed carefully. An attacker could wait for attestation to occur and then substitute the job security manager with a Trojan horse. Since memory is protected by the virtual machine monitor the attacker would have to do this by modifying the job security manager on disk and restarting it. IBM solve this problem in their Shamon architecture [108] by having the client periodically demand re-attestation. This approach is unsatisfactory because there is a time window between checks in which the Trojan horse could operate undetected.

Impersonation attacks can be prevented by having communication take place over a protected channel. The job attestation service authenticates itself to clients and sends the grid job measurements over an integrity-protected network connection. All subsequent communications with the grid job, including remote attestation, take place over the protected channel. A temporary cryptographic session key is used for integrity protection and is held in memory.

Attackers can no longer modify the job security manager and restart it without detection because the temporary session key will be lost and existing client connections cannot be resumed without re-negotiating attestation measurements. This improvement eliminates the time window in which impersonation attacks can occur.

Rather than developing new cryptographic protocols it is important to make use of existing standards wherever possible to avoid introducing new security vulnerabilities. The protected channel established by the job attestation service could be implemented using a variety of well known protocols such as TLS [35] and IP Security (IPSec) [168]. Both of these protocols would allow the user to authenticate the grid platform using host certificates. Authentication prevents man in the middle attacks where attestation measurements

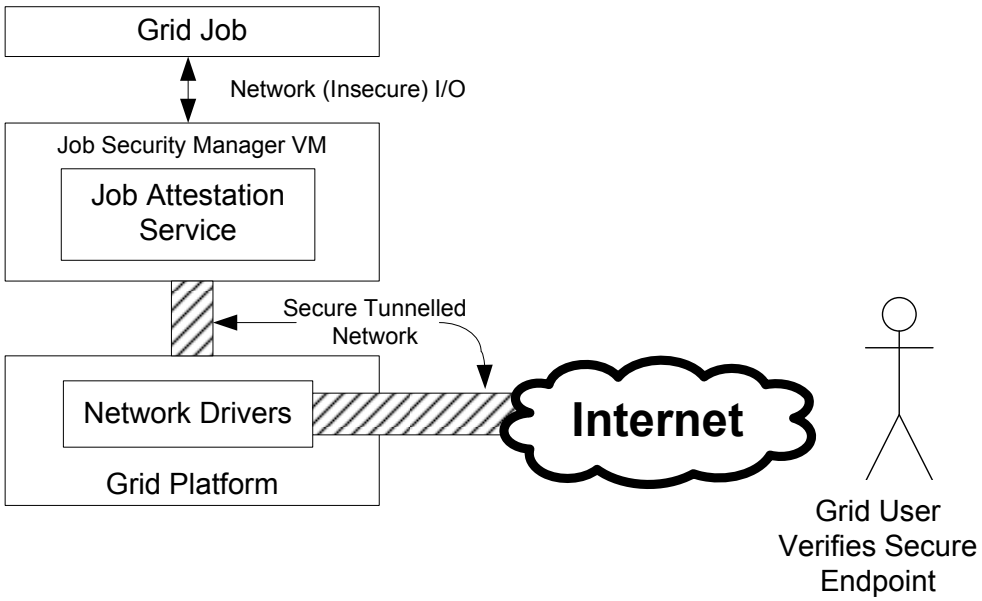

Figure 13: Job Attestation Service

are relayed from another system by an attacker to fool a user into interacting with an impersonated grid job.

A high-level design for the job attestation service is shown in Figure 13. The architecture uses multiple levels of virtualisation in the same way as the secure storage service. The grid platform software hosts the drivers that access the physical network card. All network traffic is protected by the job attestation service before it is passed to the network drivers. As a result the grid provider's networking software is not part of the trusted computing base. In existing approaches such as TGA and Terra, changes to the network management infrastructure would alter the attestation measurements. The job security manager architecture allows the grid provider to manage their network infrastructure freely without affecting users.

There are a number of different ways a user might connect to a grid job during its execution life-cycle. The user may want to perform computational steering, or retrieve partially completed results. Alternatively the grid job may connect back to the user (or a service of their choice) during execution to push down the grid job results or deliver a status update. The job attestation service proxy is able to initiate a protected channel for

both incoming and outgoing network connections, so it is flexible enough to deal with all these data exchange scenarios.

Changes in application security requirements will become commonplace as existing organisations shift from using a local trusted network to a remote grid infrastructure. A secondary benefit of this architecture is that it allows security to be easily retrofitted onto existing code. Users can make use of protected network communications without making any modifications to their grid jobs. Confidentiality protection can be added to the protected channel established by the job attestation service, in addition to integrity protection, if the user requires it. This allows organisations to add security protection to existing applications to deal with the new threats that arise in the grid.

Remote attestation cannot be completely transparent. Although no modification is required to the grid job to support attestation, the responsibility is placed on the user to verify the measurements. Grid job integrity can be verified by a wide variety of client systems because the verifier does not need a TPM in their computer. Nevertheless there are potential interoperability problems because client applications and bespoke systems need to be modified to perform remote attestation.

The job attestation service can establish the protected channel prior to exchanging the attestation measurements. This approach has the advantage that existing protocols such as TLS [35] and IPSec [168] do not need to be modified, but with the drawback that existing client software must be adapted to retrieve and verify the remote attestation measurements. An alternative is to adapt a protocol such as IPSec to include exchange of attestation measurements as part of the security association [168], permitting interoperability with legacy applications.

4.6.3 Remotely Attesting a Grid Job

Grid job attestation is not as simple as measuring the job itself. The job attestation service must allow the user to establish trust in all the components that are involved in the trust chain. Several components are part of the trusted computing base, including the job security manager and the virtual machine monitor.

Conceptually the user must verify two parts of the system. Firstly, a measurement proving the integrity of the trusted execution environment, and secondly a measurement of the grid job running within it. Both of these measurements are exchanged with the user by the job attestation service.

Attestation was initially designed to measure a single computing platform. The TCG have released specifications for the PC describing how the TPM can be used to measure each stage in the boot cycle up to the operating-system [166]. With the introduction of virtualisation a single platform might be running many virtual machines each containing their own operating-system and software. The attestation of virtual machines is complicated because they are not aware they are virtualised, and thus cannot directly measure the underlying virtualisation platform.

A new approach is needed to support attestation for virtual machines. Berger et al. describe a solution in their work on virtual TPMs [15] (see Section 2.3.7 for further details). The initial stages of the boot cycle are the same as with a normal PC platform. The virtual machine monitor is one of the first pieces of software loaded, and its measurement is stored in the physical TPM[2]. When a virtual machine is created these existing measurements are copied into its virtual TPM. The virtual TPM continues to store measurements for the virtual machine itself. The user can now verify the integrity of both the virtual machine and its underlying virtualisation platform using attestation.

There is a need to use attestation to verify networks of computers and not just individual platforms. IBM's existing work on VTPMs assumes that each virtual machine is completely isolated, so it can be measured individually. In the architecture proposed here this approach would not be useful. The grid job virtual machine cannot be measured separately because it relies on the job security manager to enforce a trusted execution environment. An extension to IBM's work is necessary to support the concept of *coalitions* of virtual machines that are tightly coupled together, where measuring one without the other would be insufficient (see Section 7.4 for a detailed discussion of coalitions).

[2] An added complication here is that the trusted virtualisation software might include other components, such as a Management virtual machine, that would also need to be measured (this is discussed further in Section 7.3)

The problem of attesting the grid job is simplified because the virtual machines form a hierarchy. The job security manager virtual machine loads before the grid job, because it provides it with secure access to disk storage. The solution is to have the job security manager use the trusted computing authenticated boot process and then measure the grid job. The virtual TPM now contains a hierarchy of measurements from the virtual machine monitor through to the job security manager and the grid job itself (see Figure 14). This is a change from the IBM architecture [15] because the virtual TPM contains measurements for more than one virtual machine. The job attestation service can now verify the measurement of the grid job together with all the other components relies upon to support the trusted execution environment.

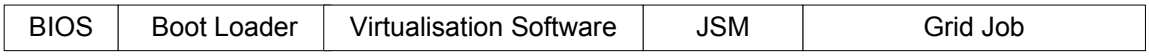

Figure 14: Example Platform Configuration Register Layout for the Job Security Manager

The job security manager needs to measure the grid job virtual machine in order to attest it. Normally virtual machines are completely isolated and cannot access each other's data. The virtual machine monitor permits the job security manager to virtualise the storage of the grid job it is attached to. The secure storage service has access to a set of data files that each contain a data storage partition used by the grid job. Before the grid job starts the secure storage service can decrypt and read these data files to generate a measurement for the grid job. The job attestation service uses this measurement to verify the integrity of the grid job.

A number of different attestation schemes can be implemented by the job security manager architecture. The job security manager can perform a new approach to attestation that is particularly suited to grid jobs (this is the subject of Chapter 5), but also a number of alternative approaches that have been suggested by other researchers. Some forms of attestation are easier to implement than others, however.

The attestation scheme used by Terra can be implemented by having the secure storage service read and hash a subset of the grid job data partitions every time attestation is requested. Some data partitions are used to hold dynamic data, and these are not measured.

These partitions are protected from modification using the integrity-protection feature of the secure storage service. If the static data is altered the measurement will change, so the measured subset of data partitions should be chosen carefully.

The secure storage service is not well suited to attestation schemes that operate on a low level of granularity. The IBM Integrity Measurement Architecture measures each executable file and library as it is loaded during software execution, which can be verified later using attestation. This style of attestation cannot easily be implemented by the job security manager because it sees the grid job data files as a collection of disk sectors. It does not have access to the filesystem drivers necessary to identify individual files and folders. Removing filesystem support from the secure storage service reduces its complexity, simplifying attestation and reducing the likelihood of vulnerabilities. It also ensures that the architecture works with legacy grid jobs implementing any arbitrary filesystem.

Attestation schemes that measure individual files can be implemented at the expense of increasing the complexity of the job security manager. Filesystem drivers can be installed within the job security manager. The device drivers used to access the physical storage device remain in the management virtual machine administered by the grid provider. The secure storage service can now read individual files from the grid job and use these to create the attestation measurement. The grid job is still unaware this measurement is taking place, so this approach maintains compatibility with unmodified grid jobs.

4.7 Conclusion

One of the key problems with existing trusted computing grid solutions is a lack of interoperability. Complex changes must be made to existing grid middleware, but this precludes re-use of existing software. There is a danger that a trusted grid might become an enclave used only by those communities that have particularly strong security requirements, rather than a widely accepted mainstream technology.

The job security manager offers a solution to this interoperability problem. The security functions are placed in a portable virtual machine. The functions operate using virtualisation, beneath the level of existing software. As a result no changes are required to existing

grid jobs or middleware to use the job security manager. This is a significant improvement over existing state of the art research.

The job security manager is a secure foundation that can form the foundation of a trusted global grid. Since it is small and simple, it is meaningful and feasible to verify using attestation. However users must check the integrity of more than just the job security manager to know that a grid job has run securely — they must also verify the grid job itself. The next chapter goes on to examine a new form of attestation that is particularly suited to grid jobs running in public-resource computing style environments.

5 Attestation on The Grid

5.1 Introduction

Attestation is an exciting security primitive that makes it possible to extend trust over the varied distributed systems that comprise the grid. When software is run on remote platforms beyond local administrative control, it is difficult to ensure that security policies will be appropriately applied and enforced. There may be numerous potential attackers, such as rival organisations sharing the same infrastructure, and insiders with physical access to the remote platform.

In conventional approaches it is necessary to take it on trust that countermeasures have been appropriately applied. In a grid context, a single mis-configuration in the data centre, or an unpatched component, could render grid jobs vulnerable to a variety of threats. Attestation offers huge potential to improve this situation because it allows users to measure the identity of remote systems, and thus ensure that appropriate technical measures are in place to protect data assets deployed on the grid.

On the face of it attestation seems to be little more than a code-signing digital signature. A digital signature is typically used by a vendor to sign a published software package in order to prove its origin and integrity to customers who receive the software through potentially untrustworthy distribution channels. Likewise, attestation uses a digital signature to prove the origin and integrity of software running on a computer system. So why have so many researchers developed alternative approaches to using attestation (as described in Chapter 3)?

Trusted computing remote attestation differs from traditional digital signatures in the potentially high-degree of uncertainty in the identity of the object being signed. In the case of a traditional digital signature the receiver has possession of the object, and therefore its identity is self-evident. When a user requests remote attestation, they are not in control of the identity of the system being measured. They must rely on the trusted computing infrastructure to correctly capture and relay the identity of the remote object. Many problems with trusted computing arise from the potential for an unknown discrepancy between the actual and the measured identity of an object.

One important use for attestation in the grid is to confirm that jobs have run free from tampering and interference. The grid job can attest itself to the user when collecting the results to show that it has not been replaced with a Trojan horse that might tamper with the data. The results data can be protected with a key sealed to the identity of the grid job to enforce its integrity throughout its life-time.

If the user requires confidentiality protection for their data it is not sufficient to have the grid job attest itself when retrieving the results. In addition the grid host must be attested when the grid job is dispatched. Otherwise, a malicious platform might tamper with the execution of the grid job causing sensitive data to be leaked to the attacker. Although confidentiality protection is optional, for completeness the rest of this chapter will normally assume that it is required.

When a platform is first switched on, an authenticated boot process takes place where all software is measured prior to execution. Once these measurements are made they will usually remain unaltered even though the state of the platform may continue to change. The attestation measurement represents the state of the platform when it was first booted, not its state at the current point in time. The idea here is that any Trojan horse software that is present on the system will be detected by the initial authenticated boot measurement.

It is important to understand that a platform reboot is a significant event in the context of attestation. After the platform is rebooted a fresh authenticated boot process takes place. At this point changes to the state of the grid job during its previous execution will be measured. For example state changes could include temporary data and results written to the disk. These changes can cause a difficulty for a user attesting the grid job at a later time when collecting the results, because the measurement will have changed from its initial value when the grid job was first dispatched.

This chapter examines a number of ways that jobs running in a grid environment can attest themselves to users. Shortcomings with existing approaches to attestation led to the development of a new scheme called origin attestation that is more suited to grid environments, where uncontrolled reboot events may occur frequently due to system maintenance or failure.

5.2 Problems with Attestation

5.2.1 Entire-state Attestation

Attesting the entire data state works well for a limited category of grid jobs. Assume that a user submits, say, one thousand jobs that each run for one hour before returning results. If any of the grid hosts needs to be rebooted (for example due to a power failure), the attestation measurement will be recalculated. If the grid job stores no state information during execution, and the results are stored in memory and not written to disk, the attestation measurement will remain the same as when it was dispatched. The user can verify this measurement when retrieving the results by comparing it to the original value.

In the more likely case where the grid job stores some state information, the attestation measurement will be altered. When retrieving the results, the grid job will fail attestation because the hash does not match that the value expected by the user. Since the grid jobs are short-term, this will not be a major problem for the user and they can simply re-submit the failed job. This analysis shows that this approach works well in environments where very few failures or platform reboots occur, or for short-term grid jobs for which the time penalty of resubmission is low.

Unfortunately most grids are designed to support long-running grid jobs with a high degree of fault-tolerance. Consider the case of a grid job that runs for many days to complete a computation. If the platform restarts, the attestation measurement will be invalidated, and the user will no longer be able to successfully retrieve results from the attested grid job. The only solution is to restart the job, leading to days of lost CPU time and an unacceptable quality of service.

Another important category of grid for which attesting the entire job state fails is public resource computing. In these scenarios, grid jobs are run on computers that are shared with other interactive users, who may be physically using the computer system. For example, consider a cycle-stealing application where grid jobs are run on workstations located throughout an enterprise, or Seti@Home style computing [8] where grid jobs are run on users' home computers. Such systems cannot be relied upon to run for any length

of time without interruption. If the computation runs for several days, it is very likely that the owner of the platform will reboot or shutdown the system, invalidating the attestation measurements and necessitating a complete re-submission of the grid job.

A final category of grid job for which attesting the entire job data is inappropriate is cloud-computing type delivery models. In this model the grid would need to support access to critical business data over a long period of time. Consider a web service, for example, where the cloud is used to give guaranteed levels of performance, storage, and throughput for a large corporate web site [130]. In software-as-a-service distribution models, it would clearly be unacceptable if the availability of business-critical data was affected because the attestation measurement was invalidated after a platform reboot. Software-as-a-service models must support very long-term execution of grid jobs with reliable guarantees over access to the outsourced data.

5.2.2 Partial-state Attestation

To successfully have long-running jobs attest themselves, it is necessary to implement more complex attestation schemes. Many researchers have observed similar problems with attestation, and proposed a number of solutions mostly based on the idea of separating out the more static parts of the system for measurement (see Chapter 3 for further details).

Unfortunately these approaches are currently unsuitable for the grid. The remainder of this section examines the limitations of these attestation schemes in the grid context.

Simplifying attestation by restricting measurements to the binary executables themselves (for example, IBM's Integrity Measurement Architecture [143]) can lead to insurmountable problems due to the number of possible legitimate variations. For example, Jones reports that in the first six months of 2007 there were 36 patches released for Microsoft Windows XP, 281 patches for Red Hat Enterprise Linux version 4.0, and Apple fixed a total of 60 vulnerabilities in OS X [78]. The number of valid attestation values for software varies with the factorial of the number of patches, assuming that the vendor allows subsets of the available patches to be applied according to customer requirements.

This leads to an unmanageable situation where the number of valid software identities

for a single product becomes far too large to cope with through the release of digital signatures. In its present state, attestation is not well suited to mainstream operating-systems and other complex software due to the huge number of potentially valid identities.

Attestation is seen as being a threat to open-source software [137]. When the developer has access to the source code they can compile their own software. In this case the identity would change depending on a variety of complex factors, including the exact version of the compiler and software libraries in use. In open-source operating-systems it is common to patch the kernel to provide support for custom hardware, for example, leading to an almost limitless number of valid kernel images.

With closed sourced systems, such as Microsoft Windows, there is only a single version of the kernel and a large but still limited number of drivers available. Under trusted computing, open-source systems are said to suffer because it becomes impossible for any distribution authority to attest to a de facto version of the software while allowing full freedom for users and developers. This has led the free software advocate Richard Stallman to tag trusted computing as 'Treacherous Computing' [57], since it threatens the movement for open-source computing platforms.

Isolating the static, attestable components, from the highly variable temporary data in applications is itself a difficult problem. Difficulties arise because legacy software has not been specifically designed to accommodate attestation. It is typically insufficient to measure only executable code, as the following example will demonstrate.

Consider a firewall application. To determine the trustworthiness of the firewall it is necessary to have an attestation measurement of both the enforcement software and the policy configuration files. Without this distinction it will be impossible to tell the difference between a firewall that allows illegitimate inbound and outbound traffic, and one that correctly implements corporate policy.

Manual analysis is necessary to determine the exact data that should be attested. This process is expensive as it ideally needs the involvement of both domain- and security-experts. A careful balance needs to be struck between ensuring that the firewall cannot be maliciously subverted (measuring a lot of state information) and reducing the complexity

of the attestation measurement (measuring less state information). The cost of adding attestation support to legacy applications is one of the key barriers to the mainstream adoption of attestation as a universal mechanism of trust negotiation between networked components.

In some cases legacy software is likely to require a significant re-design to support attestation. Take for example BOINC, a popular middleware platform for public-resource computing used for science projects such as Seti@home [7]. McCure et al. suggest an architecture where the project science server verifies results from clients using the attestation measurement of the science application [108].

A detailed analysis of the operation of BOINC [32] reveals that this approach fails in practice. The BOINC client saves a cached copy of the science project's home page every time it runs. A hidden tag embedded in the web page is used to retrieve the location of the server that hands out jobs and accepts results. By manipulating this file, an attacker could redirect the results to a rogue server. Including this file in the attestation state for BOINC clients is infeasible, as the measurement would change every time the project home page is updated. This example shows that determining the appropriate boundaries for measurement can be difficult, and often requires a detailed understanding of the underlying code.

The need for software modifications arises primarily because existing applications have not been designed to support attestation. In grid scenarios, there is typically a requirement to support the execution of dynamic, user-generated jobs. Many grid users are unlikely to have the expertise necessary to correctly implement attestation for their jobs without the support of a security specialist. For many grid applications the result is that attestation functionality will simply not be available.

Another problem with attestation in its current form is a lack of support for legacy software. Applications cannot be re-designed to support attestation if the source code is no longer available, or if the application is closed-source and the vendor is unwilling, or unable, to implement the required changes and provide ongoing support.

Trusted computing significantly increases the vulnerability surface of applications, and

therefore increases the difficulty of writing secure code. Traditionally, developers concerned with application security have focused on preventing vulnerabilities that allow attackers to execute privileged code. As a result, it is industry best practice to run code with the least privileges necessary to perform its function [73].

Preventing privilege-escalation attacks will not help reduce the impact of vulnerabilities in attested code. Most software bugs, regardless of whether they allow privilege escalation, will allow an attacker to replace an attested application with a Trojan horse. For example a buffer-overflow vulnerability will allow an attacker to inject arbitrary code into the memory space of the affected application. The attacker can then alter the behaviour of the attested software. The behaviour-conformance advantages of attestation can easily be lost, because the vulnerability surface is much larger than it is in the majority of software in use today.

The increased vulnerability surface restricts the category of applications for which attestation can be used. A necessary consequence of this is that attestation can only be used reliably for small, simple applications, or those for which the probability of bugs resulting in arbitrary code execution can be minimised. Microsoft's NGSCB [122], for example, requires legacy applications to be ported to a trusted operating-system. The need for re-implementation would cause significant problems for grid-type environments as it would restrict the ability to run open, interoperable operating-systems and code.

Remote attestation holds great promise for the grid, where there is a need to automatically establish trust in hundreds, or even thousands of service providers, where non-technical means of establishing trust break down. Attestation is not a "magic bullet" that can be easily deployed to solve these problems. This section has addressed many issues with attestation that mean that, in its present form, it is not well suited to the grid.

A grid that can only run software designed to support attestation can only meet the needs of a niche community whose members have the necessary time and security expertise to produce such systems. A specialised, secure subset of the grid is in opposition to the research community's vision of an inclusive, global grid that might change the nature of commodity computing on a wide scale.

5.2.3 Formalising Existing Approaches

This section presents a more formal analysis of the existing attestation schemes. This analysis will help clarify their shortcomings, and will serve as a useful introduction to the following section, which uses the same formal representation to describe how origin attestation improves on these earlier approaches.

Consider a simple approach for grid job attestation where the entire grid job and the associated (stored) state is measured. Let s_0 represent the state of the grid job when it is dispatched. In a typical grid job, the state will change over time as temporary values and new results are written to the disk. Each successive state s_x represents a new, changed state of the system when a new attestation measurement is made, for example following a platform reboot.

Let $M()$ be a function that takes the state of the system as input and produces a unique attestation measurement for each possible input value. Typically, $M()$ would be a hash measurement of the grid job state data, and this assumption would not hold due to hash collisions, but these issues can be safely ignored for the purposes of this discussion.

Let us assume a simple scenario where the user dispatches a grid job that runs to completion, and the results are then collected. When the user retrieves the results from the grid job in state s_x, the job is attested by checking that the measurement $M(s_x)$ corresponds to a valid job state s_x. Figure 15 shows the attestation values that would be produced for the given state.

System state:	s_0	s_1	s_2	...	s_n
Measurement:	$M(s_0)$	$M(s_1)$	$M(s_2)$	...	$M(s_n)$

Figure 15: Attestation of the Entire Grid Job State

Measuring the entire grid job state works well for grid jobs whose attestation measurement is unaltered between deployment and collection of the results. When the user dispatches the job, the initial measurement will be $M(s_0)$. Consider the case where the dispatched job is entirely self-contained – it computes a result in memory without writing any data to the disk. In this case, even if the platform is rebooted, the measurement will

remain at s_0 because the on-disk state remains unaltered.

If the same user dispatches the job and collects the results, they can easily verify the measurement $M(s_0)$ matches the expected value by remembering it at the time of dispatch. Alternatively, a third-party (such as the vendor that wrote the software) could sign a digital signature to vouch for the validity of $M(s_0)$. Digital signatures allow the results to be verified by a different user to the one who submitted the job.

Measuring the entire grid state causes problems for grid jobs whose attestation measurement is likely to be updated between dispatch and results collection. For example, consider the same scenario where the grid job contacts a server for the input data, and the grid jobs are run on users' home computers (this represents a typical Seti@home style public-resource computing scenario).

The initial state will always be the same value of s_0. Once the user shuts down and restarts their computer, the system state will change to s_x, according to the input data and current state of the output data and temporary files written to the stored state. On retrieving the results, the user will have no way to determine whether $M(s_x)$ is valid. Enumerating every possible state would be impossible for software of any reasonable level of complexity, particularly taking into account possible non-deterministic values in the input or output data (for example, writing to the operating-system virtual memory page file). Measuring the entire grid job state is therefore not suitable for a substantial category of grid applications.

An alternative method proposed by several researchers involves separating the program state into a static portion of trusted code and data, t, and dynamic, non-security relevant state u_x [54, 143, 122]. Again, assume that u_0 is the measurement of the dynamic state when the job is submitted, and u_x represents a changed dynamic state during a subsequent attestation measurement. Figure 16 demonstrates the idea behind this attestation scheme.

System state:	t, u_0	t, u_1	t, u_2	...	t, u_n
Measurement:	$M(t)$	$M(t)$	$M(t)$	...	$M(t)$

Figure 16: Attestation of the Partial Grid Job State

This approach has the same advantages as grid jobs where the entire state is measured

and the attestation measurement does not alter from the time of dispatch. At every possible system state u_x, the attestation measurement is always the same, $M(t)$. The user who receives the results can either verify $M(t)$ based on the known value during job submission, or a third-party could digitally sign the measurement $M(t)$ to vouch that it corresponds to a trustworthy software distribution.

There are a number of serious shortcomings with approaches that separate out static state for attestation. The most serious of these is the need to segment the job state into a trusted state, and a dynamic untrusted state. These decisions need to be made using manual processes that are time-consuming and highly prone to error. In general, changes may be required to the source code of the software to support a clean division of the trusted and insecure state (further discussion of these problems can be found in the previous subsection). As a result, this attestation approach does not align well with the vision of an open, ubiquitous and easy to use grid platform.

5.3 A New Approach to Attestation

5.3.1 Origin Attestation

Origin attestation is a new form of attestation that overcomes many of the problems in existing approaches. Origin attestation always returns the measurement of the grid job state at the point of dispatch, or origin, rather than its state at the current point in time (see Figure 17). Each time the state of the system changes and is re-attested, the historical measurement of the grid job at the point of dispatch, $M(s_0)$, is returned.

System state:	s_0	s_1	s_2	...	s_n
Measurement:	$M(s_0)$	$M(s_0)$	$M(s_0)$	...	$M(s_0)$

Figure 17: Origin Attestation

This has the same advantages of whole job attestation, where the value $M(s_0)$ can be easily verified because it is well known. It overcomes the problems of whole job attestation, because even if the state of the system changes during execution the attestation measurement remains a constant value $M(s_0)$. Origin attestation does not require the job state

to be divided into static and dynamic portions in order to achieve a constant measurement value like other approaches, so it can be applied automatically to all legacy grid jobs without modification.

It may seem unusual that origin attestation returns a historical measurement that tells the attester only something about the state of the system in the past. After all, an attacker could have maliciously modified the job in the intervening time to replace it with a Trojan horse. Origin attestation is designed to work on a host that provides a trusted execution environment to prevent external modifications to the grid job state. An attacker cannot maliciously modify the grid job because it is protected using memory protection and secure storage.

When using origin attestation, the attester must examine $M(s_0)$ to determine that the job is running in a trusted execution environment (using the approach described in Section 4.6.3). If this is the case, then the attester can conclude that any changes to the grid job state must have occurred through legitimate execution of the grid job. The attester can additionally verify the grid job measurement comprised in $M(s_0)$ to ensure the job is not an impostor. So origin attestation allows the job to be verified while disregarding irrelevant changes to the job state that result from normal execution behaviour.

To illustrate how origin attestation works for a typical grid job take, again, the example of a Seti@home style grid job run on home computers, where every grid job is an identical executable program. The project server dispatches the grid job in state s_0. s_0 is a well known, constant value. During execution, the job contacts the project server to obtain a unique set of input data and begins execution. The user then shuts down their computer and upon restart the system is in, say, state s_1. The job completes and contacts the project server to return the results. The job attests itself using origin attestation with the measurement $M(s_0)$, which the project server verifies corresponds to a genuine job executable, and additionally verifies that the job is running in a trusted execution environment. Since the results were produced from a valid grid job executable, and the owner of the platform did not tamper with the results, the project can conclude that the results are valid.

It might seem somewhat surprising that the TPM could produce an attestation mea-

surement for a platform state that no longer exists. The key here is to understand that, from the point of view of the TPM, the attestation measurement is based on arbitrary values given to it by software. Most existing researchers have based this measurement on executables and data currently stored on persistent storage, but there is no requirement to do this. In origin attestation trusted software that runs earlier in the authenticated boot cycle stores a measurement of the grid job when it is first dispatched to the platform, and this value is replayed in subsequent attestation challenges.

Attestation involves a digital signature of the values stored in the platform configuration registers (PCRs) within the TPM. Each measurement takes place by concatenating it with the previous value held in the requested register, and then hashing the result (as described in Section 2.3.3). Normally when software loads, a series of measurements are made into the same PCR representing the hashes of each software component that is loaded from disk.

Origin attestation involves trusted software setting the PCRs to a historic value representing the state of the platform in the past. The sequence of measurements comprising $M(s_0)$ are held in protected storage where it cannot be modified by unauthorised software. The sequence of hash values are retrieved from protected storage and each measurement in turn is extended into the relevant PCR. Once this process is complete the PCRs will be set to the identical values representing $M(s_0)$, and attestation can be performed as usual.

Attesting the trusted execution environment when the results are retrieved is sufficient to prove that the integrity protection was enforced throughout the execution life-cycle. To see why this is the case, consider that at some point prior to attestation an attacker tampered with the job stored on the disk. When the trusted execution environment restarts the secure storage service will detect the modifications and it will refuse to continue grid job execution. The changes can be detected because the hash of the read data will not match the hash value written to protected storage when the grid job was last securely shutdown.

Also, consider what happens if the grid job is tampered with prior to its first execution. When the user dispatches the grid job they verify the grid host using attestation and then transfer the data over an integrity-protected, authenticated channel. As the data is written to disk, the data hashes are generated and written to protected storage. The attacker

cannot manipulate the hashes at this stage because the platform is attested to ensure it does not contain malicious code.

Origin attestation is easy to implement within the job security manager described in Chapter 4. The secure storage service is modified to store the initial measurement $M(s_0)$ in protected storage. The measurement is calculated by hashing the initial job state as it is written to disk after being dispatched by the user over a secure communications channel. The measurement is sealed so that it can only be accessed by the job security manager, and it remains unaltered throughout the lifetime of the grid job. The job attestation service retrieves the initial measurement from protected storage each time attestation is requested.

5.3.2 Discussion

Origin attestation is more suited to short-term grid jobs because the measurement at the point of origin becomes less useful after a long period of time has passed. Consider the example of a software-as-a-service model where a company outsources an email server onto third-party infrastructure. The company would like to use origin attestation to determine that the server has the most recent security patches applied before allowing it to process an email containing their intellectual property. The origin attestation measurement will only reveal the state of the server when it was first deployed, and the measurement will not confirm what state changes, such as security patches, have occurred since that time.

These problems are not so much related to the length of running time as they are with the degree of resolution the attester needs over exactly what state changes have occurred. For example, a long-running grid job that runs independently on a computational problem would not suffer from these drawbacks, because the attester knows that all the state changes are the result of normal execution. Origin attestation is less useful when the attester needs detailed resolution of the current state of a system, especially when that system undergoes frequent security-relevant changes to its state.

A detailed knowledge of the current state of a software system is sometimes unnecessary provided it is known that the normal execution behaviour is secure. In origin attestation the trusted execution environment protects against unwanted external interference to the

job state, so all remaining changes must have been requested by the job itself. Take, again, the example of an email server on the cloud, where the user would like to determine that up-to-date security patches are installed. Origin attestation cannot prove the current state of the system. Provided, however, the email server regularly self-updates as a result of normal execution, the user may take it on trust that the current security patches have been applied.

There is a risk that the security updates failed, for example due to a broken network connection. The email server can be designed to resist such failures by refusing to service requests until the update server has been contacted and all updates discovered are successfully installed. There is a greater risk involved in using origin attestation because it does not detect silent failures, but the risk can be managed even for long-running services.

Origin attestation is weaker than traditional forms of attestation because the user has no way to determine whether malicious changes have occurred since the job was deployed. The trusted execution environment protects against external malicious changes, such as replacing a binary file with a virus on persistent storage, but not changes that are requested by the job itself during normal execution.

For example, consider a job that contains a vulnerability allowing an attacker to execute privileged code. The attacker can insert a backdoor into the system allowing them privileged access in the future even if security patches fixing the vulnerability are later applied. Using origin attestation there is no way to detect the malicious changes that have occurred since deployment. In traditional attestation approaches, persistent backdoors can be detected because they result in a change to the binary files on disk, and these changes will be detected when re-attestation is triggered, such as following a platform reset.

Other forms of attestation only provide weak protection against vulnerabilities in the attested software, so the fact that origin attestation is also weak in this area is not a significant detraction. The general problem in this area is that attestation does not detect in-memory modifications. An attacker need only shift their strategy and ensure that all attacks modify binaries in memory and not on persistent storage to avoid detection. Origin attestation is thus no weaker than other forms of attestation in this regard.

However, if the attacker does insert a backdoor into the system this may allow them access to the system in the future even if the vulnerability is patched. Other forms of attestation can identify backdoors provided the modified files form part of the measured state of the system. Provided origin attestation is used for relatively short-running jobs the lack of detection of persistent backdoors does not represent a major shortcoming.

Origin attestation is compatible with grid job migration across platforms with some additional support from the host platform. Consider a job in state s_x that is migrated to another platform. Normally this platform would measure the state of the job at the point of dispatch, which in this case would be $M(s_x)$. Unless the grid job was migrated in state s_0, origin attestation will fail after migration because the attester will have no way to verify a value like $M(s_1)$, say. A relatively simple fix to this problem is to allow the host platform to securely migrate the measurement $M(s_0)$ to the target platform during migration. When the grid job is measured it will then return the correct value $M(s_0)$ following migration. As long as the measurement cannot be tampered with during migration, and the $M(s_0)$ is associated with the correct grid job and not an impostor, the security guarantees of origin attestation should hold.

Although this section has focused on applications of origin attestation for grid jobs, the approach is widely applicable to software in other scenarios. Consider the example of an Internet bank that distributes packaged virtual machines to customers that contain a safe web browsing environment for customers to access their bank account. Using normal attestation methods the virtual machine measurement would need to be specifically programmed to disregard state changes, such as the browser cache, that result in unpredictable changes to the overall state of the virtual machine. Under origin attestation, the bank need only verify the measurement $M(s_0)$ matches the virtual machine originally dispatched to customers.

Since this research was performed a similar concept has been published by Paul England from Microsoft, which he terms *birth certificates* [41]. The focus of this work is on the use of attestation within enterprise computing environments where IT personnel wish to verify the state of machines under their control, such as employees' laptop computers. The

independent discovery of this concept in other application domains underlines that this approach has wider use beyond grid computing.

Checkpointing allows origin attestation to detect malicious changes to a system resulting from an exploited vulnerability. Returning to the example of an Internet bank that distributes a safe browsing environment to customers, the bank is concerned that some customers are running VMs that at one time contained a web browser vulnerability that attackers have used to install a backdoor Trojan onto the system to steal password details. Using checkpointing the bank resubmits a fresh virtual machine to customers after a serious vulnerability has been patched. The bank uses origin attestation to ensure that all customers are using the latest checkpoint virtual machine image.

Checkpointing is a useful solution for software that does not store data that must be maintained between checkpoints. In the example of a safe browser for Internet banking, replacing the software is easy because no important persistent data is stored on the client. On the other hand, consider a peer-to-peer collaboration environment for working on office documents. Once checkpointing has been performed all historical data would belong in a separate compartment that would fail attestation because it does not conform to the latest checkpoint image. Collaboration on historical documents would then no longer be possible, which is likely to be unacceptable to users.

Solutions to this problem do exist but they rely on a migration system that exchanges data between trusted execution environments. Such a system would be complex to build because it must ensure that it does not migrate Trojan horses along with the data. Origin attestation has been designed mainly to facilitate use in grid environments where the job can be resubmitted without any risk of permanent data loss.

5.4 Case-study: Public-resource Computing

This section explores a case-study showing how the job security manager architecture presented in the previous chapter can be combined with origin attestation to deliver a public-resource computing (PRC) solution.

First consider BOINC [7], a prevalent middleware platform for PRC projects. BOINC

implements a number of server- and client-side middleware components. On the client side, a *controller* application allows the user to select PRC projects they wish to join. The controller allows the user to run multiple science applications at the same time. The user may choose from a number of projects that are built on BOINC including Climate*prediction*.net [159], Seti@Home [8], and many others.

When a user joins a science project the controller contacts the project's servers and fetches a *science application*. It also authenticates the user using their username and password so their results can be tracked on the project leaderboard. The server-side BOINC components, which form part of the project's servers, are responsible for generating *work units* for the science application to process, and retrieving results sent in by the controller.

One key problem with BOINC is that there is no way to enforce data integrity protection. The controller and science applications run directly on a user's home computer and therefore can easily be compromised by malware. The current solution to this problem is to re-submit the same work unit to many different users and only accept the results when a particular number of the results match.

This case-study considers the following scenario. It is desired to update BOINC to run on a trusted grid architecture to avoid the need to resubmit work units unnecessarily. Integrity protection will allow any changes to the results, the science application or other trusted software to be detected, and the results discarded. The BOINC developers would like the new system to meet the following constraints:

- Make minimal changes to the existing software. This includes the controller, the server-side components, and also existing science applications.
- Not all users have a platform containing a TPM and trusted virtualisation software. The software has to interoperate with existing legacy users.
- It must be possible to run BOINC like any other virtual machine. Users should not be expected to replace their underlying virtualisation software with a more secure version compatible with BOINC.
- The new system will protect the integrity of the results. There is no desire at present

to support confidentiality, but this might be required in the future and it should be possible to upgrade without difficulty.

- Individual science projects have no particular reason to trust other science projects that use the BOINC platform. Since BOINC is an open platform, anyone can join it, including potentially malicious individuals. As a result, it should be possible to prevent one project from affecting the results of another.

The basis for the solution will be the job security manager presented in Section 4.4. The secure storage service will be used to provide integrity of the software and data while it is executing on participants' computers. The job attestation service will be used to verify the integrity of the client system when retrieving the results to ensure it has not been maliciously or accidentally tampered with.

The job security manager architecture has a number of properties that make it well suited to the project's requirements. Firstly, because the security layer is pushed down to the participant's computer, it does not require any changes to their existing virtualisation platform. In addition, it can run on any virtualisation platform offering a TPM that is trusted by the science project. Secondly very few changes are required to BOINC to support the new platform. The controller and science applications do not need any modifications, since the security functions such as attestation are performed within the job security manager[3]. Thirdly, since the job security manager can already support confidentiality (see Section 4.5.2), this capability can easily be added later on.

The architecture chosen for the deployment is shown in Figure 18. The BOINC project team plan to develop a job security manager virtual machine to help support the new version of BOINC. Each project's science application will run in a dedicated virtual machine together with a controller to handle delivery of the results. If the user is participating in more than one science project using the BOINC platform, they can control the CPU-time allocated to each virtual machine to adjust the proportion of time they wish to allocate to each (this would be a feature supported by the virtual machine monitor rather than

[3]Changes are required to support launching of BOINC within a virtual machine, as discussed later, but most virtualisation environments should provide a simple way of doing this.

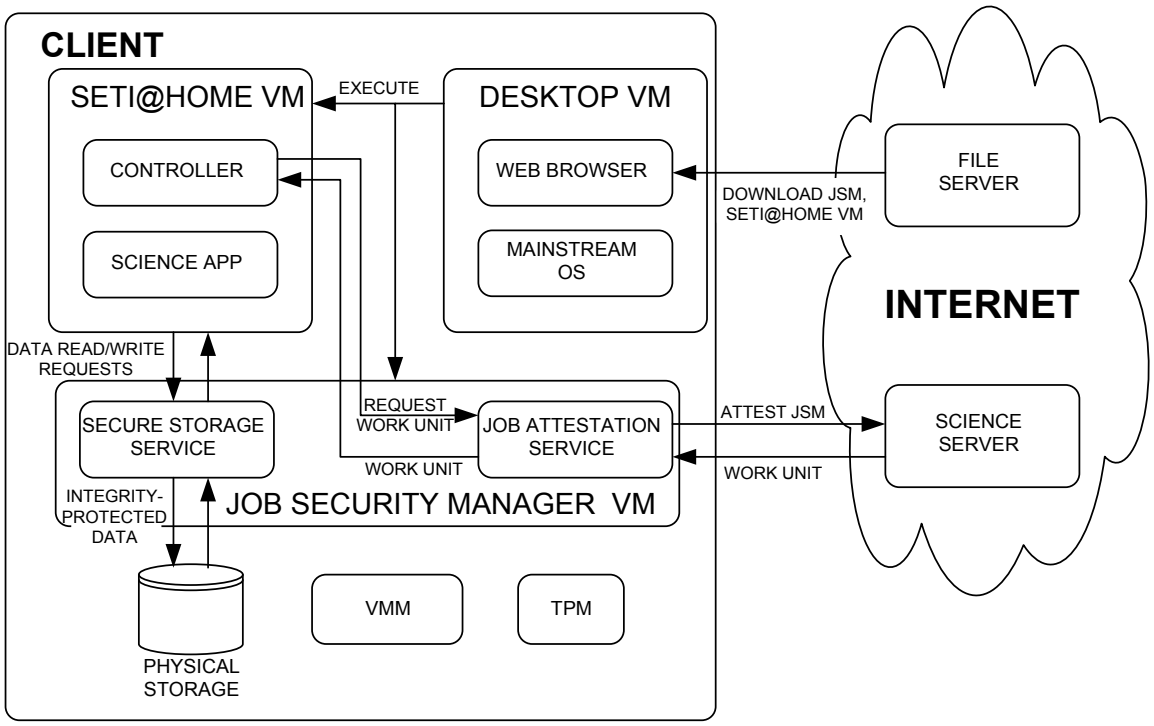

Figure 18: Public Resource Computing Case-study

BOINC itself).

Since individual science projects do not trust each other, the user will run each science application in its own virtual machine, packaged with its own instance of the job security manager. For example, a user might run one virtual machine configured to use Seti@home, and another configured to use Climate*prediction*.net. Since they run in separate virtual machines they are strongly isolated from each other and cannot affect the integrity of each others' data.

In the new architecture a user performs the following steps to join a new science project:

1. The user downloads a new software package containing images for the job security manager and Seti@Home virtual machines.

2. The user configures and launches the virtual machines (the full details of the requirements for the virtualisation platform is given in Section 7.4).

3. The secure storage service within the job security manager ensures the controller and

science application are integrity protected during execution.

4. The controller starts running and presents a GUI allowing the user to enter their username and password. In fact this is identical to the existing BOINC controller except it does not allow a user to join more than one project (this must be done by launching separate virtual machines).

5. The controller contacts the science server in order to fetch the first work unit.

6. The job attestation service, which is virtualising access to the network, intercepts the connection request and attests the client platform to the science project server (this is described in further detail later).

7. The controller continues to function as normal, requesting work units and returning results as necessary. Each time a network connection is made the job attestation service performs an attestation handshake before establishing the connection.

8. The GUI is used to present the user with feedback and visualisations to monitor the progress of the analysis performed by the science application.

Using the origin attestation protocol described in this chapter means that no changes are required to the existing BOINC software to support attestation. The origin attestation measurement will always match the data that the user initially downloaded in stage 1 above, even across platform reboots. The origin attestation works as follows:

1. The job security manager begins execution for the first time. Each data partition attached to the job security manager is enumerated. In the example this includes one partition holding the controller, and another holding the science application.

2. Each data partition is read in its entirety and the resulting hash measurement is stored in a platform configuration register in the VTPM. A log of this sequence of measurements is then sealed in protected storage to the identity of the job security manager, so it can be retrieved after a platform reboot to perform origin attestation (as described in Section 5.3.1).

3. The data needed to support data integrity for the secure storage service (for example, disk block hashes) are saved to disk using protected storage under the VTPM, sealed to the identity of the job security manager, where they cannot be tampered with or accessed by other applications.

4. The Seti@Home virtual machine is booted. The secure storage service provides integrity protection for the controller and science application throughout execution. Each time data is read from the disk it is checked against the saved integrity hashes. As data is written to disk, the integrity hashes are updated to reflect the new values.

5. When the controller initiates a connection to the science servers, it does so using a network proxy with the job attestation service (using the process described in Section 4.6). The job attestation service initiates an attestation handshake with the project science servers.

6. The next time the job security manager starts up following a platform reboot, the disk hashes and measurement log are retrieved from protected storage. Provided the job security manager has not been tampered with it will be able to retrieve this data.

7. The original measurements in the log file are then replayed into the appropriate platform configuration registers.

8. The secure storage service then reads the disk hashes and ensures the science application and controller have not been tampered with as they are executed from disk.

Consider what happens during software updates to the controller or the science application. Normally these components will periodically contact the science server in order to retrieve an update. Since the secure storage service provides integrity protection, this update mechanism cannot be tampered with. Provided it takes place over an integrity protected channel (such as SSL), an attacker (including the user of the computer) cannot directly interfere with the update process.

One of the most attractive features of this solution is that very few changes are required to existing software. Since they operate in a virtual machine, the controller and science

applications continue to function as they would have if run directly on the user's desktop. The secure storage service provides strong protection against attacks on integrity without requiring complex security software to be added to the user's virtualisation software. As a result it is compatible with many different virtualisation architectures.

One of the disadvantages of origin attestation is that it is not possible to be certain which version of the software is currently running. In Section 5.3.2 this problem was described, along with a solution called checkpointing. Checkpointing can easily be implemented in this architecture. If a serious vulnerability is found in the science application, for example, the user can be prompted to download the update. They do this on their desktop computer and replace the virtual machine image of the science application (or a tool can do this for them automatically). A configuration option is set on the job security manager indicating that the next time it starts it must generate a fresh origin attestation measurement to reflect the software updates.

Care must be taken when performing checkpointing to take a fresh measurement of the entire system. If any part of the system remained unmeasured across a checkpoint, it would provide a persistent home for malware. The attack would proceed as follows: the attacker replaces the controller with a Trojan horse which infects the science application with malware. The attacker then rolls back the controller to a legitimate version and initiates a fresh checkpoint. To avoid these attacks, fresh measurements of every component must be made to ensure they do not contain a Trojan horse. If the controller is checkpointed, the science application must also be measured for the purposes of attestation. Note that one downside of checkpointing is that partially completed results and user credentials are discarded, but a checkpoint should be a relatively rare event, so this is unlikely to be a serious issue.

Not all users will have the necessary hardware and software to support a trusted grid. This includes a TPM, and virtualisation support from their host operating-system. BOINC can be designed to inter-operate with legacy systems. When a new work unit is retrieved, the job security manager performs attestation. When this is performed successfully work units can be labelled to ensure they are not given out again. On the other hand, work units

that are retrieved from legacy software that is not trusted can be given out to multiple computers, and the majority result chosen, as is done in the existing approach. This hybrid solution can take advantage of legacy computers as well as deriving maximum benefit from computers capable of supporting a trusted grid.

5.5 Conclusion

Public-resource style computing scenarios offer particular challenges for attestation. Each time the host computer is restarted the attestation measurements are re-calculated, making it difficult to pin down any static notion of identity. Software can be specially designed to overcome these limitations, but that creates barriers for wide interoperability and availability of a trusted grid on a global scale.

This chapter has explored origin attestation, a new approach that permits full interoperability with unaltered, legacy software. This is achieved by having the virtualisation layer measure the software before it is first executed. Since this measurement is stored in protected storage and retrieved each time the grid job is restarted, it remains consistent over time. Provided the initial state was secure, and a trusted execution environment has protected the job from outside interference during execution, a trust decision can be made on the basis of an origin attestation measurement.

The following chapter goes on to examine in detail how trusted execution environments can be adapted for use on the grid. The grid raises particular challenges for attestation, because the user might be offline at the time the job starts and unable to perform an integrity challenge. To solve this problem a new component, the key management service, is introduced to perform attestation on the user's behalf.

6 A Solution to the Grid Middleware Problem

6.1 Introduction

The middleware problem is the high likelihood of security vulnerabilities in the software that drives grid systems. Grids currently rely on middleware software that has been designed to optimise performance and interoperability. These goals necessarily result in complex software that is likely to contain vulnerabilities. Indeed, the short history of grid middleware systems has proven this to be the case (the full justification for this argument is given later in Section 6.2).

The middleware problem leads to high impact attacks. Grid infrastructure is implicitly trusted in most current architectures. An attacker who compromises the grid middleware will gain access to all the grid user credentials delegated to that resource. They will also gain access to the sensitive grid job data accessible from that resource. The middleware problem therefore undermines the trust placed in grid systems.

There is a trust asymmetry problem in the grid. The user authenticates and delegates their credentials to the grid resource. Authentication helps the grid resource provider gain trust in the user, but the user has no way of knowing whether the grid resource is trustworthy. The user is forced to delegate their credentials to the grid resource with very little assurance over whether or not they will be misused.

One of the problems with grid credentials is that a large number of hosts typically require delegated access to them: schedulers, data management systems, information systems, and so on. The large number of systems holding delegated user credentials increases the probability that one of them will be stolen and misused.

Existing grid middleware solves the problem of credential misuse using proxy certificates. A proxy certificate is a copy of a user's long-term credential with a shortened life-span. Attacks on the grid middleware leading to credential theft have a reduced impact because they can only be used for a limited time.

It can be argued that short-lived credentials do not fully solve the problems with delegation. Proxies typically last for around twelve hours [178]. In that time a significant

quantity of sensitive data could be modified or stolen. Proxy certificates do not support revocation [178] so attacks cannot be stopped within the user credential lifetime. Even if the lifetime is short, an attacker can repeatedly steal the credential using the same attack until they are discovered and mechanisms put in place to prevent it happening again.

There are usability issues with using very short-lived proxy credentials, as they may expire within the life-time of the grid job. The problem is that the credential cannot be refreshed while the user is offline. This problem has been solved by having the grid middleware store users' long-term credentials [50]. When the proxy credential expires, the grid middleware can immediately create a new one. Although this overcomes the usability problems of credential expiry, the security cost is significant because attackers can now steal users' long-term keys.

Restricted credentials also suffer from usability problems. Restricted delegation can reduce the impact of attacks by limiting the privileges available to a user credential. However once the privileges attached to a credential have been restricted they cannot be re-gained without the user re-issuing a fresh credential. This creates usability problems because a delegated grid job may have insufficient access rights to execute successfully.

The previous discussion has demonstrated that proxy credentials are an inappropriate solution to the middleware problem because of security and usability limitations. A novel solution suggested in this chapter is to add additional security controls to prevent credential misuse. All grid jobs are pre-encrypted and trusted computing attestation is used to ensure that the data can only be accessed within a trusted execution environment. Credential theft no longer gives an attacker access to sensitive data because it is encrypted and integrity protected on disk during execution.

Exercising controls over data use on the grid can be seen as an example of a digital rights management platform. Trusted computing has received a negative reception because of its ability to be used to control content distribution by large providers such as the film and music industry [57, 9]. On the contrary, in this approach the user is empowered to control how their grid job is disseminated and protected on third-party systems. Grid data is given out with strict controls on the environment in which it can be used in order to

enhance security for the users of the system.

One of the key difficulties in building this architecture is maintaining interoperability with existing middleware systems. It is important to ensure that grid users continue to enjoy the performance and features of the existing grid middleware, while providing enhanced security for those that need it. Interoperability could be a major barrier to the introduction of the new architecture, so allowing an upgrade path is an important consideration.

Building trusted computing support into middleware systems is a time consuming and expensive process. Most existing grid systems are built around the assumption that the middleware is fully trusted. Adding trusted components would require substantial changes to the existing software design. One of the goals of the solution presented here is to decouple the security policy enforcement layer from the existing infrastructure, so it can easily interoperate with any grid middleware platform.

The rest of this chapter is structured as follows. Section 6.2 argues that the middleware problem will become an increasing issue in the future and a deterrence to some organisations joining the grid. Section 6.3 discusses related work in solving the middleware problem, and its limitations. Section 6.4 describes a novel solution to the middleware problem that works by pre-encrypting grid jobs and distributing the keys within trusted execution environments using a new component called the *key management service*. Section 6.5 discusses a number of security policies that can be used to implement digital rights management and mandatory access controls for the grid.

6.2 The Middleware Problem

Grid middleware is a weak link in grid security. This section presents evidence that grid middleware will be one of the key security problems facing users of the grid in the future. The middleware problem is caused by the high likelihood of vulnerabilities that result in grid users' data and credentials being compromised. Moving forward the middleware problem may lead to setbacks in the commercial adoption of the grid, and for other organisations who possess sensitive data.

To some extent grid middleware functionality mirrors, and seeks to replace, what was

previously the domain of the operating-system. Middleware allows grid jobs to abstract over the differences in underlying technology and provides a uniform interface to common grid functionality. Much like an operating-system, middleware has a high level of code complexity and large quantities of privileged code, a combination that in the past has led to operating-systems being one of the most popular targets for attack [172]. Grid middleware shares many features in common with operating-systems that are likely to make it the target of frequent and successful attacks in the future.

Software complexity tends to lead to security vulnerabilities [118]. The Globus toolkit grid middleware software currently consists of nearly two million source lines of code (see Figure 19). By comparison the Linux operating-system kernel consisted of over two million lines of code in 2005 [17]. A recent study demonstrated evidence of one security vulnerability for every 45,000 lines of code [4]. The complexity of grid middleware software is likely to lead to a high number of security vulnerabilities.

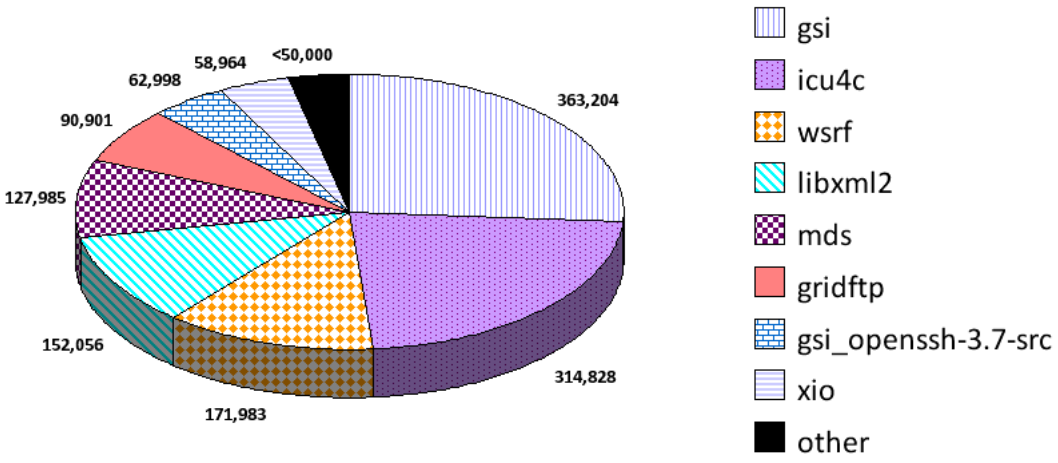

Figure 19: Source Lines of Code (SLOC) Breakdown of the Globus Toolkit Version 4.0.3 (Total SLOC 1,816,203)

Figure 19 shows a breakdown of the size of various software components within a recent version of the Globus Toolkit. Nearly three-quarters of the code is concentrated in the largest six modules, implementing critical functionality including security, information services, data management, and the web services framework. Although this analysis focuses

on the Globus toolkit, this fact demonstrates the complexity of the core functionality that is common to all grid middleware software.

Security functionality is not the same as assurance. The complexity of grid middleware results in a lack of assurance that undermines the security functionality it offers. For example grid authentication and authorisation systems can be bypassed by exploiting bugs, such as a buffer overflow, that allow an attacker to execute arbitrary code. In Figure 19 the largest of all components in the Globus toolkit is the security layer, known as the GSI [50]. This is indicative of the huge amount of research invested in grid security services, the complexity of which potentially undermines the security they offer.

Looking back at the trends of vulnerabilities in the Globus Toolkit [60], there has been a general increase between versions (see Figure 20 below). It is hard to determine the exact cause of the increase. One reason is likely to be the increasing code complexity as new features are added. These figures are currently far below that of a modern operating-system, but one could argue that grid computing has received relatively little attention from the security community thus far. As the grid is increasingly used to process high value business assets, these figures can be anticipated to grow significantly.

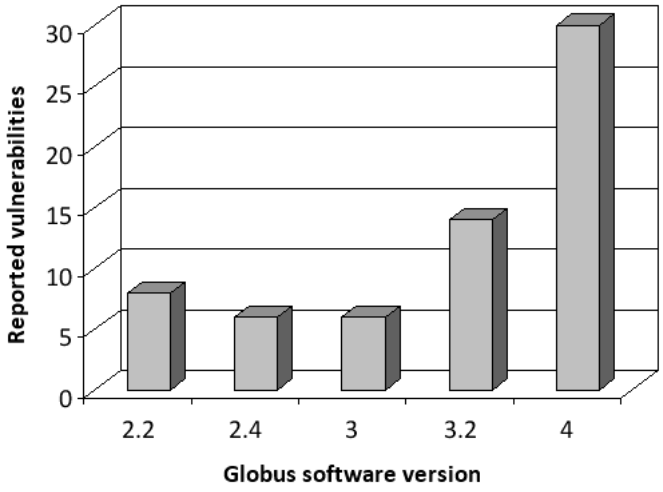

Figure 20: Globus Toolkit Security Advisories

Although this analysis has focused on the Globus Toolkit it is intended to be indicative of grid middleware in general. The Globus Toolkit is among the most mature frameworks,

with a large user community, and as a result is the most promising candidate for analysis. It is also one of the few such projects to carefully document security vulnerabilities.

Grid middleware is an excellent focus for attack because it tends, by its very nature, to be ubiquitous across many domains. Similar to an operating-system, the middleware may represent an attractive target for attackers because a vulnerability can be re-used to compromise many different systems, representing a relatively high return on the time invested in discovering the vulnerability.

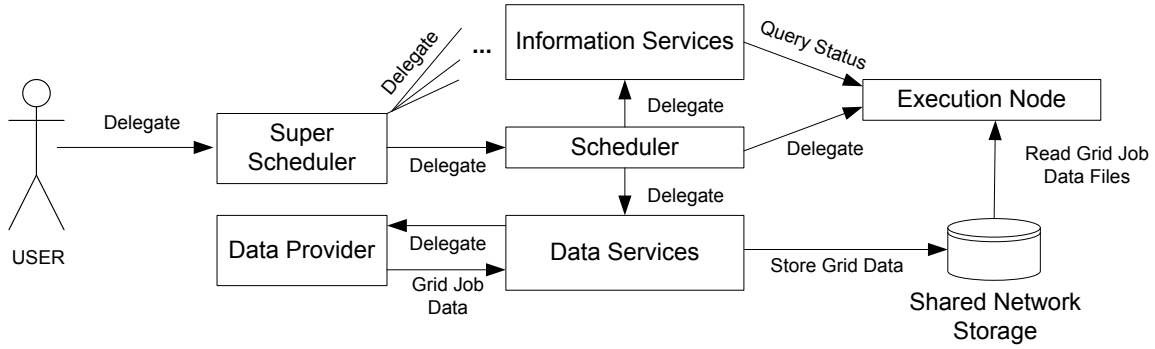

Figure 21: User Credential Delegation in a Typical Grid Middleware Architecture

It is a good design practice to limit the amount of privileged code in a system. Grid middleware software eschews this principle. Many components have access to sensitive user credentials. Figure 21 shows a typical architecture. The user delegates to a super-scheduler, which selects an appropriate grid service provider based on information services, and then repeatedly delegates to grid services which coordinate to execute the grid job. What should be clear from the diagram is that many components are trusted to handle delegated user credentials.

The grid middleware problem leads to severe difficulties in the protection of grid credentials. The variety of services that need access to credentials greatly increases the amount of software that the user has to trust. This problem is made worse by the fact that popular grid middleware systems typically store the credentials unencrypted on the hard-disk [158].

The grid middleware problem also leads to problems protecting grid data. As shown in Figure 21, network-based storage is an efficient way to build grid infrastructure. Many execution nodes can share the same storage area, avoiding the need to copy the grid job

data onto the execution node selected by the scheduler. The downside is that network-based storage increases the vulnerability surface of the grid provider. There is effectively no isolation of user data in the system, so that grid data integrity and confidentiality can be compromised by exploiting a vulnerability in any part of the service provider's grid infrastructure.

The grid middleware problem is not trivial to solve. It is easy to criticise the existing designs for a lack of isolation of sensitive data. The grid middleware, however, needs access to user credentials to perform delegation. If these credentials were isolated in some way, for example using encryption, then it would be impossible to delegate user credentials, and jobs could not be executed. In other words, strong access controls are not a viable solution because the grid middleware needs authorisation to access credentials and sensitive data.

6.3 Analysis of Existing Solutions

A number of solutions to the grid malicious host problem have been suggested in the literature. The proposed solutions generally fall into two categories. The first attempt to enhance the protection of grid data, and the second look at ways of securing user credentials. A common feature of these solutions is a failure to address the middleware problem, as will be seen in the analysis that follows.

Many current grid platforms support confidentiality and integrity for the transfer of grid data, and this might lead users to believe that their grid data is adequately protected. For example the Globus GSI [50] supports the use of SSL to provide an integrity protected and encrypted network channel for transferring grid data using the gridFTP protocol [5]. These security mechanisms only protect the grid job data as it is passed over the network. Once the data has been transferred it is stored unprotected on the grid service providers' infrastructure where it is left open to attacks on integrity and confidentiality.

One vision of the grid is as a large reliable storage platform. The idea is to have a remote data storage 'cloud' into which organisations can deposit data that can be read back at a later point. The gLite grid middleware architecture facilitates data encryption for remote data storage [149]. In this strategy the data is held offline on the grid and the

keys are retained by the data owner. Although this solution overcomes the middleware problem it is a rather specialised category of data grid. In the more general case of a grid that supports remote execution, this use of offline encryption is not possible.

Although virtualisation can provide enhanced isolation between grid jobs the protection is undermined by the middleware problem. A typical architecture is that used in the Globus Workspaces project [84] where the middleware and grid jobs are each run in their own virtual machine. The middleware virtual machine has access to the data and credentials for all the grid jobs running on the platform. If the attacker compromises the grid middleware they can compromise all the grid jobs and credentials available on the platform. The grid middleware is trusted not to break the job isolation, but the middleware problem undermines this trust.

The Trusted Grid Architecture (TGA) [96] attempts to solve the middleware problem using trusted computing attestation. The grid middleware first attests itself to the user, proving that it will protect data and credentials from attack. The grid job is then transferred over an encrypted and integrity protected network transport protocol. Grid jobs and middleware are run in virtual machines where they are protected against attacks from the host platform. Further details on the TGA are given in Section 4.3.

Attestation does not in itself enhance software security. If code is insecure, attesting it does not help the user because all it does is to prove that they are communicating with insecure software. This problem relates specifically to the solution used by TGA. The middleware is placed within the trusted computing base, despite the fact that the middleware problem means it is not trustworthy. Following attestation there is nothing to prevent compromise of the grid data and credentials. Attestation in itself cannot be a solution because the problem stems from a lack of isolation in the architectural design of grid middleware.

A straightforward way to enhance credential storage is to make use of trusted computing hardware. Researchers have used trusted computing to enhance the security of grid proxy credentials by protecting the private key using a hardware security co-processor [97, 102, 104]. The private key is protected by the hardware module, and only authorised software can access it to perform cryptographic operations (this research is discussed in further detail

in Section 3.4.2).

Grid middleware must be authorised to access user credentials. When the private key is protected by the trusted computing TPM, grid middleware is authorised to authenticate using the credential. The digital signature operation is performed inside the hardware chip, so the private key is never directly exposed to the grid middleware. This prevents the key being read and copied out to another platform, but it does not prevent mis-use of the key for unauthorised delegation and authentication. If the grid middleware is compromised an attacker can masquerade as another user by launching the attack from the system holding the credentials. It could be argued that these solutions protect the confidentiality of the private key, when what actually needs protecting against is the misuse of the corresponding credentials.

6.4 Solving the Middleware Problem

6.4.1 Grid Data Protection

Existing grid middleware does not sufficiently isolate sensitive data. A reasonable solution would be to re-design the grid architecture so that only trusted components are permitted to handle user data. The idea is to reduce the vulnerability surface by segmenting the grid middleware into a number of different trust levels.

Consider how grid middleware might be re-designed to enhance isolation between components. Data management servers might transfer data encrypted with cryptographic keys that are known only to the execution nodes. Execution nodes would be trusted to access the grid data, but data management services would not be. By segmenting the grid middleware in this way an attacker who compromised the data management service would be unable to access users' data.

The problem with isolating sensitive grid data is that it requires extensive changes to the existing grid middleware. There is evidence that security issues resulting from a lack of isolation in grid middleware are gradually being addressed. For example the Open Grid Forum, a body involved with grid standardisation, lists isolation as a key requirement [13]. However, introducing isolation throughout the existing grid services would be a major

change to the current architecture and is unlikely to occur in the near future.

Technical challenges remain in using trusted computing to remotely establish trust in service providers. Even if a more secure grid middleware architecture became widely available, it would be difficult to use attestation to determine whether it was installed and configured securely. Attestation can measure the software running on a single computer but it cannot directly measure a network of computers running a variety of grid services. In addition, it is difficult to measure complex variable data such as configuration information to determine that it is secure.

Even if these problems were solved (perhaps by consolidating multiple measurements) the complexity of the overall system would be too high to enable trust determination for a large-scale community. The number of alternative, valid measurements across service providers would be so large that maintenance would quickly become unmanageable, as every configuration change made by a service provider would result in the need to publish a new measurement. Attestation works well when it is used to measure small amounts of trusted code, it does not work well when it is used to measure an entire network infrastructure.

A final problem is interoperability. As the onus is placed on grid service providers to implement a secure architecture, the likelihood increases that the grid might become either highly variable in the security protection it offers, or highly segmented to meet the demands of users with varying security requirements and expectations.

In light of these problems, improving the security of the existing grid middleware through greater isolation of sensitive data is not a feasible solution at this stage. The time and difficulty of re-engineering the existing architecture imposes a high cost. The complexity of the resulting software means that it would be difficult to use trusted computing to establish trust in the overall infrastructure.

The remainder of this chapter describes the design of a grid middleware security architecture that overcomes these problems. The security layer is decoupled from the existing grid middleware so that no re-design is necessary. The service provider is free to implement the software of their choice (on the grounds of manageability and performance, for example) without compromising security.

The job security manager architecture, initially described in Chapter 4, is expanded to protect grid job confidentiality and integrity as it moves across untrusted grid middleware software. A single trusted computing attestation measurement is sufficient to determine trust in the overall system, and the trusted software is simple enough that a large degree of trust could be placed in it.

If the user only requires integrity protection for their grid job, and has no confidentiality requirement, the problem can be partially solved using the job security manager architecture that has already been presented in the previous chapter. The grid middleware is given access to the unprotected grid job data. If the user's data is tampered with, it will be detected when the results are retrieved. The job attestation service will attest to a measurement of the grid job, and the user can detect that the measurement does not match the expected value.

The job security manager architecture has limitations because the integrity of the grid job can only be verified while it is executing. The job attestation service runs as part of the job security manager, a virtual machine that is deployed alongside the grid job. The virtual machine terminates when the grid job has completed, so integrity measurements can no longer be made. It is common for results to be staged out by the grid middleware once the grid job has completed execution to avoid unnecessarily tying up execution resources.

A solution to providing offline integrity is to have the secure storage service store a digital signature of the grid job while it is shutting down. The digital signature is written out as a file on disk. The digital signature is then staged out by the grid middleware along with the other grid job data files. The receiver can verify the integrity of the grid job by examining the digital signature and verifying the attestation measurements it contains.

One shortcoming of offline attestation is that it cannot directly protect against replay attacks. The measurement generated by the storage service is signed using a trusted attestation identity key in exactly the same way as a regular online attestation challenge. Online attestation protocols include a nonce challenge provided by the verifier in the digital signature to prove that the current measurement is live and not a replay of a historic value. The nonce cannot be included in the digital signature used for offline attestation because

the verifier might be offline and unable to take part in a challenge and response handshake.

A suitable alternative to avoid replay attacks in offline attestation might be to include a trusted timestamp in the digital signature. There are difficulties with this approach however: all parties must have access to a consistent source of secure time. This is difficult to guarantee in a distributed environment like the grid where systems are under different administrative control. Alternatively, for many grid jobs it might be possible to confirm successful termination by examining the output data. For example many applications write status information to a file once they have successfully completed processing.

Running grid jobs in a trusted execution environment is one means to achieve confidentiality in addition to integrity protection for grid data. In Chapter 4 a solution was described where grid data is transferred directly into the job security manager over an encrypted and integrity-protected network channel. Unfortunately this solution is not suitable for use on the grid due to the need to support data staging while the grid job is offline.

One way to enable confidential data staging is to run the grid data transfer service within the job security manager. This solution is unsatisfactory because of the middleware problem. The software responsible for staging grid data is too complex to be part of the trusted computing base. For example, the implementation of the GridFTP protocol used by the Globus toolkit consists of approximately 100,000 lines of code, and it only makes up part of the data transfer service.

A solution that is built on throughout the remainder of this chapter is to have sensitive data pre-encrypted before it is downloaded. The use of pre-encrypted grid jobs facilitates compatibility with existing grid middleware. The data management software can make use of data staging, allowing the grid job to be downloaded prior to execution. This approach maintains compatibility with existing data management technology such as data striping and replica management. It makes no difference that the data is encrypted because these services work with arbitrary binary data.

The security architecture is overlayed on top of the existing grid middleware services so that no changes are necessary. The job security manager is distributed unencrypted alongside the job data. The grid execution management services schedule and execute the

job security manager like any other job. Once it starts executing the job security manager is responsible for retrieving the keys and decrypting the grid job prior to execution. The job security manager is distributed with the grid job and does not need to be pre-installed by the service provider.

The security infrastructure is completely decoupled from the grid middleware. Moving the security software from the middleware into the job security manager is a powerful feature because it facilitates interoperability with existing grid middleware platforms. The user can run a grid job with strong confidentiality protection on any host on the grid simply by distributing it alongside the job security manager. This opens up the grid so that the user can make the best use of available resources without being restricted to providers that support the necessary secure software.

A new set of infrastructure components must be introduced to handle secure negotiation of the grid data encryption keys. The keys are held by *key management services* that are owned and managed by data owners. This infrastructure is completely separate from the grid middleware. The job security manager contacts the key management service for the keys so it can decrypt and execute the grid job. The key management service will only release the keys following successful attestation of the job security manager. The policy enforcement means the user can ensure the job can only be executed within a trusted execution environment.

The malicious host problem makes it difficult to protect user credentials when they are delegated on the grid. With these improvements the secrecy of user credentials is no longer wholly relied upon for security. Impersonating a user allows the attacker to download the encrypted grid job, but not to steal the data it contains. The key management service will only release the keys to a trusted job security manager. Once they are released the keys are stored in protected storage under the trusted computing TPM so they cannot be stolen from the host platform.

Using the job security manager is essential to make trusted computing attestation feasible. In other approaches attestation causes problems because complex grid middleware software must be measured. Placing the security enforcement software within the job secu-

rity manager means that only a single attestation measurement is necessary to determine that integrity and confidentiality protection will be enforced. The grid middleware does not need to be measured as part of attestation since it is not trusted. The job security manager is designed to be small and simple so that attestation measurements rarely change. The measurement will not vary between grid jobs provided they use the same job security manager.

6.4.2 The Key Management Service

The primary purpose of the key management service is to control access to data on the grid. The grid middleware is not trusted. Instead the grid job is encrypted before it is accessed by the grid middleware. The key management service authorises release of the decryption keys according to a security policy. The keys are delivered into a trusted execution environment running on the execution node where the data is protected from attack by the host platform. The key management service is a critical component because it is wholly responsible for policy enforcement.

Anyone can run a key management service but it will not necessarily be trusted. The key management service helps broker trust on remote systems. It will typically not run on remote infrastructure alongside existing grid middleware, but instead in a secure environment belonging to the data owner where it can be protected from attacks. Trust is implied when decryption keys are deposited in a key management service which is then relied upon to control access to those keys. A data owner will typically only trust a limited set of key management services.

The steps involved in executing an encrypted grid job are as follows (see Figure 22):

1-3. The job security manager is downloaded (along with the encrypted grid job) and executed using existing, unmodified grid middleware services.

4. The job security manager performs an authenticated boot process and attests itself to the key management service to prove it will securely manage the keys once they have been released.

5. If attestation is successful, the key management service transfers the decryption keys and hashes for the requested grid job over a secure communications channel.

6. These keys are then used to decrypt the encrypted job data, and the data hashes are verified to enforce integrity. The grid job then runs within a trusted execution environment where it is protected against attacks from the host platform (using the approach described in Chapter 4).

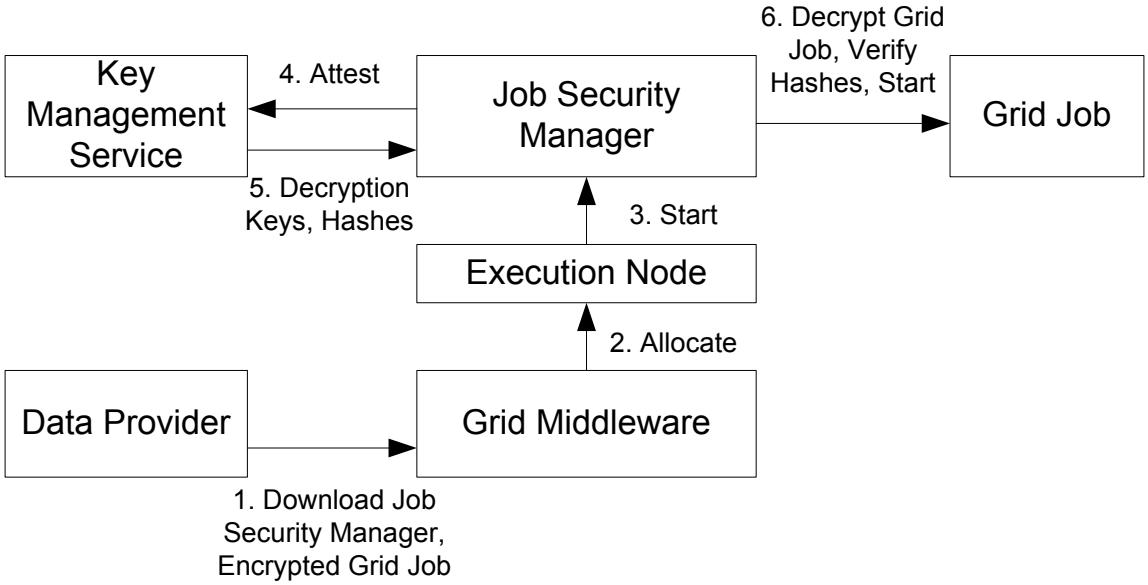

Figure 22: Solving the Middleware Problem

The job security manager is trusted to hold keys and other data securely. One of the intentions behind the design of the architecture is to minimise the likelihood of security vulnerabilities in the trusted parts of the system. This is achieved by offloading functionality to untrusted software wherever possible. The majority of the complex functions such as data staging and execution management are left to the grid middleware.

The integrity of the grid job must be verified to prevent the introduction of a Trojan horse into the encrypted data. Integrity is verified using hashes of the data. The job security manager generates a hash of the grid job prior to execution and compares it to the expected value received from the key management service. If data tampering is detected

the job security manager will refuse to execute the grid job, so it is not possible to introduce a Trojan horse.

There might be multiple key management services involved in executing a single grid job. The key management service authorises access to data. A grid job might be composed of data that comes from multiple organisations. For example, a grid job might be comprised of software from the grid user, a data set obtained from the virtual organisation and yet another data set obtained from a third-party for a license fee. Each data owner might trust a different key management service in a separate physical domain. Before the grid job can start, each key management service needs to be contacted in turn to obtain all the decryption keys.

In a large grid the user may be unaware which key management service to contact to obtain access to some encrypted data. To avoid this problem, each encrypted data item is tagged with the location of the corresponding decryption key. This metadata is stored in a separate file alongside the encrypted data on the data server. The user requests the middleware to download the metadata like any other data file using data staging services. When the job security manager starts up it reads each metadata file in turn and retrieves the location of the key management service. This approach removes the responsibility from the user to locate decryption keys. It also allows data owners to alter the location and configuration of the key management service over time without needing to inform users.

Data owners are free to implement many different security policy enforcement strategies. For example the data owner might want the key management service to perform host authentication before releasing the decryption keys, or a more complex digital rights management system (see Section 6.5). The policy enforcement and decision point is implemented entirely within the key management service so it can be customised to the needs of the data owner.

The architecture attempts to balance the need for customisation against a desire for standardisation and interoperability. The key management service policy will be configured to trust only a limited set of job security managers. If every grid job security manager was unique attestation would become too difficult to manage. Using standardised protocols

to pass authorisation requests to the key management service, such as the web services protocols, can ease the need to support several different job security managers.

Across projects, however, different functionality might be required. The design of the grid job security manager might be altered to trade off performance against security, for example. There is likely to be a number of competing designs for the job security manager that will adopted by different virtual organisations and projects.

The key management service is not attested to the job security manager (see Figure 22). A benefit of this approach is that the job security manager is interoperable with the many different key management service implementations that are likely to exist in a large cross-organisational grid.

In summary, the key benefits of using an architecture based on key management services are as follows:

- **Security assurance**: The grid middleware problem is overcome because the grid infrastructure is no longer trusted to protect users' data. The reduced complexity of the job security manager and other trusted components significantly decreases the likelihood of security vulnerabilities.

- **Secure credentials**: Credential theft no longer results in the compromise of sensitive data, overcoming the difficulty of protecting grid credentials in existing grid architectures.

- **Interoperability**: The security enforcement will work with any grid middleware layer and grid job without modifications.

- **Flexibility**: The user is free to choose the security software and interfaces based on their individual requirements. Equally, the service provider is free to choose their own middleware software implementation without impacting on users' security.

- **Manageability**: The trusted software is small and simple, so the remote attestation measurement is unlikely to change frequently over time or across projects and organisations.

6.4.3 Problems with the Key Management Service

Even though user credentials are not completely trusted, a significant security weakness arises if the key management service does not authenticate the user. Although the architecture minimises complexity, it is still likely to contain bugs. If the trusted software or hardware contains a vulnerability, an attacker who exploits it can steal the decryption keys. Assume an attacker compromises a computer containing such a vulnerability. The attacker can request the decryption keys from every key management service available on the grid, and potentially compromise a significant quantity of data before they are discovered. Without additional defensive measures the impact of a vulnerability will be very high.

Additional countermeasures can minimise the impact of a vulnerability within the trusted hardware and software. The job security manager should be required to authenticate the user before it receives the decryption key. The security architecture does assume that the grid middleware is untrustworthy, and that grid credentials can be stolen, so this may at first seem to be of little use. Even though grid credentials can be stolen, however, it is unlikely than an attacker could steal the grid credentials of every user on the grid. The impact of the attack is reduced using this countermeasure because the attacker can only steal decryption keys authorised for release to the compromised user accounts.

One problem with requiring the job security manager to perform user authentication is that it might require tight integration with the grid middleware. Standard web services-based protocols have been devised for credential delegation on the grid [48]. Provided these are supported the job security manager should be compatible with any underlying grid middleware implementation. However implementing a web services delegation client within the job security manager does raise the code complexity and likelihood of vulnerabilities somewhat. In light of these concerns, the decision about whether to require user authentication is left open as an implementation choice.

It is also important to think about how attacks can be prevented once they have occurred. Minimising the impact of a large-scale attack, as outlined above, will not stop an attacker who is determined to attack a single user or organisation. To protect against these attacks it is important that the impact of vulnerabilities can be minimised once they are

discovered.

Key revocation provides a potential solution to this problem. The key management service verifies the identity of the host running the job security manager. The digital certificate of a compromised host can be revoked. The key management service should check for revocation before releasing decryption keys. Similarly, if a vulnerability is discovered in the trusted software, the digital certificate containing the attestation identity can be revoked. These measures allow the impact of an attack to be mitigated while existing systems are patched and upgraded to eliminate the vulnerability.

As well as examining the advantages, it is important to look at some of the disadvantages of using the key management service. Chief among these is the problem of key management. Key management in distributed systems has long been considered as an independent research topic [56]. The security policy controlling the release of the shared data can quickly become unmanageable, and the need to resolve policy conflicts arises.

A global object naming system for each data object is required to locate the appropriate key management service. Data owners are free to implement key management services in any way they see fit, so the latest research on the solution to these problems can easily be incorporated into the architecture (see Chapter 7 for further details).

One key barrier to the introduction of the key management service architecture is the lack of availability of data in a suitable encrypted form. Many users of the grid have large catalogues of data and assembling all of these into an appropriate encrypted format might be prohibitively expensive. The potential cost is a significant barrier in adopting such an architecture on a large scale.

To overcome this problem the architecture allows for the gradual introduction of trusted systems on the grid. Existing grid credentials are not replaced, and they can be used to access existing data repositories. Unprotected data can be mixed with encrypted and integrity-protected data where necessary, provided the risks of doing so are appropriately managed. The ability to continue to use existing data repositories while more sensitive data is introduced in protected form considerably improves the feasibility of the architecture. Further solutions to this problem are examined in Chapter 7.

It is important to avoid single points of failure in distributed systems. A data owner must take resilience into account when deploying key management services. If the host running the key management service fails then the impact would be significant because none of the data it holds keys for could be accessed. It is possible to provide multiple duplicate key management services for failover but this can introduce concerns that the keys might be compromised. Researchers have addressed this problem in other architectures by splitting and distributing the decryption key into multiple pieces so that only a subset of the keys are needed to access the data [16].

There is potentially a large security risk when using the key management service because all the decryption keys are held in one place. The assumption is that a strong level of protection can be put in place because it is situated within the data owner's trusted network. Physical, network-based, and other countermeasures can be managed and enforced by the data owner. If necessary the risk of compromise can be reduced further by using split decryption keys [16]. These split keys can then be held by key management servers in multiple locations while mitigating the risk if one of them is compromised.

Some data cannot easily be pre-encrypted ahead of time, especially if it is highly dynamic. For example, consider grid data held in a large distributed database. Many databases encrypt data stored on disk and decrypt the data in response to a user query. Database records can be obtained while the grid job is running by transferring the query results over an encrypted network protocol. A problem arises if the data needs to be pre-staged. If the grid middleware was authorised to query the data it would gain access to the unprotected database records.

Despite that, highly dynamic data is less likely to need to be staged prior to grid job execution by its very nature — the grid job is likely to need access to the most up to date copy during execution. Pre-encryption is not feasible with some types of data, especially when they are held in proprietary storage systems that cannot be adapted to work with the key management service architecture.

6.5 Digital Rights Management for Grid Computing

6.5.1 Trusted Subgrids

The key management service enforces digital rights management controls for grid data. Data access is authorised by releasing decryption keys for data, subject to a security policy. The grid job is encrypted not just for confidentiality, but also to support access controls. The use of encryption ensures that the policy is enforced wherever the grid job is accessed throughout the grid.

Digital rights management is often seen in a negative way, as a technology that unfairly prohibits legitimate data use [57, 9]. On the contrary, the use of digital rights management can empower users by allowing new forms of working that would not otherwise be possible due to security concerns. Digital rights management is used by the key management server to allow the job owner to maintain ultimate control over their data.

Digital rights management is powerful because it ensures that the data access policy is consistently enforced throughout the grid. Digital rights management ensures that the checks are consistently applied. Any host that wishes to access protected data must obtain authorisation from the key management service before it can retrieve the decryption keys.

The key management service is designed to be flexible. Users are free to choose a security policy that meets their individual requirements. The security policy can restrict the users, resources, and software that is able to access grid data. Several example security policies are described in the remainder of this chapter.

In a large grid there will be varying levels of trust between the members. It is infeasible to think that organisations will trust the entire grid. Members of the same virtual organisation might trust each other, but not organisations outside the virtual organisation. Enterprises might trust only those cloud service providers with which they have a contract in place. One could say that in the grid the virtual organisation forms a trusted boundary. Parts of the grid outside of this boundary will typically not be trusted.

Delegation makes trusted boundaries difficult to enforce. When a grid job is executed on the grid it generally runs outside the owner's administrative domain. The job owner has

no direct control over the external organisation. The service provider must be trusted to ensure the grid job executes within the trusted boundary.

Most current grid architectures use host certificates to help enforce a trusted boundary. Host certificates are typically signed by a certificate authority that is trusted by the virtual organisation. Once a host obtains a valid certificate it is admitted to the trusted boundary of the virtual organisation. Grid services typically perform mutual host authentication before authorising grid credential delegation [50]. Grid hosts that are not within the trusted boundary will fail authentication, so delegation cannot take place.

Authentication of host certificates is insufficient as a means to enforce a trusted boundary. Host authentication offers no guarantee that the host is secure. The middleware problem significantly raises the likelihood that hosts on the grid may be compromised. This is a significant problem because once a host has been compromised it cannot be relied upon to enforce a trusted boundary.

Trusted boundaries are not only broken by attacks on the host platform. The problem is that in most current grid implementations the service provider, and not the job owner, decides which host certificates to trust. For example imagine a company that has outsourced part of its business processes to an external grid provider. The service provider may decide to delegate some of the jobs to a partner without obtaining approval from the customer. In fact one of the advantages of the grid is that the service provider can choose to execute the job where it is most appropriate. This is fine when performance is the primary concern, but not when security is an over-riding factor.

One possible solution to the enforcement of a trusted boundary is the notion of *subgrids*. Subgrids are a trusted subset of the grid with a strong boundary. Subgrids are defined and controlled by the data and job owners, and not the service providers. Sensitive data cannot be leaked outside of the subgrid even if the service provider attempts to do so. The hosts within the subgrid are measured using trusted computing attestation to ensure they can be trusted to enforce the subgrid.

The key management service can be used to enforce subgrids. Each time a host needs access to protected data it must obtain authorisation from the key management service. The

owner organisation can therefore configure a centralised policy within the key management service defining hosts on which the grid job can safely execute. The policy allows the owner to define the trusted boundary.

The owner of a grid job assumes the job is safe provided it executes within the trusted boundary. This assumption relies on two key requirements. Firstly, the hosts within the trusted boundary must secure the grid job against attacks. Secondly, the hosts within the trusted boundary must prevent sensitive data from being leaked outside. A grid host must provide protection against the malicious host problem to ensure the boundary cannot be bypassed.

The key management service uses remote attestation to determine that an execution host can be trusted before authorising data access. The checks made by the key management service prior to releasing the decryption keys must be comprised of the following policy requirements to enforce a subgrid:

1. The identity of the host on which the job security manager is running and its physical Trusted Platform Module are within the subgrid;

2. The job security manager and the rest of the trusted computing base is trusted to enforce the subgrid.

The host and the TPM can be authenticated by verifying the attestation identity key of the platform's TPM. The job security manager can be attested using the process described in Section 4.6.3.

Subgrids are useful when a member of a virtual organisation does not completely trust the other members. Assume for example the data provider is a hospital trust, and the user is conducting a medical trial. The hospital trust has a duty of care to ensure that patient data is protected. The digital rights management controls can enable this type of research to happen in a large distributed system by enforcing a trusted subgrid around those systems authorised to participate in the trial. Without strong data protection trials would need to take place on isolated networks that are expensive and difficult to maintain.

In more complex scenarios there might be multiple levels of trust in the grid. Organi-

sations might belong to multiple virtual organisations, some of which they trust more than others. For example an enterprise might trust a grid made up of its business partners more than it trusts an external cloud provider. To deal with different levels of trust subgrids can be implemented as multiple overlapping subsets of the grid. Grid jobs might be restricted to particular subgrids depending on the sensitivity of the data they contain.

The key management service can implement policies for subgrids supporting multiple levels of trust. The policy can be configured on the level of individual data items. Data can be grouped according to sensitivity or areas of interest. For each group of data the policy can define a different subgrid that is authorised to execute or receive that data. Thus by varying the granularity of the security policy enforcement it is possible to implement various data protection schemes.

6.5.2 Recipient Keys

It is important to think about how to get data securely back out of grid systems. The previous section described how grid data can be kept within a subgrid. It is equally necessary to enforce a strong trusted return path for the results. Combining these features together creates a secure chain where both the incoming grid job and the outgoing results are contained within the trusted subgrid. Stakeholders can then set a security policy that will be applied consistently over an entire grid workflow.

Pre-encrypting the grid data creates problems for key management. The recipient may receive encrypted results data without knowing the decryption key. The solution used by the Trusted Grid Architecture [96] is to have the grid job send the results out over an encrypted channel during execution. The disadvantage of this approach is that the grid data cannot be staged out by the grid middleware after the grid job has completed execution. The grid middleware has no way to access the decryption keys and without these the recipient will be unable to decrypt the results.

An extension to the functionality of the key management service can enable the secure staging out of grid job results. The security policy for a data item is extended to include one or more authorised recipients. Recipients can be specified for the entire grid job or

for individual pieces of data. The key management service ensures that the data is only available to the intended recipients by exporting the data encryption keys for those users.

Access to the data encryption keys is controlled by wrapping[4] them for the intended recipient. When a key is wrapped it is encrypted in order to control distribution. For example assume each recipient has a long-term private key and the key management service knows the equivalent public keys. The key management service creates the recipient keys by encrypting the data key for each recipient's public key in turn.

The recipient keys are handed to the job security manager along with the hashes and data encryption keys. The recipient keys are stored as files on storage that can be staged out by the grid middleware along with the grid job results after the grid job has finished. The recipient can then unwrap the data key and use it to decrypt the results. If the recipient is an impostor they will not be able to unwrap the key to decrypt the data, since they do not possess the intended recipient's private key.

Encrypted data staging maintains full interoperability with existing grid middleware. The job security manager writes the recipient keys out to disk storage. The recipient keys are stored on the file system so they can be read by the grid middleware. At any time during or after job execution the grid middleware can stage out the encrypted data together with the wrapped keys. The encrypted data and wrapped keys can be transferred using existing grid middleware without any modifications.

The recipient keys are generated outside of the job security manager to permit flexibility in the cryptographic mechanisms used to wrap the keys. The key management service wraps the keys on behalf of the job security manager. For example the cryptographic key length could be altered without needing to redistribute a new job security manager implementation and manage a new set of attestation measurements. A secondary benefit of external key generation is that the complexity of the job security manager is further reduced, lowering the likelihood of security vulnerabilities.

The list of recipients is likely to alter on a job-by-job basis. The key management service security policy might be configured to allow hundreds of authorised recipients. Distributing

[4]The term *wrapping* is used here as a term to denote an encryption key encrypted under another key for the purposes of secure distribution.

those recipient keys for every job would result in unnecessary overheads. Instead the job security manager is configured by the user to request a particular set of recipients. If it is authorised by the security policy, the key management service will give out the recipient keys that were requested.

Care must be taken that the attestation measurement of two identical job security manager implementations remains constant. The requested recipient keys can be configured by the user in a file contained within the job security manager. This configuration is not attested as part of the authenticated boot measurement. The configuration is not security-relevant so there is no need to attest its value. If the user attempts to specify a recipient who is not authorised to retrieve the data, the wrapped key will not be given out by the key management service. If a valid recipient adds themselves to the list without the authorisation of the job owner, then they will get a copy of data that they are already entitled to receive, and could obtain independently, so this does not break the security policy.

There is a great deal of flexibility in the way recipient keys can be used. The key management service security policy can be configured to implement a number of different protection schemes. The recipient key could be held by an organisation or a single person. Alternatively it could be held by a data repository. In this approach, all the grid job results are staged out to a single data repository. The repository decrypts the grid job results and then uses standard access controls to authorise access to the results by end-users.

6.5.3 Mandatory Access Controls for Grid Data

The Need for Mandatory Access Controls

One of the challenging aspects of grid security is the large number of stakeholders involved. The stakeholders might include the owner of the grid job, the (potentially multiple) owners of the data comprising the grid job and the owner of the grid resources. Each of these participants might want a say in defining how their data and resources are used. An effective grid security framework must allow each party to enforce their own security policy.

In a mandatory access control system the user cannot alter the security policy. Instead

the policy is defined by the various stakeholders involved. Even though users may be given access to data, the ultimate control over its use remains in the hands of the data providers.

The policies described previously in this section can be seen as a form of discretionary access controls. The grid job owners must be trusted because there is nothing to prevent a grid job transferring data to a third party during execution. Recipient keys and subgrids are only used to protect against attacks originating from the host platform or the grid middleware. An alternative approach involves using a mandatory access control policy that cannot be over-ridden by the job owner.

Not all users in a large virtual organisation are likely to be fully trusted. One of the advantages of mandatory access controls is that the policy cannot be subverted by malicious members of a virtual organisation. The policy is enforced by a security reference monitor that is designed to resist attacks. As a result members of the virtual organisation are unable to bypass the security policy.

Trusted computing attestation can extend access controls in an exciting way. Traditionally mandatory access control policies have been defined based on a classification of subjects and objects given read and write access [14]. Attestation allows the policy to define the type of data access permitted at a more fine-grained level. For example, the user can be restricted to accessing the data with a particular software package that only permits certain usage, like playing a film a certain number of times. Attestation allows mandatory access control policies to be specified in terms of domain-specific behavioural controls.

Mandatory access controls will be useful for the grid because they allow data owners to control how their data can be used. For example the data provider might collect consumer information, and customers may only be allowed to calculate aggregate statistics on those values in order to maintain privacy. This can be achieved by giving customers a database together with a client application that extracts aggregated values from the database. The security policy forces users to access the database via that particular application. A similar approach can be used to control access to non-anonymised patient records in a medical trial.

Mandatory access controls can help prevent benign security failures as well as deliberate

attacks. An important category of security failures on the grid is caused by accidental user errors. For example a grid job might include unpatched software containing security vulnerabilities. As another example, users may be socially engineered or otherwise tricked into running a Trojan horse. By controlling the software used to access their data, owners can help protect against attacks resulting from software vulnerabilities and malicious code.

Data owners and other stakeholders have a vested interest in preventing security compromise caused by user error. As the grid grows in popularity and the number of users increases these problems will become more and more prevalent. Mandatory access control policies allow the stakeholders to prevent untrustworthy software from executing on grid systems.

Service providers can also make use of mandatory access controls to mandate the software users can access on their systems. The service provider might want to mandate, or at least audit, the software that runs for legal and compliance purposes under their obligation to monitor usage of the network. For example illegal activities such as computer hacking tools or software involved in propagating child pornography could be prevented by controlling the software users can execute.

Service providers can enforce mandatory access controls by running their own key management service. In this model all software is under centralised control by the grid authority. For example, the security policy might be administered by an entity like the UK National Grid Service [58]. The policy would be configured so users can provide their own input data, but they can only choose from the pre-defined list of applications selected by the service provider. These might correspond to a set of pre-installed virtual machines containing the most up-to-date version of eScience applications, for example.

Implementing Mandatory Access Controls

The key management service can enforce mandatory access controls. The security policy defines the software permitted to execute within the subgrid. Before giving out the decryption keys the key management service examines the grid job to determine whether it is authorised to access the data. The grid job is first measured by the job security manager.

The key management service receives the measurements and examines the policy to determine the access rights for the given software. Data will only be released if the software is authorised to process it.

Depending on user requirements both a whitelist and a blacklist approach is possible. In a whitelist the policy would list the complete set of allowed applications. A blacklist would identify software that is specifically prohibited from executing. These two approaches trade off complexity of policy configuration against security. Whitelists are harder to manage but ensure that Trojan horses are not allowed. Blacklists are easier to manage because users are not impeded from using new software, but there is a window of opportunity for attacks between policy updates.

Policy enforcement is considerably easier if the grid job is distributed in a single package. Assume that the grid job is a set of packaged data and software in a single virtual machine image supplied by an individual data provider. The mandatory access control policy then states that the grid job cannot be instantiated alongside any other data files. It is sufficient to verify that the grid job has not been substituted or run alongside a Trojan horse before authorising access to the data.

The power of mandatory access controls can be seen more clearly once a grid job is split into multiple components. The data might come from several data providers, each of whom specifies a different policy for the protection of their data. For example, consider a grid job that is used for medical research. An experimental data model is provided by a research company, and patient records are provided by a hospital trust. The research company has a key management server configured to allow any experimental model to be run provided it is kept within a trusted subgrid. This allows them to participate in several medical trials at the same time. The hospital trust also has their own key management server controlling access to the patient records. The hospital trust policy requires patient data to be processed within the research company by a specific experimental model that cannot be altered by the user. In this example there would be two key management services implementing mandatory access controls on behalf of each of the respective stakeholders.

A straightforward approach to grid job integrity measurement leads to problems when

data owners do not trust each other. Consider what happens if the key management service pushes the expected data hash values down to the job security manager along with the decryption keys. The downside of this approach is that the key management service loses visibility of grid job data that belongs to other data owners. Each key management service pushes down the hashes for the data it controls. If the grid job contains data from multiple owners, each owner will only verify the hashes for their own data. Unless every data owner trusts the other, there is a danger that there might be an unseen Trojan horse in the unverified data.

Instead, the job security manager should accurately report all the data making up the grid job. The key management server relies on this information to make policy decisions. If some measurements were omitted Trojan horse software could be hidden within the grid job. To ensure the information is comprehensive the job security manager reports measurements of the grid job in a manifest. The manifest lists all the grid job data files and the corresponding data hashes, as well as the requested recipient keys for each data item. The manifest is passed over the secure channel established with the key management service to prove its authenticity and integrity. The key management service can make an informed policy decision based on analysing the contents of the manifest.

Mandatory access controls should be resilient against attacks by corrupt users or service providers. To prevent users spoofing the data integrity measurements, each data owner verifies the hashes of their own data files before authorising release of the decryption keys. Since the job security manager is attested, the data provider has confidence that these hashes will be correctly verified after the data is decrypted. If a data provider lies about the integrity values, the job security manager will detect a discrepancy in the data when it performs the integrity check prior to execution of the grid job.

Security attacks might be possible if the manifest does not capture sufficient information. In cases where the grid job is split into multiple data partitions it can be important to know which one is the bootable image. Otherwise an attacker might be able to run a job with two applications, one of which is a Trojan horse. The key management service might authorise data release to the genuine software without being aware that the Trojan horse will execute

in its place. To prevent these attacks the manifest marks one data file as the bootable image. The job security manager ensures that this is the image executed first. It does this by restricting the initial read access to the bootable data image within the virtualised storage device layer.

The basis of the effectiveness of digital rights management is secrecy of the encryption key. Even when the data is accessed by the user they must not have access to the decryption key. If an attacker gains access to the keys, the data can be decrypted and accessed without the digital rights management controls in place.

Recipient keys offer a potential way to bypass the digital rights management controls. An attacker can obtain the decryption keys for the data if the policy defines them as the recipient. This will give the attacker access to the decryption key, allowing them to bypass the digital rights management protection. To prevent these attacks the key management service must ensure that the recipient list is empty for all data that is under mandatory access controls.

6.6 Case-study: Cloud Computing

This case-study explores how digital rights management controls can be used to restrict access to sensitive patient data by a company performing medical research. The company wants to make use of an external compute cloud to provide high-performance data analysis.

The company has a legal obligation to enforce controls over the appropriate use of patient data. A trusted grid solution will be used to mitigate the following threats:

1. A company employee deliberately or accidentally leaks patient data to an unauthorised user;

2. Another user of the cloud (for example a rival company) gains unauthorised access to patient data.

The company employees are split for the purposes of access controls into two main groups (see Figure 23). Modellers are responsible for developing new software analysis tools but do not have access to patient data. Analysts are permitted to analyse patient data, but

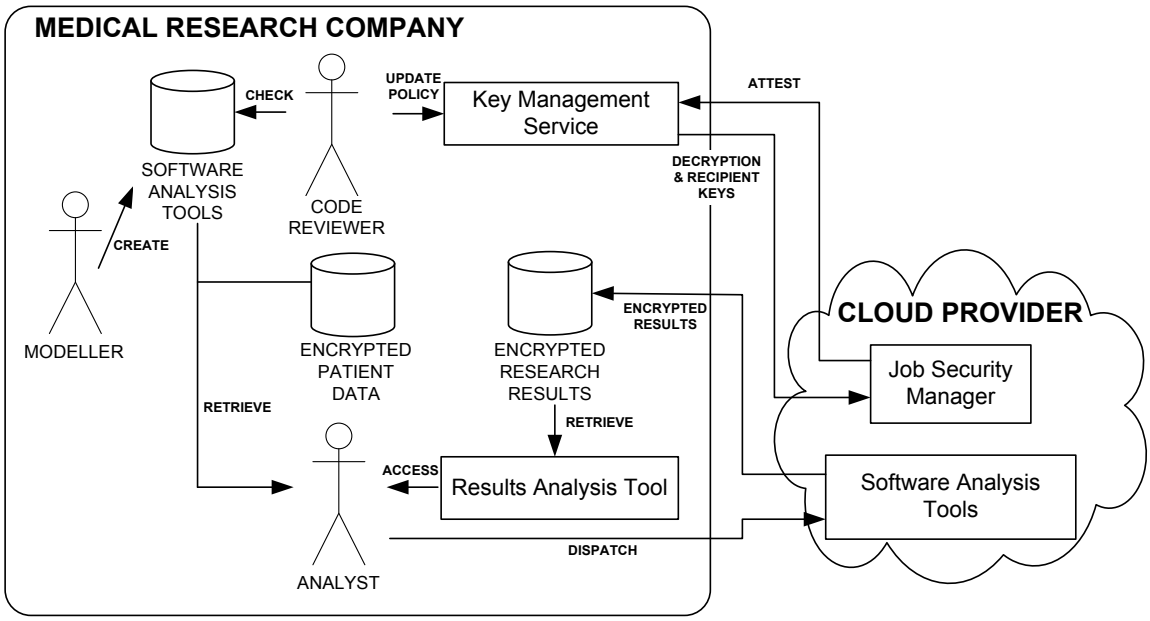

Figure 23: Cloud Computing Case-study

are restricted to do so using the approved software analysis tools. Analysts dispatch this software onto the cloud along with the encrypted patient data. The results are then sent back to a company database to which the analysts are permitted restricted access.

A key management service together with a job security manager can be used to provide strong protection for the confidentiality of patient data using features described in Section 6.5. Mandatory access controls can be used to ensure only permitted software analysis tools can be executed on the cloud. Subgrids can be used to ensure analysts are only able to submit grid jobs to a cloud provider approved by the company. The cloud provider must support execution of a job security manager to provide confidentiality protection against other users of the cloud. Finally, recipient keys are used to ensure the results are held in a database that enforces access controls on the data.

Before the patient data can be securely accessed from the cloud it must be pre-encrypted. The company does this by encrypting various data-sets and making them available on a public data server. The decryption keys are kept strongly protected within an internal key management service. The high-level key management service policy is established by trusted administrators at this stage. This policy consists of the subgrid and recipient keys,

and will be discussed further later.

Code reviewers are responsible for updating the mandatory access control policy. The modeller develops new program code which is approved by a reviewer to ensure it does not contain backdoors or unintended functionality that could result in data leakage. Once the new program is approved the reviewer updates the policy of the key management service to allow the software to execute on the cloud. This is done by adding the hash value of the software analysis tool to the policy for each data-set in order to authorise its release to that application.

Analysts are permitted to configure and dispatch the analysis tools onto the cloud to perform experiments on the patient data. The countermeasures are chosen to protect against the possibility of a malicious analyst. The analyst is not permitted direct access to patient medical data because it is encrypted. Instead the analyst configures the input data for the experiment and dispatches the software onto the cloud. The company's key management service will only release the decryption keys for patient data to authorised software.

A trusted subgrid is used to restrict access to an authorised cloud provider. This check is performed by verifying the host certificate of the platform requesting access to the patient data. The certificate identifies the host as belonging to the approved provider. The certificate authority trusted to verify this claim is configured as part of the high-level policy of the key management service.

Recipient keys are used to ensure that analysts are not given direct access to experimental results. Instead access is mediated by a trusted database. The exact protection mechanism here is out of scope, but for example it might ensure that only aggregate statistics can be retrieved from the data. This guards against the possibility of an analyst revealing personally identifiable information.

To set up this protection the database server is allocated a private host key. The corresponding public host key is stored in the key management service as the recipient key. The recipient key is given to the job security manager executing on the cloud. Once the

analysis tool has terminated successfully, the results are encrypted with the recipient key[5] and staged out to the database server. Only the database knows the corresponding private key so the results cannot be obtained by an unauthorised user.

Integrity protection is used to ensure these countermeasures cannot easily be subverted by an attacker. If the analysis tools are tampered with prior to execution on the cloud, the hash value measured by the job security manager will be altered. Access to the decryption keys of the patient data will be refused because the application will no longer belong to the list of hashes of authorised software. The analysis tools are designed to prevent access by external attackers, so there should be no way to compromise data confidentiality without finding a vulnerability in the trusted components (see Section 8.2 for further discussion of the security of this architecture).

6.7 Conclusion

Until the grid middleware problem is solved there can be no secure foundation on which to build a trusted grid. Vulnerabilities in the underlying grid middleware layer undermine grid security, since they allow higher level services such as authentication and authorisation to be bypassed. Attackers will always focus on the weakest link in a system, so there is a fundamental need to overcome the middleware problem before a trusted grid can be developed.

The key management service is a key component introduced in this chapter. It prevents attacks on grid middleware from compromising grid user data or credentials. User data is protected using encryption and data integrity protection. The key management service contains a policy that will only authorise key release to a verified, attested job security manager. Attestation foils credential theft, since these credentials only allow data use within a trusted execution environment.

One of the most exciting properties of this approach is interoperability with existing systems. It does not require any alteration to the grid middleware itself, so it can immediately be used alongside the plethora of different middleware implementations that are in

[5]This is an over-simplification. In reality the results would be encrypted with a random key that is itself encrypted with the recipient key and staged out along with the data. See Section 6.5.2 for further details.

use today.

A second benefit is that the key management service permits the construction of digital rights management and mandatory access control policies. These policies are strongly enforced using trusted computing attestation. Digital rights management can be used to empower users aswell as data owners, and other stakeholders, to control how their data is used and disseminated on a global grid.

The following chapter examines in further detail how this architecture could be implemented in practice. It turns out that some changes are required to existing virtualisation software to support the job security manager. Since this goes against the original aim of this work, which was to achieve full interoperability with existing software, some alternative solutions are examined which weaken the security architecture somewhat in order to achieve that goal.

7 Ideas for Implementation

7.1 Introduction

Several difficult challenges must be overcome before the trusted grid can be implemented on a large scale. New trusted infrastructure services must be developed and integrated into current systems. In addition, these new services must be capable of being used without prohibitive management overheads or usability issues. The purpose of this chapter is to describe how these challenges can be addressed during implementation. This will enable an exploration of the feasibility of building a widely available trusted grid.

The approach suggested in this chapter places the emphasis on practicality and the need to ease the transition to trusted grid services. A key idea is to maintain backwards compatibility with existing data, and re-use existing solutions wherever possible. Where changes to existing services must be made, the goal is to offer a range of alternative implementations that trade off interoperability with legacy hardware and software against the level of security required by users.

A key management service is one of the new components needed to support a trusted grid. Organisations might find it difficult to add a key management service to their existing infrastructure for a number of reasons. Policy and key management is likely to be a complex and time-consuming task. Also, the vast majority of existing grid jobs and data repositories are not available in a suitable encrypted data format.

In Section 7.2 solutions are developed to

- implement automated policy updates to make operating the key management service largely transparent to end-users and service providers, and
- allow existing legacy data formats to be used within a trusted grid.

Most existing virtualisation software cannot fully support a trusted grid. Section 7.3 explores the changes necessary to create a trusted virtual machine monitor (VMM). A trusted VMM prevents administrators and owners of the platform from accessing data within virtual machines belonging to users. In addition the amount of trusted code is

minimised to reduce the likelihood of security vulnerabilities. These improvements are necessary to support fully encrypted and integrity-protected grid jobs running on remote, and potentially malicious, hosts.

The job security manager uses a novel architecture where the security layer is pushed down to the grid node in a virtual machine running alongside the grid job itself. Implementing this concept requires significant conceptual changes to the virtual machine architecture. In this new approach software can be deployed across many intercommunicating virtual machines that present themselves to the user as a single application. Section 7.4 develops a solution to this problem using a new concept termed a *coalition* of virtual machines, and explains how existing virtualisation software can be adapted to support this paradigm.

It might take a significant period of time for trusted hardware and software to be adopted by a large part of the grid community. In Section 7.5 a number of alternative solutions are examined that maintain support for existing platforms running legacy hardware and software. The different solutions demonstrate the benefits and drawbacks of implementing a trusted grid on platforms without trusted virtualisation software, and even without a trusted platform module chip.

7.2 Key Management Service

7.2.1 Key Management Service Architecture

The key management service acts as both the policy decision and enforcement point. Part of the intention behind this design is to keep the job security manager, that runs on remote grid infrastructure, as simple and robust as possible. Potentially complex access control policies are implemented by the key management service within the data owner's trusted network perimeter.

In a large distributed system like the grid, organisations are likely to adopt individual policy management solutions that are customised to their own requirements. It is important that the key management service can interoperate with many different policy enforcement and decision points. This allows organisations to continue to rely upon their existing, tested solutions with which they have already gained valuable experience.

Adopting widely used standards can facilitate the integration of the key management service with existing infrastructure. Trusted Network Connect (TNC) [167] is a standard developed by the Trusted Computing Group to add client attestation to network-based access control devices. Existing TNC solutions already solve many of the requirements for the key management service.

TNC is typically built into a gateway device that performs user and host authentication. TNC adds support for client attestation to verify that trusted software is in use before a client is granted access to the network. TNC is becoming popular in commercial network devices and is supported in the latest version of the Windows operating-system.

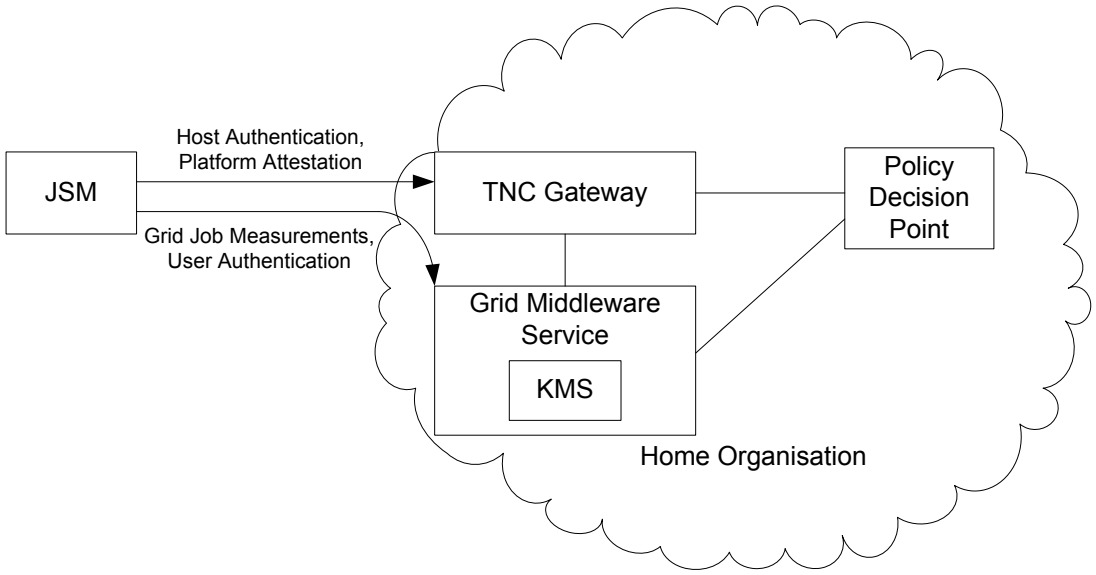

Figure 24: Key Management Service Architecture

Much of the access control functionality required for the key management service can be delegated to a network access gateway supporting TNC. An example architecture is shown in Figure 24. When the job security manager (JSM) contacts the key management service (KMS), access is mediated by the gateway. The gateway is responsible for authenticating the host platform and attesting that trusted software is in use. The gateway will refuse access to the network if attestation or authentication fails. The key management service runs as a service behind the gateway. Connections to the key management service are not

possible unless they are authorised by the gateway.

In this approach attestation is split into two distinct phases (see Figure 25). In the first phase, enforced by the gateway device, the remote grid platform is attested using TNC. The job security manager and all underlying components, such as the virtualisation layer, are attested at this stage (using the process described in Section 4.6.3). The gateway is configured to permit only a limited number of job security manager and virtual machine monitor solutions that are trusted by the organisation. In the second phase, the grid job itself is attested. This exchange is performed by the key management service. Mandatory access control and user-defined policies are enforced at this stage.

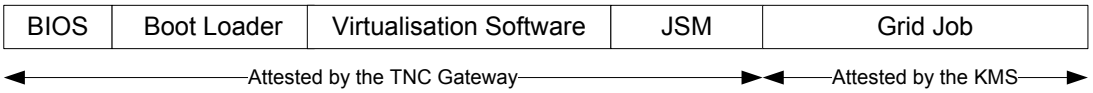

Figure 25: Example Platform Configuration Registers Used to Attest a Remote Grid Host

The TNC gateway provides a secure communications channel. For example, many gateways support a virtual private network for remote access that provides communications integrity and confidentiality using a protocol such as TLS or IPSec [115]. The job security manager running on the remote grid platform establishes a secure channel that is terminated at the TNC gateway.

The key management service and other components behind the gateway can communicate without support for the virtual private network. The integrity protection and end-point authentication provided by the gateway ensures the attested platform cannot be spoofed throughout a network session following attestation. Confidentiality protection is important to protect the decryption keys as they are transported to the job security manager over untrusted networks.

The key management service can be integrated with existing grid services. Grid middleware can then be relied upon to support user authentication based on delegated grid credentials. This significantly reduces the amount of new functionality that needs to be implemented. The use of grid middleware allows the key management service to support widely used standards for authentication in distributed systems such as Kerberos [113] or

X.509 proxy certificates [72].

The key management service must support a protocol to enable the exchange of grid job measurements and decryption keys. Implementing this protocol using web services languages such as SOAP [114] further improves interoperability with different job security manager implementations.

7.2.2 Policy Updates

The key management service has access to a policy database that defines the conditions for release of the decryption keys. This can include restrictions on the authenticated user and the grid job data measurements. The policy can be updated to reflect the users' requirements for protection of their data.

Key generation can be made the responsibility of the user that dispatches the grid job. The user stores the key securely in the key management service where it can be retrieved later. This improves interoperability with different cryptographic mechanisms because keys can then be stored and retrieved as uninterpreted binary objects. A database can be used to store the keys. Recipient keys can then be implemented by indexing separate wrapped keys for each data file according to each possible recipient.

Strict controls must be placed on changes to the key management service policy. These access controls can be enforced by the underlying database management system. This is one of the advantages of implementing policy and key storage using a database. In a discretionary access control policy users are permitted to manage their own policy updates to control the software that is authorised to run on the grid.

Mandatory access control policies can be implemented by preventing users from making changes to certain policy entries. Users will typically have no access to the policy controlling client host authentication and integrity measurements, as only administrators have access to the gateway access control device. This is appropriate because the permitted hosts and trusted software (such as the job security manager) should be decided on an organisational level.

Policy management is one of the key overheads in using the trusted grid. Each time

an encrypted job is submitted the user must make the required changes to the policy, and ensure the new decryption keys are made available. Requiring users to make manual updates to the access control policy is tedious and error prone. For example imagine a user is compiling a software application to analyse a data set. Each time the software is recompiled the user must update the policy configuration to enable access for the new version. This involves uploading a new set of data hashes, and possibly a new decryption key. This approach is unlikely to scale well for highly dynamic grid jobs in a large user environment.

Automated policy updates can make the underlying security layer completely transparent to users. Policy updates can be made by middleware that supports the process of job submission (refer to Figure 26). The middleware can contact the key management service on the user's behalf and update the policy to allow the grid job to execute. In addition, organisations might set global policies that cannot be over-ridden by users. For example, that all grid jobs must run with confidentiality protection. The advantage of this approach is that the security infrastructure becomes just another part of the grid middleware that non-technical users do not need to be directly concerned with.

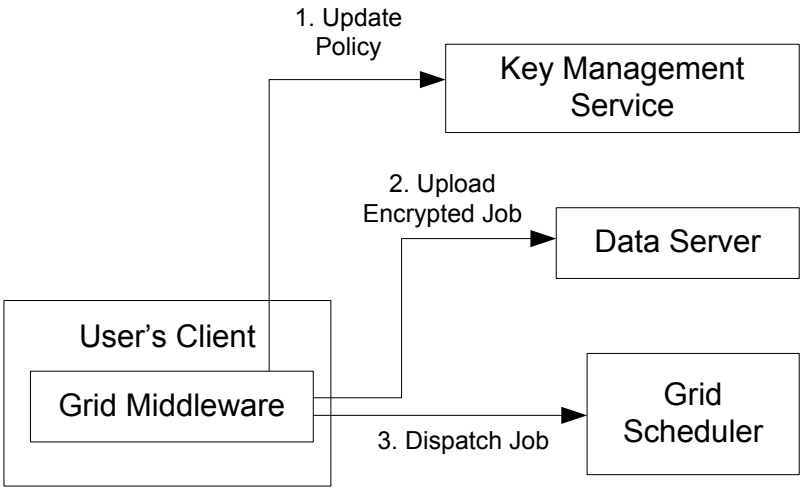

Figure 26: Automated Key Management Service Policy Updates

7.2.3 Data Management

One of the key difficulties in a trusted grid is making encrypted data available on a large scale. The cost of converting existing data repositories into a compatible encrypted format is likely to deter many data providers from participating in the trusted grid. A second problem is the resulting difficulties with key management. Key management activities such as rolling over old keys, handling key revocation and managing data provenance and destruction are likely to add further cost overheads and complexity.

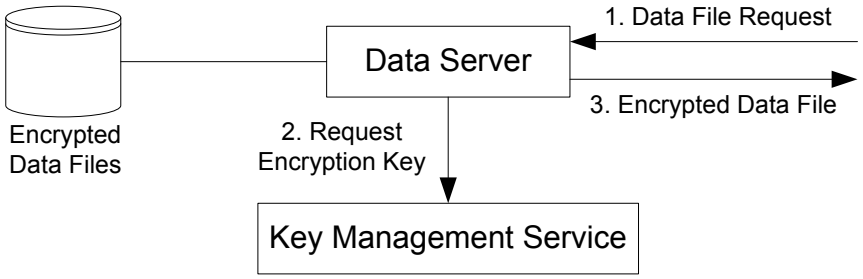

Figure 27: On Demand Grid Job Data Encryption

There are two different approaches to the encryption of grid data for digital rights management. To some extent these approaches trade off interoperability with the complexity of key management and data conversion. The first, from here on called *ahead-of-time encryption*, involves pre-encrypting the job data files and making them available on a data server (refer back to Figure 26). An alternative approach can be termed *on-demand encryption*. In this approach data is not encrypted until access to it is requested. The data is encrypted dynamically as it is transferred from the data server (see Figure 27).

There are advantages and disadvantages to these two approaches and the choice of which to use depends on the user's requirements. Ahead-of-time encryption is suitable where ease-of-implementation and interoperability is the over-riding concern. It is fully compatible with legacy grid middleware services. The data server is able to transfer the encrypted data like any other binary data. As a result it is fully compatible with data caching and replication services, data striping and other data transfer performance enhancements.

On-demand encryption requires special support from the data server. When a protected data file is requested, the data server performs several actions before allowing the file to

be downloaded. First an encryption key is generated and stored in the key management service. The file is then encrypted as it is read from disk.

A key feature of on-demand encryption is that no modifications are required to the grid middleware running on the external service provider. The data transfer is requested and retrieved using standard grid middleware protocols, and the key management and data encryption is handled independently by the data server. Therefore no change is needed to the existing grid data transfer protocols. However some performance optimisations like data striping — using simultaneous downloads — are incompatible with on-demand encryption because each replica will be encrypted using a different randomly generated key.

There are various strengths and weaknesses to the two approaches. Using on-demand encryption helps ensure that the data being accessed is the most up to date copy. With ahead-of-time encryption there is a danger that the encrypted copy of data will become out-of-date as changes are made to the original data. Also, data storage requirements are reduced because it is no longer necessary to hold encrypted copies of data. On the other hand, encrypting data as it is retrieved adds a performance overhead that might affect high-speed data transfers. Organisations are free to choose the most appropriate solution for their requirements, or to adopt both approaches for a different subset of their data.

7.3 Virtual Machine Monitor

7.3.1 The Need for a Trusted Virtual Machine Monitor

Making the trusted grid widely available depends on ubiquitous support for trusted virtualisation. There are signs that the grid is beginning to embrace virtual machine technology. Researchers predict that virtualisation will be widely available on many client and server platforms in the future [45, 83, 88, 147, 183]. This trend is reflected by the hardware support for virtualisation that is now available in mainstream computing platforms [162, 62], as well as virtualisation support in operating-systems such as Microsoft Windows Server and RedHat Enterprise Linux.

A trusted grid requires the availability of a trusted virtual machine monitor. The current generation of virtualisation technologies is designed primarily with the aim of server

consolidation [1]. Security is not an over-riding concern because the servers are kept within secure enterprise networks. A trusted grid, on the other hand, requires a VMM that can resist attacks from a potentially malicious host platform managed by an external service provider.

A second problem in achieving a trusted VMM is the diversity of current implementations. There are many large vendors pushing their own virtualisation technology [163, 12]. Standardisation is a particularly difficult issue. It could be said that vendors have little incentive to standardise VMMs because each vendor wishes to advocate exclusive use of their own solution. However there are signs that this is changing, as the Distributed Management Task Force industry consortium has recently released an open standard to enable interoperability of virtual machines [117].

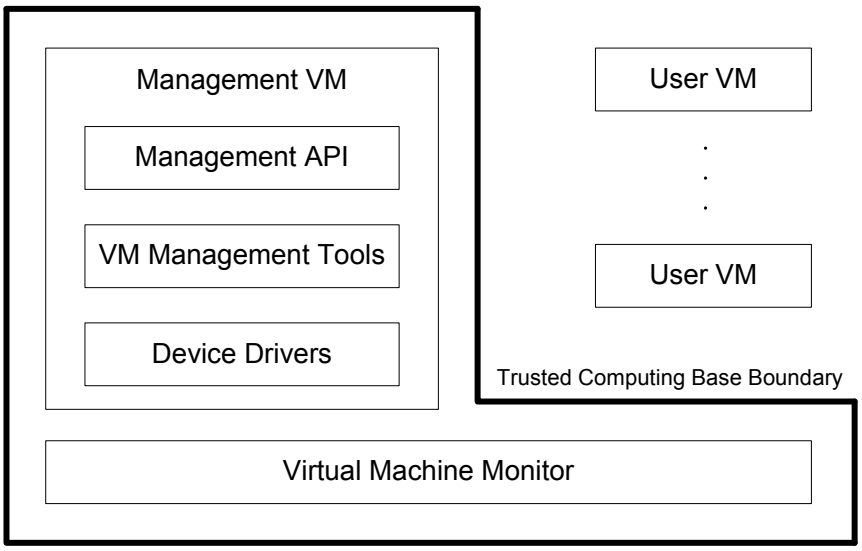

Figure 28: A Typical Virtualisation Architecture

Despite the variety of implementations, the high-level architecture of the current generation of VMMs is broadly similar. Many commercial VMMs including those by VMWare [163], Microsoft and Citrix (who own the commercial rights to the Xen hypervisor [12]) share the same high-level architecture shown in Figure 28. The management VM is used to configure and control all the virtual machines that run on the platform. The management VM is part of the trusted computing base because it has privileged access to the VMM and

hardware devices.

Mainstream VMMs do not currently provide protection against insider attacks. Administrators are generally permitted access to the management VM in order to use the VM management tools. However, once a user has privileged access to the management VM they can compromise virtual machine isolation, because they will have full access to the physical memory of the host platform. The management VM must have access to physical memory in order to host device drivers and to perform other privileged operations.

One of the goals of the trusted grid is to resist attacks from administrators, but current VMMs cannot meet this requirement. Administrators with access to the management VM can bypass the protection provided by the job security manager by reading sensitive data before it is encrypted, or by overwriting it with a Trojan horse.

A second major problem with current-generation VMMs is that the trusted computing base is too complex to be trustworthy. The management VM typically runs on a mainstream operating-system like Linux or Windows Server. The advantage of using these popular operating-systems is that they include wide device driver support. These operating-systems also have, however, a history of repeated security vulnerabilities [4, 78, 137].

The Xen hypervisor is one of the few popular VMMs to include support for a trusted computing platform module (TPM) [15]. The TPM is virtualised within the management VM using software that simulates the behaviour of a genuine hardware module. Here, again, the security of the management VM is critical because it holds sealed keys and attestation measurements. If the management VM is compromised this data can be stolen or manipulated by an attacker. For example an attacker could alter the attestation measurement of a virtual machine running a Trojan horse so that it appeared to be a genuine job security manager.

Within the commercial space changes are underway to improve the assurance of virtualisation. VMWare [163] offer a VMM that is implemented in firmware by supported hardware platforms. Implementing the VMM within hardware significantly enhances the protection of the trusted computing base, but this solution still permits full access to authorised users and administrators.

The current state of the art for trusted grid designs further undermine trust in the VMM. In the Trusted Grid Architecture [96] the grid middleware is placed within the management VM. This is convenient because it provides easy access to the VM management tools. However it means that attackers exploiting the middleware problem will gain complete control of the host platform, and from there they can compromise the job security manager and all the grid jobs running on the platform.

Building a trusted grid requires improvements to the current generation of virtualisation software. A trusted VMM is one that has the following properties:

- The trusted computing base resists attacks from authorised and unauthorised users and external attackers;
- The VMM runs virtual machines within an isolated environment with full memory protection;
- The VMM can attest to a remote user that it supports strongly isolated virtual machines using trusted computing.

No currently available VMM meets all the requirements for a trusted VMM. There have been several attempts to build a trusted VMM in the literature [54, 64, 122, 124, 138, 141]. These approaches, however, are not currently integrated into mainstream commercial virtualisation software. The remainder of this section concentrates on how to achieve a trusted VMM for use on the grid with minimal changes to existing solutions.

7.3.2 Implementing a Trusted Virtual Machine Monitor

It may take some time before commercial trusted VMMs become widely available. It is possible to adapt the current solutions to meet the requirements for a trusted grid. The purpose of the changes is to harden the trusted computing base by significantly reducing its complexity and removing the ability for administrators to compromise security-critical functions.

A suggested architecture is presented in Figure 29. The grid middleware is moved into an unprivileged virtual machine. All of the changes are confined to the management VM and

VM management tools. This considerably reduces the complexity of the implementation because no changes need to be made to the VMM itself, or the grid middleware.

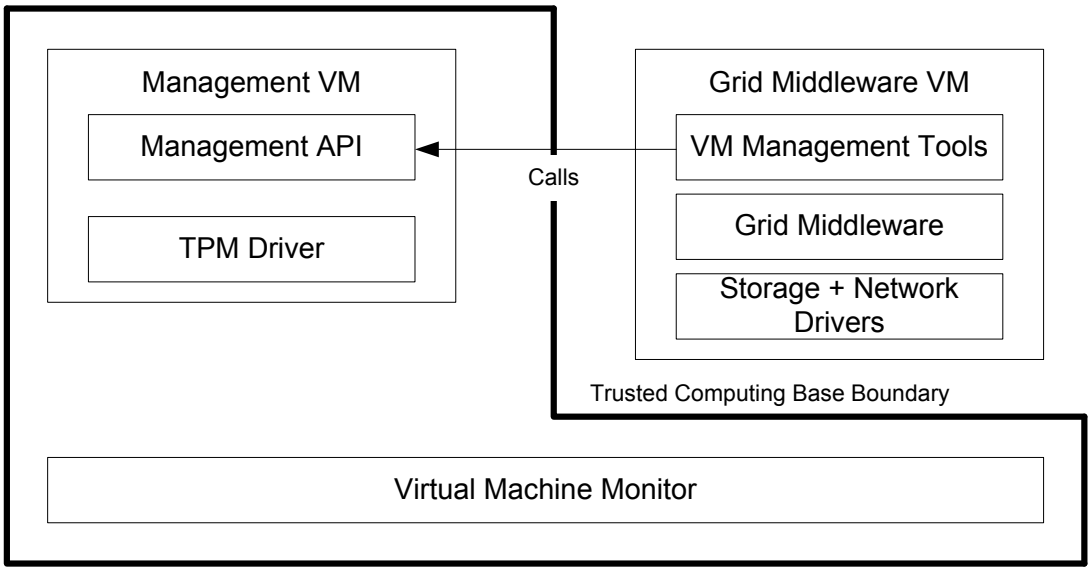

Figure 29: A Trusted Grid Virtualisation Architecture

Administrators must have access to the VM management tools, however, they do not need complete access to the host platform. In the new architecture the VM management tools are moved into an unprivileged VM that runs the grid middleware. Virtual machines are controlled via a restricted management interface exported by the management VM. The middleware VM is able to call the management interface using inter-VM communication such as a virtual network connection established by the VMM.

Administrators are no longer given privileged access to the physical memory of the platform, or to sensitive data such as the contents of virtualised TPMs. Instead they are only able to access the essential VM management functions needed to administer the platform such as configuring, starting and stopping virtual machines.

Many vendors are actively developing management interfaces for virtualisation software. The purpose of these management systems is to permit remote and automated management of virtual machines. This existing code can be used as the management interface for a trusted VMM. For example, VMWare have published a proposed standard management

interface that is shared by all their products [6]. This means that a significant part of the new functionality required for trusted a VMM is already being implemented by VMM vendors.

Buggy device drivers have been shown to cause 85 per cent of the system crashes in the Windows XP operating-system [165]. Many operating-system vulnerabilities have resulted from coding errors in kernel-mode drivers [39]. Once these vulnerabilities are exploited the attacker gains full, privileged access to the compromised system.

The trusted VMM architecture removes the majority of device drivers from the trusted computing base. Instead they are placed in the untrusted virtual machine running the grid middleware. The storage and networking drivers do not need to be trusted because the job security manager ensures all data is integrity and confidentiality protected before being handed over to the management VM.

Trusted computing functions are strongly protected against attack in the trusted VMM architecture. Unlike other device drivers, support for the virtual TPM is kept within the management VM. This prevents administrators and unauthorised users from attacking attestation measurements and encryption keys. An attacker who compromises the grid middleware will be unable to affect the integrity of the job security manager, so users' data will remain protected. This architecture overcomes the middleware problem because the virtual machine running the grid middleware is not privileged to access protected system resources.

Normally a virtual machine that hosts device drivers for physical devices will have uncontrolled access to memory. This is because many hardware devices implement *direct memory access* (DMA) to boost data transfer performance. DMA allows hardware devices to read and write to arbitrary locations in memory. Attacks utilising DMA must be prevented in a trusted grid architecture. Otherwise an attacker who compromises the middleware virtual machine could gain complete access to the host platform by communicating with the storage and network devices [122, 67].

New hardware virtualisation support being introduced on PC platforms provides protection against DMA attacks [179]. These improvements allow the management VM to restrict

access to DMA so that the middleware virtual machine can efficiently implement the storage and network drivers without bypassing memory-protection. Any access to unauthorised memory, such as that allocated to the VMM or other virtual machines, will be prohibited.

7.4 Job Security Manager

7.4.1 Virtual Machine Coalitions

Delivering the job security manager in a virtual machine is a powerful concept. Pushing the security layer onto the platform alongside the grid job means that the solution is interoperable with all compatible virtualisation software. Isolating the security layer from the grid job itself means that the solution works seamlessly with legacy software submitted by users. The digital rights management controls are isolated and protected within a dedicated virtual machine providing enhanced protection for encryption keys. Code complexity can be minimised because the job security manager is dedicated to a single purpose. This has a secondary benefit for management of attestation identities because simplified and specialised software is less likely to change often.

There is a problem, however, because currently available virtualisation software cannot run the job security manager. The reason for this is that they assume all virtualised device drivers are pre-installed. Every virtual machine is configured to use some subset of the available virtualised devices like storage, networking and so on. The job security manager, on the other hand, provides its own virtualised storage device that adds new security capabilities. The grid job is configured to use this security-enhanced storage device to provide data integrity and confidentiality. Current generation virtual machine monitors do not support user-definable virtual devices.

To understand the changes needed to support the job security manager, it is first necessary to understand in some detail how device virtualisation works. When a virtual machine attempts to execute a privileged operation to access a hardware device, the virtual machine monitor takes over. The access request and associated data is forwarded to the virtual machine hosting the virtual device driver. The data is copied into its memory space and the operation is performed, possibly on the real underlying hardware. The results are then

returned in the same way.

The virtual machine monitor can copy memory between virtual machines because it can access the entire physical memory space of the platform. To save copying the data, it is more efficient to share the memory pages between both virtual machines. A key part of device virtualisation involves setting up communications channels between virtual machines using shared memory.

Implementing the job security manager requires the creation of what might be termed a *coalition* of virtual machines. A coalition is a group of virtual machines that are tightly coupled together. They provide services to each other such as virtualised devices or other high-speed data channels.

On a technical level, the virtual machine monitor allows the virtual machines in a coalition to share memory with one another. This allows two virtual machines, that would normally be completely isolated from each other, to implement a communication channel such as inter-process communication. The virtual machines in a coalition can communicate with each other in the same way as multiple processes running on a shared operating-system.

Coalitions do not require significant new functionality to be implemented. All existing virtual machine monitors can already support virtual machine coalitions. This is because they use the same shared memory channels that are fundamental to existing device virtualisation.

In current virtualisation software the shared memory channels are fixed, as they are always established to the same device driver domain. The difference in coalitions is that the user can define their own shared memory channels. For example, the grid job and the job security manager virtual machine can be placed in the same coalition, allowing the grid job to have its storage device virtualised by the secure storage service.

7.4.2 Implementing Virtual Machine Coalitions

Support for coalitions can be added to virtualisation software by providing support for their definition and configuration. Instead of supporting a fixed set of virtualised devices the management layer must permit new virtual devices to be created and hosted within a

user-provided virtual machine.

Figure 30 is an example of a suitable configuration for the coalition containing the job security manager and a protected grid job. The job security manager configuration specifies that it hosts a virtual device called `secure_storage` that will be used to implement secure storage. The grid job virtual machine has its storage configured to use the `secure_storage` virtual device. This means that a shared memory channel will be established between the two virtual machines. The job security manager will receive data write requests from the grid job, encrypt the data and then forward the request to the underlying grid job storage image file (which is connected to `HARD DRIVE C`). The same process happens in reverse for data read requests.

```
BEGIN COALITION
        VM LABEL "Job Security Manager"
                HARD DRIVE A = "file:///jsm.image"
                HARD DRIVE B = "file:///jsm_config.image"
                HARD DRIVE C = "file:///grid_job.image"
                VIRTUAL BLOCK DEVICE "secure_storage"
                VTPM = true
        VM LABEL "Grid Job"
                HARD DRIVE A = "vbd://secure_storage"
END COALITION
```

Figure 30: Example Configuration for a Virtual Machine Coalition

Normally virtual machines are configured and created independently. In a coalition, a group of virtual machines are defined and started together. In the configuration file in Figure 30, the grid job will be booted up at the same time as the job security manager. There is a problem here, because the grid job cannot read any data from disk until the job security manager has successfully launched and obtained the decryption keys. This process is handled by the virtual machine monitor in the same way as for standard device driver virtualisation. The grid job virtual machine will block once the read request is made. The response will only be received once the job security manager reads the data on its behalf. The virtual machine monitor will wake up the grid job virtual machine after copying the

response into its shared memory area so it can continue to boot from the encrypted image file.

Another point of difficulty is how to use attestation with a virtual machine that is part of a coalition. Normally attestation involves attesting a single platform, or virtual machine. However attestation must capture information about a coalition or group of virtual machines. The reason for this is that becoming part of a coalition involves opening up shared memory pages with another virtual machine. The attestation measurement must capture the existence of shared memory, otherwise a malicious virtual machine could modify the behaviour of an attested virtual machine by manipulating its memory (refer to Section 4.5 for a full description of this attack).

Attestation of a coalition of virtual machines involves measuring the configuration of the entire coalition. This can be achieved by having the management VM include the coalition configuration in its attestation measurement. This allows the attester to verify that all shared memory channels are set up correctly in the authorised manner. Any manipulation of shared memory will be detected because the attestation measurement will change.

Consider for example how the job security manager (configured as in Figure 30) would be attested. The user dispatches the grid job along with the coalition configuration file. Assume the virtualisation architecture is implemented as it is described in the previous section (see Figure 29). The grid middleware will create the coalition by calling the VM management API. When the job security manager starts, its virtual TPM measurement is extended to include a measurement of the configuration of the coalition. If the configuration has been altered, for example by an attacker that has compromised the grid middleware, this will be detected because the attestation measurement will change.

There is a convincing argument for the use of coalitions in a wide range of applications, beyond providing a trusted grid. Coalitions allow any existing application to be broken down into separate virtual machines to enhance security and isolation. High-profile organisations have already shown interest in this idea. It was the basis of Microsoft's NGSCB architecture [122], where for example a web browser might run in an untrusted virtual machine, and receive support from another trusted virtual machine for security-critical op-

erations like completing online purchases. The idea has also been explored in the research community to enhance the security of existing applications [34, 42, 144].

Coalitions must become widely supported before they can be used in a large-scale grid. This is only likely to happen once coalitions are adopted and implemented by the virtualisation vendors themselves. This section has shown that coalitions can be implemented without adding significant new functionality to virtualisation platforms.

7.5 Alternative Implementations

The lack of availability of trusted virtualisation platforms is currently holding back the deployment of a trusted grid. Changes to the virtualisation architecture are necessary to prevent administrators from over-riding the security mechanisms, and to support the concept of distributing the security layer in the job security manager. Although these changes might be introduced by vendors of virtualisation products in the future, it is likely to be some time before they widely are available.

Even though coalitions are not currently supported by virtualisation vendors, there is nothing to stop these improvements being made by the user community. An open-source virtual machine monitor such as Xen [12] could certainly be adapted to support coalitions. This modified software could then be adopted throughout a virtual organisation. If changes can be made to the virtualisation software, however, there are much easier ways to implement a job security manager. After all, one of the key ideas behind its design is interoperability with any virtualisation platform. Once that constraint is removed the design can be considerably simplified.

7.5.1 Adapting the Virtualisation Layer

If the virtualisation software can be modified an alternative solution is to pre-install the job security manager in the host platform. The job security manager is installed and managed by the service provider as part of the underlying virtualisation layer.

Some changes to existing virtualisation software are still necessary to prevent administrators from accessing the data encryption keys. A suitable approach involves installing the

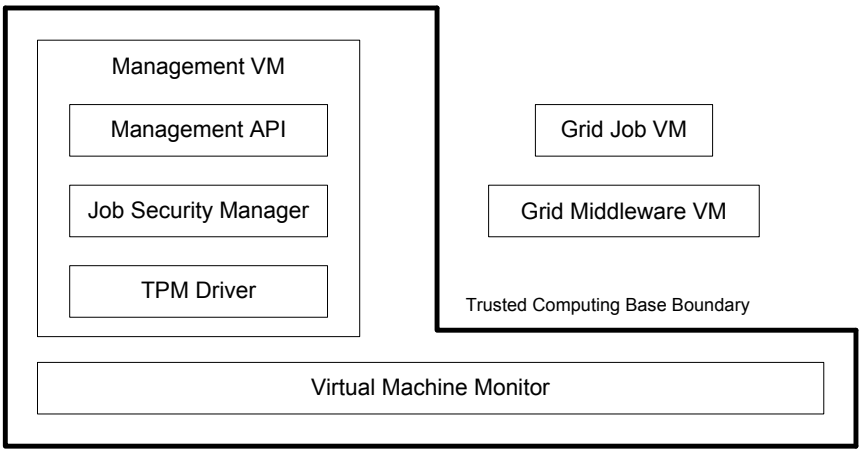

Figure 31: The Job Security Manager Installed in the Virtualisation Layer

job security manager inside the management VM (see Figure 31), using an adapted form of the architecture presented in Section 7.3. Administrators can only access the management VM through a restricted API interface, from which they are unable to steal users' encryption keys.

Support for virtual machine coalitions is not needed if the virtualisation software can be modified. The secure storage driver can be provided like any other device driver that is pre-installed on the host platform. With these improvements a virtual machine can be configured to use either normal, unprotected storage, or the secure storage device provided by the job security manager. The Xen hypervisor, for example, already supports secure storage drivers on behalf of virtual machines [177]. For virtualisation platforms that currently implement secure storage the only extra functionality that needs to be added is support for obtaining decryption keys from the key management service.

There are advantages to pre-installing the job security manager. Firstly, a single job security manager can be shared between many grid jobs. When coalitions are used, each job has its own dedicated job security manager which results in a memory overhead, particularly if thousands of grid jobs are run on a single platform. Secondly, there is no need to support a virtual TPM device. The job security manager can access the physical TPM directly, because there is no longer a need to share it between multiple virtual machines. Avoiding the use of virtual TPMs overcomes key management difficulties in tying a software-based

TPM to a physical platform [15].

One of the advantages of using virtual machine coalitions is interoperability. Pre-installing the job security manager lacks flexibility because users can no longer provide their own customised implementation. Service providers may choose to adopt different job security manager implementations. Since the user can no longer push down their own implementation they will have to trust each available job security manager. They also need to keep track of changes to the attestation measurements.

Pre-installing the job security manager does not scale well in a global-scale grid. In large virtual organisations it will be difficult to mandate that every member uses customised virtualisation software with the necessary security enhancements, especially if these systems are not supported by the software vendors. Widespread support for coalitions is therefore essential if the trusted grid is to become universally adopted.

The security architecture is weakened quite significantly when the job security manager is placed in the management VM. In this approach a vulnerability in the job security manager gives the attacker access to the entire physical memory space. A successful attack therefore leads to a compromise of every grid job running on the platform. The attack also harms the service provider because sensitive management functions such as billing and bandwidth monitoring can be compromised.

When coalitions are used the job security manager runs as an untrusted virtual machine so a successful attack would only compromise the single grid job it is attached to. This means that poor security will only harm the party that is responsible for it. Other users and organisations running a secure job security manager on the same platform, and the service provider themselves, will not be directly affected.

7.5.2 Adapting the Grid Job

There is another alternative implementation for the job security manager that is fully compatible with all virtualisation platforms without any changes. This step forward in interoperability can be achieved by implementing the job security manager within the grid job virtual machine (see Figure 32).

The job security manager can be installed in the operating-system running the grid job. Disk volume encryption support is already available for popular operating-systems [85]. The job attestation service can be built upon an IPSec client or a locally-installed SSL network proxy. A significant advantage of this approach is that the job security manager will work with any virtualisation platform available today. No virtual machine coalition or secure storage support is required from the virtual machine monitor itself.

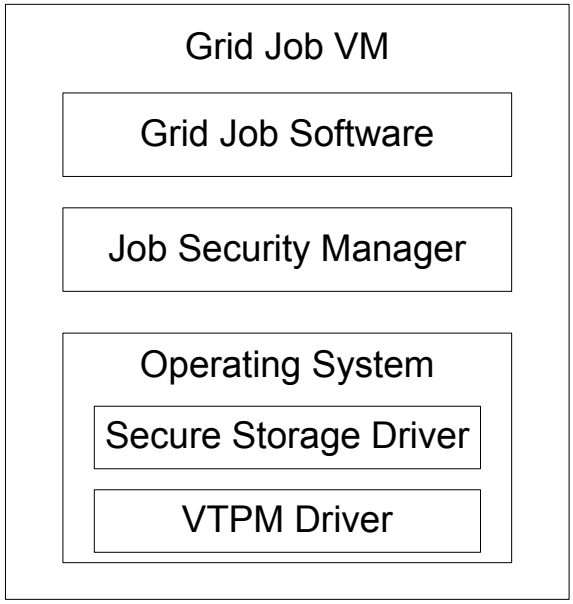

Figure 32: The Job Security Manager Installed in the Grid Job Virtual Machine

The grid job virtual machine can no longer be entirely encrypted if the job security manager is installed within it. The job security manager must obtain the decryption keys before the grid job can be decrypted. This can only happen once the operating-system has booted and the job security manager software has started running. It follows that although the grid job software and data can be encrypted, the operating-system cannot. In most cases this is not a problem because the operating-system itself is unlikely to contain proprietary software. Complications do arise however, such as ensuring that the swap file is encrypted so an attacker cannot steal sensitive data that has been paged out to disk.

Although the operating-system cannot be encrypted, it must be integrity protected. The operating-system running the grid job can be adapted to perform a trusted computing

authenticated boot process using a virtual TPM. The job security manager is measured in turn by the operating-system before it is executed. The key management service attests these measurements before releasing the decryption keys. Operating-system support for attestation is vital to prevent attacks on the integrity of the job security manager, leading to compromise of the data encryption keys.

Once the decryption keys have been obtained the grid job is ready for execution. The decryption key is passed to a secure storage driver within the operating-system. The driver is responsible for encrypting and integrity-checking all disk read and write requests to the grid job data partition. The job security manager then decrypts and measures the grid job to verify its integrity prior to execution. The job attestation service can use this measurement at a later time to allow the user to verify the integrity of the grid job when collecting the results.

Legacy software is no longer supported once the job security manager is installed inside the grid job. The user is forced to use an operating-system that is compatible with the job security manager. When coalitions are used the grid job runs in a separate virtual machine that can use any operating-system. A second interoperability problem is that the operating-system must be modified to support attestation. Linux currently supports authenticated boot, but other popular operating-systems like Microsoft Windows do not, and legacy operating-systems are unlikely to ever support attestation. When coalitions are used this problem does not arise because the grid job can be attested externally by the job security manager.

Like other alternatives, installing the job security manager in the grid job significantly weakens the security architecture. The problem is caused by the resulting lack of isolation between the grid job and the job security manager. An attacker who compromises the grid job will also compromise the job security manager. The attacker can now request all the other decryption keys the user has access to by sending fake grid job data measurements. The job security manager will be successfully attested to the key management service, which will then give out the decryption keys even though the attacker has gained complete control over it. This attack is prevented if coalitions are used because there is strong

isolation between the grid job and the job security manager.

7.5.3 Trusted Computing Hardware Requirements

Trusted computing hardware is evolving rapidly. There are now a range of hardware technologies that may or may not be available on a trusted platform. A trusted grid must be able to cope with a heterogeneous mix of available hardware and software resources. Better security will be available on platforms running the latest trusted computing technology. Older platforms should not be precluded from participating in the trusted grid altogether. Achieving the vision of a global grid requires support for a wide range of devices offering different levels of assurance.

The trusted platform module is at the heart of a trusted platform. Early versions of the TPM adhered to the 1.1b specification [166]. This was later succeeded by version 1.2 [170] adding support for a number of new features. Further revisions are likely in the future. Many features in the specification are optional. There are a number of different TPM chip manufacturers and each have chosen to implement a different subset of the specification [135]. This situation means that the trusted grid must interoperate with a variety of different TPM chips.

The core trusted grid functionality uses fundamental trusted computing functions. The main function used by the trusted grid is attestation, and this has been present since the very earliest generation of TPM chips. An older TPM chip implemented to the 1.1b specification is sufficient to support a trusted grid. The TPM 1.2 specification introduces new features that can be useful in some specific scenarios. Monotonic counters can be used to prevent sealed data being rolled back to a previous version by an attacker [100]. This can enhance the security of public-resource computing style scenarios, where there is a need for persistent access to data across platform reboots.

Even if a platform does not contain a TPM chip it is still possible to take advantage of some of the security benefits of a trusted grid. Attestation is not needed if the user fully trusts the owner of the platform. Provided the job security manager is integrated into the virtualisation layer (as in Figure 31 in Section 7.5.1) it is sufficient to prove that the

platform is owned and managed by a certified service provider. The service provider is then trusted to ensure that the job security manager and the rest of the trusted computing base is correctly installed and configured. In this case no TPM is required and the user retains the benefits of secure storage and the isolation of untrusted grid middleware.

When a platform does not contain a TPM it is still necessary to verify the trustworthiness of the remote platform. This can be done using host certificates instead of attestation. The grid execution host is authenticated by the job security manager when the connection with the key management service is established. The grid job measurements are then exchanged over the secure channel. This approach relies on a certification procedure for service providers that includes verification that the systems are kept secure and well-managed. It is particularly suited to cloud computing and smaller data grids where there are fewer participants. As the grid becomes larger it becomes harder to maintain close evaluation and certification of all the members in this way.

Another optional trusted computing component is hardware support for direct memory access (DMA) protection. DMA protection allows device drivers to be placed outside the trusted computing base within the middleware virtual machine (as in Figure 29 in Section 7.3.2). DMA protection, for example using Intel's Directed I/O technology [1], has only recently become available on PC hardware. It is possible to implement DMA protection in the virtualisation software without direct hardware support [179]. If neither a hardware nor a software solution is available, all device drivers must be placed in the trusted management VM. This increases the complexity of the trusted computing base, but retains the other advantages such as the restricted management interface.

Traditional trusted computing attestation requires all the firmware chips involved in the boot process to be measured [170]. This includes the BIOS and option ROMs contained in devices such as storage controllers. In a cloud computing environment containing mostly homogeneous equipment this might be practical, but as the grid increases in size the number of potentially valid measurements soon becomes difficult to manage.

Intel Trusted Execution Technology simplifies attestation because it introduces a new trusted launch functionality that does not rely on firmware for secure execution [62]. This

technology is only available on the latest hardware as it requires the TPM version 1.2 chips and a supported motherboard and CPU. It will be some time before trusted execution technology is widely available but until that time very large scale grids will be more difficult to manage due to the variety of valid attestation measurements for heterogeneous hardware equipment.

It is clear that a global grid will need to accommodate a large variety of trusted computing hardware and software. Depending on their requirements users may be willing to trade off security against performance and interoperability. An effective way to deal with this problem is to extend the existing grid information services framework with data about the availability of trusted computing support. Grid job schedulers can then take this into account along with other factors when selecting a suitable service provider and execution node. This approach permits the user great flexibility when matching existing grid resources to their own individual security requirements.

7.6 Case-study: Grid computing

This final case-study considers how a trusted grid can be achieved in a virtual organisation with varied support for trusted computing and virtualisation. This brings together several of the concepts in this and previous chapters to show that a trusted grid could be achieved by gradually moving over to more secure technologies, without the need for wholesale replacement of existing hardware and software.

Consider the example virtual organisation (VO) depicted in Figure 33, consisting of a medical company in a research partnership with two universities. The company intends to release sensitive medical data to the universities to help them conduct research experiments. In order to fulfill its duties for the protection of medical data, the company wishes to ensure that only the university researchers directly involved in the experiments can access the data. Other users of the grid hosted by the universities must be denied access.

In existing grid technology these requirements would be solved using access controls. User authentication and authorisation systems can be used to control access to the patient data. These existing technologies have two fundamental security weaknesses that

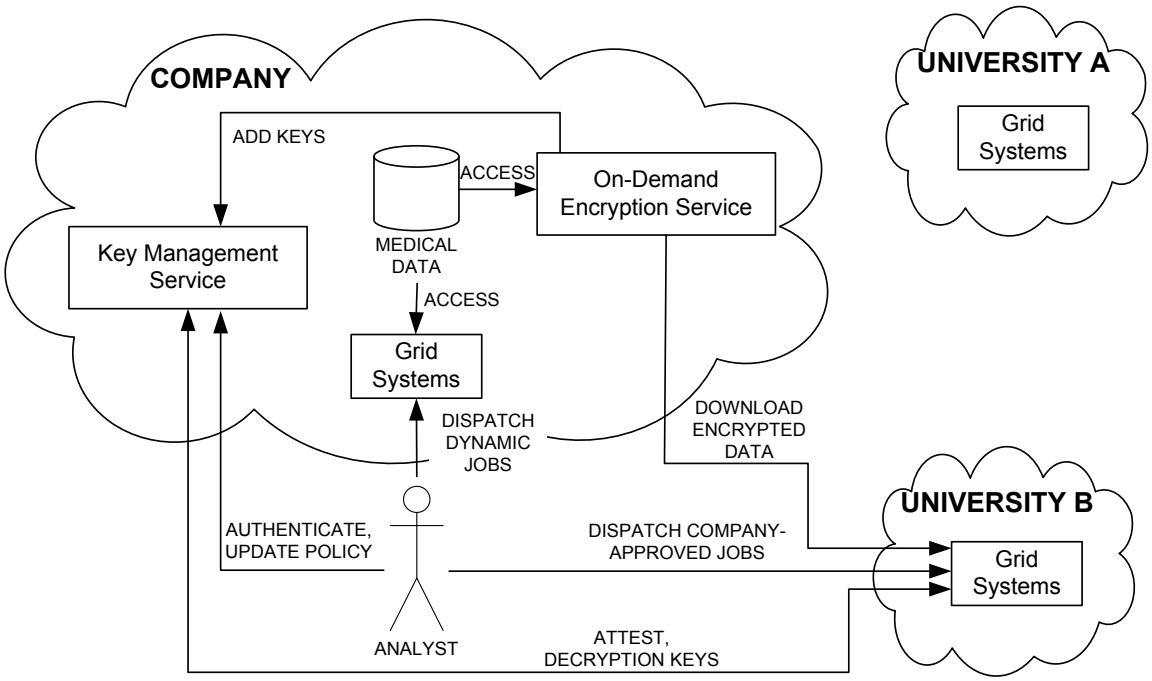

Figure 33: Grid Computing Case-study

the trusted grid attempts to solve. Firstly the company must trust that the grid systems managed by the universities are able to secure user credentials from theft by an attacker who could then masquerade as a legitimate user. Secondly the company must trust the universities to safely handle the data after it is released, and to securely wipe it once it is no longer needed.

The company has decided that on a large university campus, containing a wide variety of grid and non-grid systems, the likelihood of vulnerabilities resulting in data loss is too high. Instead a trusted grid will be used to provide strong data confidentiality protection while it is processed on university systems. To achieve this the company builds a key management service to authorise release of the data to pre-approved analysis software. The key management service policy lists the hashes of software that is trusted to access the data, and the company verifies new software before adding it to the list.

Whereas a cloud provider has to support many users with different security requirements, a smaller VO can adopt a more specialised, unique solution. The universities partic-

ipating in the VO have each installed a customised virtualisation layer that provides a job security manager in the virtualisation layer (as described in Section 7.5.1). This is feasible because the universities have a close working relationship and are willing to agree upon a common software and security solution. A significant advantage of this approach is that it only involves making minor changes to existing open-source virtualisation software, rather than developing full support for VM coalitions, which is a much more complex change (see Section 7.4).

One downside of this approach is that researchers do not have full control over the analysis software that performs the experiments. New software must be vetted by the company and added to the key management service before it can be used. New experiments can be run using the existing authorised software by changing the input parameters. However if a novel experiment requires the development of new software, there will be a delay while this is authorised by the company. For experiments that require constant re-development of the analysis software this approach would introduce unnecessary overheads.

The company decides to offer the use of its own internal systems to the universities to permit the execution of dynamic software. The company is willing to allow this because it has perimeter security and other countermeasures within its internal network that carefully control allowed data import and export. These countermeasures prevent potentially malicious grid jobs from exporting sensitive patient data to an attacker on an external system.

This use-case can be implemented by adding an additional set of policy rules to the company's internal key management service. These rules permit release of data decryption keys to any grid host that is internal to the company. University researchers would be authorised to update these policy rules before dispatching new software onto the grid. As discussed in Section 7.2.2, automated software could perform these updates on their behalf, making use of the trusted grid a transparent user experience.

The company wishes to avoid unnecessary data encryption. The encryption operations are expensive in terms of time and data processing overheads. When grid jobs are executed within the company's infrastructure, there is no need for the medical data to be encrypted.

Instead, the unencrypted data can be transferred directly to the target grid host, since the company fully trusts its own infrastructure to provide suitable protection for the data. On the other hand, medical data must be encrypted before it is transferred to external university grid systems. A suitable solution here is to use on-demand encryption (see Section 7.2.3).

This case-study is intended to illustrate how a trusted grid can be achieved without significant changes to existing infrastructure. Note that a key management service is only needed if there is a requirement for data confidentiality. For example if the universities wish to execute grid jobs on their infrastructure with only integrity protection, then a key management service is not needed. Instead the TPM is used to provide a strong guarantee over software integrity once the results are retrieved. This shows that a great deal of benefit can still be gained from a trusted grid without implementing the potentially complex data access policies needed to support data confidentiality.

7.7 Conclusion

The trusted grid architecture presented in this thesis places strong requirements on the underlying virtualisation layer. In many ways virtualisation software is still maturing, and a trusted hypervisor is still only available in research prototypes. However new hardware and software developments mean that a fully trusted solution is likely to be commercially available in the very near future.

The job security manager is a new concept for virtual machine deployment. Virtual machines must work together co-operatively in a coalition. Since this is a new research idea borne out of the work for this thesis, coalition support is not currently available in existing virtualisation software. However the changes it requires have been shown to be straightforward, existing mainly at the higher level VM configuration layer, rather than requiring changes to the underlying hypervisor.

To some extent requiring changes to the virtualisation layer goes against the original stated aim of achieving full interoperability with existing software. This chapter has therefore gone on to show that a number of alternative solutions are possible, installing the job

security manager either alongside the hypervisor, or in the virtual machine running the grid job. Both these approaches weaken security slightly, with the huge benefit of maintaining interoperability with existing software.

The following chapter evaluates the proposed solution on the basis of both performance and security. The aim is to demonstrate that the proposed architecture is not just feasible to implement, but could also meet the strong performance requirements placed on grid software. It is difficult to find a concrete way to measure security, so instead the approach taken is to explain how defense in depth is achieved even if individual trusted components become compromised.

8 Evaluation

8.1 Introduction

The purpose of this chapter is to evaluate the feasibility of the trusted grid architecture. Two separate aspects are examined. Firstly, whether the architecture offers sufficient protection for users' data when it is processed on the grid. Secondly, performance is considered, and whether the overheads introduced by the security architecture could seriously affect the usability of the grid.

No system can completely eliminate security vulnerabilities. Trusted grid architectures should aim to minimise the amount of trusted software such as grid middleware. Security problems remain, however. The correct operation of hardware and software is relied upon for the security enforcing functions.

It is important to understand the vulnerabilities that the trusted components are exposed to, their likelihood, and the impact of compromise. A vulnerability analysis (Section 8.2) helps highlight the security-critical parts of the architecture, as well as identifying additional countermeasures that can be put in place to mitigate the risk. It also gives an insight into design decisions that were made to enhance the robustness of key components.

Operations like encryption and hashing necessarily incur performance penalties. This is in contradiction to the goals of the grid, which aims for very high speed computation and data processing. The performance analysis (Section 8.3) identifies key areas that are likely to be crucial for achieving acceptable performance. The main areas of concern are the overhead of virtualisation and protected storage. The section goes on to discuss a number of design decisions and emerging technologies that will help ensure a high quality of service for users on the trusted grid.

8.2 Security Analysis

8.2.1 Attacks on the Trusted Hardware and Software

Trusted computing mechanisms are susceptible to hardware attacks if the attacker has physical access to the platform. Prevention of hardware attacks is not one of the design

goals for the trusted platform module (TPM) [62]. Researchers have demonstrated how a number of hardware attacks are possible as a result.

To reduce costs the TPM chip is soldered directly onto the motherboard with the pins exposed. This leads to an attack where the TPM attestation measurements can be set to zero by placing a piece of wire to the reset pin, allowing the attacker to spoof the attestation measurements of a running platform [82]. The attestation measurements sent to the TPM over the low-pin count bus can be modified with relatively inexpensive equipment to hide the actual state of the platform [89]. A custom hardware modification attached to the system memory modules can cause them to provide false data during attestation [74].

Hardware attacks can be performed without physical access to the platform. For example vulnerabilities in the CPU can bypass memory protection, allowing an attacker to break the isolation between virtual machines. According to the Intel Specification Updates, the Intel Core 2 CPU has 128 confirmed bugs, many of which remain unfixed [75].

Modifications to the BIOS can subvert trusted platforms, and many machines permit unauthenticated firmware updates [82]. Trusted computing attestation relies on the integrity of the BIOS because it performs the first measurement.

Although the TPM is designed to resist software attacks, there is also the possibility that it could fail given malicious input. Recent research has demonstrated some API level vulnerabilities by formally modelling the TPM specification [132], and further work is ongoing in this area [69].

The implications of compromising trusted hardware varies depending on the application. An important factor is the level of physical security that can be put in place. Dedicated data centres used for cloud computing are likely to have physical countermeasures in place that make such attacks unlikely. Hardware attacks are more likely in a data grid where shared IT infrastructure is used and unauthorised individuals might have access to grid systems. In public-resource computing scenarios no physical security is likely to be possible. The owner of the platform has complete access to the machine, but may not have the skills required to modify hardware. A similar scenario in games consoles has lead to an industry developing around the sale of hardware chips that bypass built-in security mechanisms [174].

New developments in trusted computing are likely to make hardware attacks more difficult. The late launch feature of Intel Trusted Execution Technology removes the BIOS and other firmware from the trusted computing base [82, 62]. Intel have plans to integrate the TPM into the motherboard chipset [62]. This will make physical attacks considerably harder because data will pass over the high-speed bus, which requires expensive hardware probing equipment to access [62].

Attacking trusted software is in many ways easier than attacking hardware. Unlike hardware, the behaviour of software can be altered by modifying program code in memory. The TPM is normally used to measure software as it is loaded from disk [142, 91]. An attacker can avoid detection after compromising the system by ensuring that they do not modify the disk storage for attested applications. The result is that attestation changes the method of attack somewhat, but it does not thwart software attacks.

A trusted grid relies on many software components for its secure operation. The trusted computing base includes the job security manager, the key management service, the virtualisation layer and the grid job itself. Memory isolation and other security mechanisms are used to limit the impact of a successful attack.

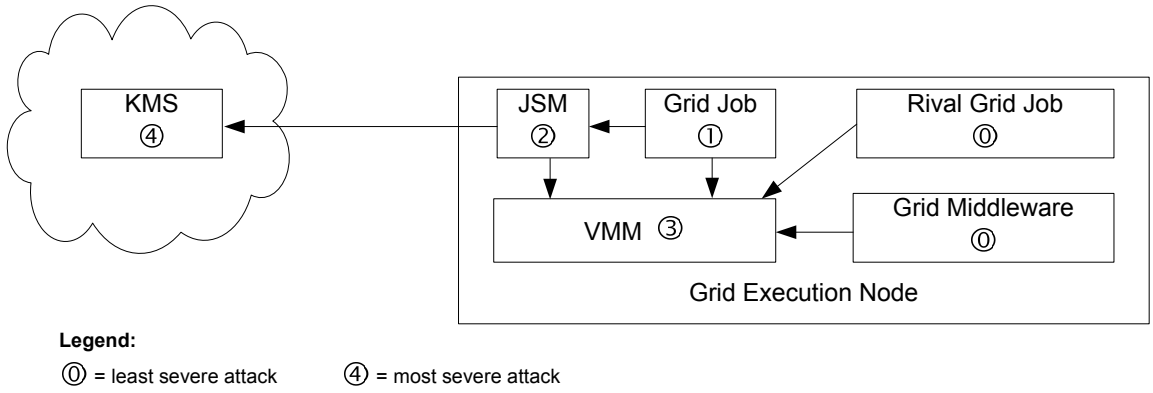

Figure 34: The Trusted Computing Base of the Trusted Grid

Figure 34 shows an overview of the trusted software components. The numbers indicate the impact of compromise, where a higher value is more severe. The arrows indicate trust dependencies between components. For example compromising the job security manager implies that the grid job is also compromised because it has full access to its storage data.

The remainder of this section examines each of these areas, including the attacks against them and the countermeasures that can be put in place to add additional protection.

8.2.2 Grid Job

It is the user's responsibility to ensure that their grid job is secure. The trusted grid does not attempt to prevent bugs and security issues with grid applications. Instead the aim is to protect against new vulnerabilities introduced by running software on untrusted remote infrastructure. If a grid job is compromised an attacker may gain access to the sensitive data being processed. The trusted grid countermeasures such as secure storage help protect the grid job from attacks via the underlying infrastructure and middleware software, not the grid job itself.

Isolation is used to limit the impact of a successful attack on a grid job. Virtual machine isolation means that an attacker cannot directly compromise other grid jobs running on the same platform. The job security manager is also isolated within its own virtual machine. This separation prevents an attacker who has compromised the grid job from gaining access to the decryption keys belonging to that user.

Users can in some cases significantly reduce the likelihood of their grid job being compromised. For example, a grid job that renders a single animation frame could be run without any network access. Once the computation is complete the encrypted results can be returned using the grid middleware data transfer service (using the approach described in Section 6.4.2). Since no remote access to the grid job is possible the vulnerability surface is considerably reduced. This approach is not practical for all applications. In these cases network security components such as a virtual private network can be installed in the job security manager to limit the hosts and users that can connect to it.

8.2.3 Job Security Manager

Successfully compromising the job security manager gives an attacker access to far more sensitive data than compromising a single grid job. An attack is possible that permits the attacker unauthorised access to all the data available to the grid user. The attack proceeds

as follows. The attacker connects to the key management service and the job security manager attests itself. The attestation will succeed because only program code running in memory has been changed. The attacker then sends spoofed measurements of encrypted data and receives the decryption keys from the key management service. By repeating this attack the attacker can gain unauthorised access to all data accessible to the compromised user account.

It is vital that the job security manager is robust against attacks. The job security manager has been designed to eliminate unnecessary functionality and reduce complexity, hopefully resulting in fewer security vulnerabilities. Many tasks, such as key generation, policy management and enforcement, are offloaded to the key management service. This considerably simplifies the implementation of the job security manager.

The secure storage service encrypts blocks of data and is unable to interpret their contents. The filesystem drivers are instead situated within the grid job virtual machine. This approach improves security because buggy filesystem drivers will not compromise the job security manager. This attack path could otherwise be exploited to allow an attacker who has gained control of the grid job virtual machine to attack the job security manager and escalate their privileges on the system.

A single organisation might distribute multiple implementations of the job security manager, each containing the minimum necessary functionality for different tasks. For example if the grid job does not need secure network communications during operation the job attestation service (described in Section 4.6) can be omitted. Organisations might set a security policy requiring the most sensitive data to be accessed using a job security manager that runs a firewall, virtual private network and other software that restricts network access. The number of potential security vulnerabilities is reduced by stripping out unnecessary functionality.

A possible source of remote attacks on the job security manager comes from a malicious key management service. There are two ways to conduct this attack. The attacker can either force the job security manager to connect to a host under their control, or they can compromise a legitimate key management service. The attacker can then attempt to exploit

a vulnerability in the protocol used to obtain the data decryption keys. If the attacker is successful they can redirect the job security manager to another key management service and steal all the decryption keys the victim is authorised to access.

Authentication can be used to defend against redirection attacks but may not be a practical solution in a large-scale grid. The job security manager can be configured to trust a set of public host keys belonging to authentic key management services. To ensure the set of public keys have not been tampered with by an attacker, each key management service needs to verify the list of keys during attestation of the job security manager. In a large-scale grid this is not practical because the owners of the key management services may not all trust each other, or they may lack the shared public-key infrastructure needed to establish mutual trust.

The design of the trusted grid rests on the assumption that the job security manager can be implemented securely. Protocol attacks are only possible if bugs or other vulnerabilities are present. The only effective way to guard against a malicious key management service is through secure and robust implementation. The job security manager is designed so that the code implementing the protocol can be small and easily verifiable. The only interaction necessary is sending measurements and receiving decryption keys. No further communication is entered into once the decryption keys have been obtained. This is a significant step forward over existing grid systems where many complex protocols, such as those used to support execution and data management, must be implemented correctly to avoid security vulnerabilities.

8.2.4 Virtual Machine Monitor

Compromising the virtual machine monitor has a greater impact than a weakness in a single job security manager. The virtual machine monitor is responsible for enforcing isolation between all the virtual machines. If it is compromised the attacker can take control of all job security managers running on the platform. This is possible because the attacker can read and write to any location in physical memory. Every user that runs their grid job on the compromised platform risks losing all the decryption keys they are authorised to access

if the virtual machine monitor is compromised.

High assurance virtual machine monitors are not currently widely available. VMWare, one of the largest virtualisation platforms vendors, published 19 security advisories in 2008 [176], many of which address multiple vulnerabilities. The lack of secure design in current virtualisation platforms has been noted by many other researchers [80, 86]. The trusted grid places even more stringent requirements on the security of the hypervisor, because it must prevent administrators accidentally or deliberately subverting the security policy.

Research and development is currently very active in the area of secure virtual machine monitors and isolation kernels. The Green Hill real-time operating-system has successfully completed evaluation for Common Criteria EAL6+ certification, which requires semi-formal verification of the system design [31]. SEL4 is an attempt to engineer a formally verified embedded version of the L4 microkernel [40]. VMWare [163] has recently released a product called VMWare ESXi. This is a minimal virtual machine monitor that runs within firmware built into the PC platform. These developments are likely to pave the way for a highly trusted grid in the future by significantly decreasing the likelihood of an attacker compromising the virtual machine monitor.

Covert channels are a security problem that has traditionally beset multi-level secure systems [92], but they can also affect virtualised platforms. A covert channel is an unintended data channel that can be used to break the isolation between virtual machines. For example a covert timing channel might use CPU utilisation as a signal to transmits bits of data between two virtual machines that are running on the same processor. High-speed covert storage channels are possible if, for example, the virtual machine monitor fails to zero out memory that is then re-allocated to another virtual machine.

The impact of a covert channel on a trusted grid is not immediately apparent, because it requires two virtual machines to be compromised. For an attacker to exploit a covert channel they would have to take control of the victim virtual machine in the first instance. This would imply that they already have access to sensitive data stored in the grid job, or the job security manager, so they have no need to exploit a covert channel.

Covert channels do harm the ability to protect grid jobs from attack in a trusted grid

architecture. For example assume that a firewall and intrusion detection system are running in the job security manager. If the grid job is compromised a covert channel can be used to transfer the data directly to another virtual machine, bypassing the network-based countermeasures. Covert channels allow an attacker to circumvent the information flow policies enforced by the virtual machine monitor and this can lead to data compromise.

8.2.5 Key Management Service

Compromising the key management service is one of the highest severity attacks that can occur in a trusted grid. The key management service protects the sensitive data belonging to an entire organisation. If it is compromised the attacker can steal the decryption keys for all the users managed by that system. Mandatory access controls rely on the integrity of the key management service, so they can also be bypassed.

The architecture of the trusted grid makes the key management service one of the easiest components to secure. The key management service runs from within the owner's organisation. This makes it easier to secure than the other components that run on untrusted remote infrastructure. The home organisation can put physical and network security countermeasures in place. An organisation has a greater level of control over the security of the key management service, so it makes sense to concentrate the security enforcing functions within it. The resulting high impact of attack is balanced by the fact that the owner has a large degree of control over the countermeasures that protect it.

One potential source of attack on the key management service comes from external attackers. The key management service cannot be completely isolated from the network because it must accept connections from remote job security managers. There is a danger that a vulnerability could be exploited by an attacker by manipulating this network protocol. The likelihood of this attack is significantly reduced by immediately attesting the job security manager before any further protocol exchanges. Once attestation has taken place attacks become much less likely because a valid job security manager will not send malformed protocol packets and will only call the external interfaces in the expected manner.

The key management service is also open to attack from authorised users within the

data owner's organisation. Authorised users are permitted to manage the policy rules in order to authorise their grid jobs to run on remote systems. Deliberate or accidental policy misconfiguration are two possible vulnerabilities. This could allow a user to access decryption keys belonging to another user, or bypass mandatory access control policies. These vulnerabilities can be mitigated using access controls to restrict changes to policy rules, and accounting and audit measures in order to monitor mis-use of the system.

Unauthorised users might seek to attack the key management service if they have access to the internal network of the host organisation. An example of this type of attack might be a buffer-overflow type vulnerability in the management interface used to update user policy. Once the attacker has taken control of the key management service they could gain access to all the decryption keys and policy managed by the system. The impact of these attacks can be reduced by using a TPM in the platform running the key management service to protect the decryption keys in sealed storage. This approach can follow a similar design to that used to enhance the security of grid credentials in the online MyProxy service, where keys are sealed to the user's authentication password [104, 97].

8.3 Performance

8.3.1 Performance Issues

High performance is one of the key drivers behind the development of the grid. Security sits uneasily with aspirations that the grid will take high speed computing to new levels [47]. Encryption, authentication and other cryptographic operations that are needed to create a trusted grid necessarily incur a performance penalty. Users with strong security requirements may be willing to make some sacrifices in terms of reduced speed of computation provided performance does not suffer to the extent that the grid becomes unusable.

In terms of the overall trusted grid architecture there are two areas that are likely to be crucial for performance. The first is the overhead of virtualisation. Studies suggest that virtualisation is a scalable approach for high performance distributed systems, with a typical overhead of 20 percent over native execution [133]. The job security manager, however, uses an unusual virtualisation architecture, and this is likely to introduce additional

overheads over existing approaches. A second key area for concern is the overhead of the data encryption and integrity-checking performed by the secure storage service, because this could increase the time it takes for grid jobs to access data on secondary storage.

Other aspects of the trusted grid architecture are less likely to affect the run-time performance of a grid job. The overhead of communication with the key management service is one potential area for concern. The job security manager must set up a secure communications channel using encryption and authentication. Only a negligible amount of data is transferred, including the data encryption keys and data hashes, so the overall CPU overhead is likely to be low. Unlike data staging, this data transfer must occur once the grid job is scheduled for execution, rather than in advance. A typical grid job will involve the transfer of gigabytes of data so the time taken for this interaction is likely to be very small in comparison.

Attestation normally involves expensive public key operations to digitally sign the measurement. The trusted grid architecture minimises the use of attestation. Data is only digitally signed once execution has completed and the grid job has shut down. This approach has significant performance advantages because only one digital signature operation must be performed during the lifetime of the grid job. The downside of this approach is that data cannot be staged out with tamper detection while the grid job is running. Live data staging would require a new digital signature operation every time data is modified resulting in an unacceptable performance overhead.

Another issue to consider is the memory overhead of the job security manager. Since each job is run with its own instance of a job security manager this could soon add up to a significant amount of memory. This problem is reduced because the job security manager can run a highly restricted operating-system with minimal memory requirements. For example the *damnsmalllinux* operating-system requires only 8Mb of RAM [152]. Memory is also needed to support sufficient buffering for disk read and write operations so that may increase this figure. The memory requirements are likely to be acceptable provided a user does not wish to run hundreds of short-lived grid jobs on a single platform. An alternative solution is to share a single job security manager between multiple grid jobs, as described

in Section 7.5.

There is a storage constraint involved with integrity protection. The list of disk block hash values needs to be stored alongside the other metadata. Pletka et al. [125] estimate the space requirements as one percent of the size of the data (when using the SHA-256 hash algorithm). To reduce the storage overhead block hashes can be generated dynamically by the job security manager at the expense of decreased start-up time. Alternatively the list of hashes can be transferred over the network as part of the job data files at the expense of increased data transfer size.

Although there was insufficient time within the scope of this work, a more detailed understanding of performance issues could be obtained by building a fully functional prototype as described in Section 9.2.1.

8.3.2 Virtual Machine Monitor Performance

Virtualisation is a major source of performance overhead in a trusted grid. A study of the Xen virtual machine monitor has shown that disk driver virtualisation adds around 4 per cent to the total CPU time involved in storage access [26]. The trusted grid architecture involves the use of two levels of disk virtualisation. The grid job has its disk virtualised by the job security manager. After the data is encrypted it is written to the virtual storage device provided by the underlying virtualisation software. Since this requires data to be copied between two sets of virtual machines, the performance impact would be expected to be at least double that of existing virtualisation architectures.

A single activity contributes to the majority of the performance penalty of virtualisation. When a virtual device is accessed requests and responses need to be transferred to the virtual machine hosting the device drivers. Over 75 percent of the time taken to access a virtual device is spent making these data transfers between virtual machines [163]. Reducing the number of virtual machine transitions during device access is therefore a major focus when optimising virtual machine monitors.

Modified device drivers can significantly reduce the overhead of device virtualisation. These drivers are installed in the operating-system running in a virtual machine. The mod-

ified drivers are specifically designed to optimise performance in a virtualised environment. For example, physical device access requests can be buffered and sent in groups to eliminate unnecessary virtual machine transitions. These changes have been shown by VMWare to more than double throughput rates on the same hardware [163]. These improvements can be adopted in a trusted grid by distributing grid jobs that are pre-installed with modified device drivers, but this does limit support for legacy operating-systems.

A technique called para-virtualisation can eliminate almost all the overhead involved in virtual device access [12]. Para-virtualisation requires extensive modifications to the virtual machine. Device drivers are stripped out altogether. The operating-system kernel is modified so that device access results in an explicit access request to the virtual machine monitor. Expensive data copying operations can be avoided by sharing virtual memory pages between the two virtual machines. Data is transferred directly to the driver domain using this shared memory channel. This approach can be used in a trusted grid by adapting the job security manager to use para-virtualisation. Grid job virtual machines can continue to use the normal, slower form of device access to improve interoperability with existing software.

8.3.3 Secure Storage Performance

One of the powerful ideas behind distributing the security layer alongside the grid job is that security functionality can be tailored to the user's requirements. Kher et al. provide a comprehensive review of the many different secure storage solutions that are available [85]. Approaches differ significantly in terms of performance, security and functionality. Many of these alternatives are suitable for use within the job security manager, so users are free to select the option best suited to their individual requirements.

It is difficult to accurately measure the performance overhead of disk encryption. The results vary significantly depending on the exact workload characteristics under test [180]. Despite these difficulties many studies agree that full-disk encryption is practical for wide-scale use [44, 125, 180]. This is evidenced by the fact that full-disk encryption is now built into popular operating-systems such as Windows and Linux [44, 85]. Studies suggest that

the overhead of encryption over unprotected file operations varies from 5 percent [44] to 20 per cent in the worst case [125] depending on the implementation.

The Terra virtualisation platform enforces data integrity by checking hashes of data blocks as they are read from disk [54]. The correct hashes are stored along with other file metadata. These hashes are in turn verified by hashing the entire virtual disk image prior to start-up and comparing the resulting value to a certified, digitally signed measurement. In terms of performance this means that start-up time is increased while the hash value is verified, but the creators of Terra find the overhead to be acceptable and do not seek further optimisation [54]. Other researchers suggest the use of a Merkle hash-tree, which can reduce the overhead to the logarithm of the data size for sparsely accessed data [100, 125].

In many cases the performance overhead of encryption can be avoided altogether. Encryption is not required if the user only needs data integrity. The results can be attested using a digital signature to verify that the grid job was not tampered with during execution. This digital signature is produced once the grid job has shut down, so there is no impact on run-time execution.

If data confidentiality is required, performance can be improved by disabling integrity protection. Instead, encryption may be sufficient to guard against the most likely attacks on data integrity. Since filesystem metadata is usually encrypted along with the file contents it may be difficult for an attacker to decide which data blocks to modify to tamper with a specific file. Even if an attacker successfully malforms software they still need a way to access the leaked data. If the grid job has no access to the network during execution there will be no way for the attacker to access the leaked data even if the grid job is successfully compromised through an integrity attack, because access to physical devices is controlled by the virtual machine monitor.

Compatibility between data encrypted in different formats is a problem in a trusted grid. Existing approaches to secure storage assume that data is encrypted and decrypted on the same device. In a trusted grid there is a need to pre-encrypt data and then make it available for decryption on many different execution nodes. If the job security manager assumes that data will be encrypted in one format, but the data provider makes it available in another

format, then there is a compatibility problem and the job cannot be executed. Incompatibility might be caused by the use of different secure storage implementations, for example.

Full interoperability between different encrypted data formats can be achieved using two levels of encryption. First the entire data file is symmetrically encrypted by the owner. The job security manager retrieves the data decryption key from the key management service and begins to decrypt the file. During decryption each data block is written to disk in an encrypted format supported by the job security manager. This solution is clearly bad for performance because the entire virtual machine image must be decrypted and re-encrypted prior to execution, but it supports interoperability between many different implementations.

Many grids are likely to be able to avoid these interoperability problems. In both cloud and public resource computing computing use-cases the owner of the data also provides the job security manager. The data can be provided in a suitable format so there is no need for conversion. Data grid use-cases can overcome these problems provided all parties agree on the same job security manager implementation. As the data grids become larger it is increasingly likely that data providers and users might choose different encrypted data formats. The only solution in these cases is to accept that performance will suffer due to the need to transform the data into another format prior to execution.

Another alternative to speed up encryption and integrity protection is the use of hardware acceleration. Hardware-based cryptographic accelerators can offer considerable performance improvements for symmetric encryption, public key operations and data hashing. Due to the cost-constraints the TPM will generally not rival these solutions for speed and feature set, indeed, the TCG standard does not require any support for symmetric encryption [170]. For example an implementation of the AES 128-bit symmetric encryption algorithm in hardware can achieve a throughput of 241MBit/s [101]. However it is unclear whether these hardware solutions could be adapted to work efficiently in a virtualised environment, where the cryptographic hardware is shared by many job security manager virtual machines.

New hard drives are becoming available that provide hardware encryption suitable for virtualised environments [169]. These drives encrypt and decrypt data at native disk speeds.

The protection mechanism works by having a virtual machine set up a public key with the disk drive over a secure channel. The corresponding private key can then be used to authenticate and gain access to the encrypted storage. The drive is capable of storing multiple public keys to support access to many data compartments.

The job security manager could use disks with built-in encryption to implement secure storage. The job security manager would authenticate itself to access the compartment using a private key held in sealed storage. Security is also improved because the code implementing the secure storage can be almost entirely removed from the job security manager, meaning there is less scope for bugs. These disks do not presently support integrity protection, but that is likely to change in the future. Another downside of this approach is that availability is currently limited, especially for enterprise storage solutions such as network-attached storage and storage area networks.

8.4 Conclusion

It is difficult to demonstrate the extent to which security improvements have been achieved. This chapter has attempted to do that by examining the impact of a successful attack on each component in the trusted grid. This has demonstrated tangible benefits over existing approaches. In particular, the large quantity of grid middleware software used on existing grids is no longer trusted.

In addition, compromise of a job security manager only affects the user that deployed it, and no other user of the system. This means that users can be empowered to choose implementations of the job security manager that they feel offer appropriate levels of security robustness. In a global grid there might be a variety of implementations available to suit users with different assurance and performance requirements.

Performance is clearly a tradeoff with security. Since grids focus on very high speed computation, adding expensive operations such as encryption is always going to be difficult. This analysis has shown that the overall time overhead is likely to be in the region of five to twenty percent in the worst case. New developments taking place in hardware are likely to further improve performance by reducing the overhead of virtualisation.

9 Conclusion and Further Work

9.1 Contributions

Five key security and interoperability goals for a trusted grid were identified in Chapter 1. In this section the main contributions of the thesis are presented in the context of these goals and the extent to which they have been achieved.

9.1.1 Solving the Middleware Problem

The middleware software that drives the grid is relied upon to secure users' data and credentials. Even in the relatively short history of the grid a number of security vulnerabilities have been discovered (see Chapter 6). The software systems involved perform complex functions that are inherently difficult to implement to a high level of assurance. A lack of isolation in the design means that vulnerabilities can easily lead to successful attacks on users' data.

It is this combination of factors — the likelihood of vulnerabilities together with the high impact of attacks that result from them — that causes the grid middleware problem. Unless the middleware problem is addressed it is likely to become a crucial stumbling block in the future development of the grid. Businesses and other organisations handling sensitive data may not be willing to trust grid service providers that are unable to guarantee strong protection of their data.

One of the major contributions of the thesis is a solution to the middleware problem. The new trusted grid architecture significantly improves the protection of users' data. This is achieved through a large reduction in the amount and complexity of privileged software. The solution works by introducing a new set of trusted grid services. The job security manager isolates user data from grid middleware using encryption and integrity protection. Theft of user credentials no longer compromises users' data because the key management service will only release decryption keys to a trusted platform running authorised software.

Data deployed on the grid is subject to a potentially high level of threat from other users. A grid user might be sharing the same grid computer with a rival organisation. The

proposed solution has been specifically designed to resist attacks from other users of the grid. The job security manager runs within a protected virtual machine where it cannot be accessed even if other parts of the host are compromised. It can be implemented with simple and robust code because the key management service is relied upon to perform the complex policy enforcement functions, while the key management service itself is protected within the internal network of the grid job owner.

9.1.2 Resisting Admin Attack and Error

In large virtual organisations it is difficult to be certain that all the administrators in charge of grid systems are trustworthy. Administrators can potentially disable trusted grid services by tampering with the code and data. Perhaps more importantly, users must trust that administrators have not accidentally compromised the security of the system, for example by forgetting to install a crucial security patch.

Trusted computing attestation allows a user to verify that a remote grid system is secure. Existing solutions to attestation do not work well in a grid. Every time a service provider upgrades or reconfigures grid software they must inform their users about the change. This approach becomes infeasible in a large grid containing thousands of organisations using a diverse range of customised software solutions. Users would have no easy way to decide whether a system is trustworthy unless all grid service settled on a small number of accepted implementations, which goes against the very problem the grid aims to solve.

The trusted grid solution presented in the thesis solves these problems with attestation. The grid middleware itself does not have to be measured, so service providers are free to use their own customised solution. Only the job security manager and the virtualisation software are measured. Since both of these components are designed to be small and simple they are unlikely to change often. Furthermore, the job security manager is provided by the user themselves, so they only need to confirm it has not been modified in transit. By measuring just these two components the user can gain assurance that a grid system is trustworthy.

Trusted computing hardware does not currently resist physical attacks. This limits

the protection that is possible against administrators and other attackers with physical access to the system. In cloud and data grid environments this may be acceptable because physical, procedural and other security measures may be in place. Lack of protection is most apparent in public resource computing grid environments, where the platform might be a home user's computer. In the future trusted computing is likely to provide higher levels of protection against physical attacks that might make it useful even in highly adversarial environments such as these [62].

9.1.3 Enforcing Digital Rights Management

Securing information exchange will become increasingly important as virtual organisations grow in size. As grids get larger it becomes more difficult to ensure that every member is equally trustworthy. Members of a virtual organisation might give a third-party access to data without the owner's authorisation. This may be done maliciously, but also for more benign reasons (as part of an outsourcing contract, for example). Most current models for delegation do not allow data owners to strongly enforce controls over data dissemination [18]. Hosts on the grid offer proof of their identity, but not their good behaviour.

Like many new information processing systems the grid is subject to threats from insiders. A company should be concerned about their users leaking company secrets to competitors over the grid. The huge computational power of the grid increases the scope for its mis-use. Companies may be legally obliged to show that they can control and monitor its use. Simple data access controls are not enough because users are given either complete access to data or none at all.

Digital rights management is ideally suited for controlling data dissemination in large distributed systems. Encryption binds access control policies to the protected object, meaning that controls are enforced wherever the data is accessed on the grid. Users can be restricted to accessing data only through authorised applications. This possibility opens up new opportunities for richer and more flexible access control policies. Service providers and users on the grid might only be trusted to use data in certain ways, for example to respect patient anonymity in a medical trial.

The thesis has shown that the job security manager and key management service combine to form a robust digital rights management platform for the grid (see Section 6.5). The proposed architecture is able to enforce the following access control policies:

- A subgrid places a strong boundary around a virtual organisation and ensures that protected data cannot be passed outside the subgrid even if one or more of the members is malicious.
- Users can be prevented from sending sensitive data on the grid to unauthorised recipients.
- Stakeholders can control the software that is available to users on the grid. This is a powerful feature because it permits complex security policies to be enforced by the end-user applications.

A key downside of digital rights management is the overhead and cost involved in managing the access control policies. Involving end-users in policy definition is likely to be error prone and expensive, undermining the security advantages digital rights management offers.

The architecture proposed in the thesis attempts to solve these problems by minimising the need for user interaction. Client-based grid middleware can be modified to automatically update the key management service with a suitable access control policy as part of the job submission process. Provided the policies can be automatically defined in the majority of cases, this approach has the potential to make policy management largely invisible to end-users.

9.1.4 Interoperability with Existing Grid Middleware

It has taken many years for grid software to reach its current state of development. At this stage making major changes to support a trusted grid is likely to be met with significant resistance from the grid community. If trusted grid functionality becomes available in a small number of projects, there is a danger that the trusted and legacy grid communities will become isolated from each other. And in the long-term, even if the trusted grid

becomes ubiquitous, the resulting plethora of different implementations are unlikely to be equally trusted by every member of the grid, leading to continuing interoperability problems between virtual organisations.

The trusted grid architecture presented in the thesis can interoperate with all existing grid solutions without modification. This is possible because the trusted grid services operate independently from the grid middleware in a virtual machine submitted by the user. This logical separation of the security layer from the underlying grid resource is a major step forward for interoperability. It allows service providers to offer a common grid infrastructure to the entire grid community, while still offering individual virtual organisations the freedom to adopt their own trusted security solution.

Availability of trusted computing technology remains the major hurdle in delivering a global-scale trusted grid. Trusted computing encompasses a wide range of hardware and software, so any solution must cope with a highly varied environment. The proposed solution to this problem is to provide a number of alternative trusted grid implementations (see Section 7.5). The different solutions trade off security and interoperability against compatibility with existing grid resources. This approach fits in well with the concept of the grid, where user requirements are matched to heterogeneous hardware and software resources. Security then becomes another user requirement alongside other factors such as bandwidth and storage.

9.1.5 Support for Legacy Grid Jobs

A major selling point of virtualisation is compatibility with existing legacy software. Being able to place software directly onto the grid without changes is a significant advantage for users. Existing trusted grid solutions do not support legacy code because they require modifications to support attestation [96, 54]. Making these changes to grid jobs requires access to the source code, and can be a costly and time-consuming process involving security experts to ensure that the correct part of the software is being measured.

The trusted grid solution developed in the thesis works seamlessly with all legacy grid jobs. Attestation is supported in the job security manager instead of the grid job itself.

The job security manager measures the grid job and reports those measurements to the key management service without the involvement of the grid job software. This solution allows users to take full advantage of the trusted grid without adapting their existing applications. A vision for the trusted grid is that where appropriate the new security measures can be transparent to end-users to minimise the need for training and the likelihood of accidental mis-configuration that might undermine security.

Origin attestation allows legacy grid jobs to be attested in dynamic environments. This is particularly useful in public resource computing where the grid job will run over many platform reboots. Previous approaches do not work well in this situation because a platform reboot would cause the attestation measurement to change unpredictably as new data is written to disk. Origin attestation overcomes this issue by replaying the attestation measurement taken when the software was first executed. This solution is particularly suited to short-term grid jobs where the changes in the time after deployment are unlikely to be significant.

9.2 Further work

9.2.1 Trusted Grid Prototype

The security benefits of the trusted grid architecture do not come without drawbacks. An effective way to evaluate whether these problems are acceptable is to perform a user-based trial and acceptability study. This should be based on a production grid project so the system can be evaluated in the context of a genuine use-case scenario. The study will help uncover system management, usability and performance issues, as well as demonstrate whether user requirements are fully met by the proposed solution.

The trial should investigate the following areas of potential concern:

- The difficulty of managing the access control policies for digital rights management, and also investigating the policies that would be most appropriate for real-world use-cases;
- The performance overhead of running grid jobs protected by a job security manager;

- The claim that the job security manager can be implemented with a small amount of secure code.

Before a prototype can be implemented, however, it will be necessary to produce a detailed high-level design of the system. To some extent determining what needs to be specified in further detail would be part of the exercise reserved for future work, however the following areas are likely to be included:

- A full protocol specification showing the detailed interactions between components involved in the job submission process (including the job security manager and key management service — see Section 6.4.2).

- A specification of the new interfaces and components needed at the virtualisation layer to support virtual machine coalitions and the job security manager (see Section 7.4).

- A design for how the grid middleware will interact with the job security manager in order to provide access to a user's proxy credentials for the purposes of authentication with the key management service (see Section 6.4.3).

It will be necessary to implement a prototype system to perform the evaluation. The testing should be done in stages in order to simplify development.

In the first stage a reduced functionality prototype can be used to test the digital rights management system. The prototype can be built quickly by integrating the job security manager into the virtualisation layer (as described in Section 7.5.1). By making the assumption that the administrators are trusted the architecture can be further simplified because it is not necessary to perform attestation, or even use a TPM. The key management service can be constructed by adapting an existing policy decision point. Several existing distributed access control solutions might be a suitable starting point, such as VOMS [3] and PERMIS [21].

The second-stage prototype would consist of a more complete implementation of the job security manager. The virtualisation platform must be adapted to add support for the

attestation of virtual machine coalitions (as described in Section 7.4). Accurate benchmarking tests can then be performed to measure the CPU and disk access performance overhead when the job security manager is run in a virtual machine.

This prototype will also help justify the claim that the job security manager can be implemented securely. Further work in this area might be possible to improve assurance, such as formally verifying the design and ultimately the implementation [171].

The third stage prototype should concentrate on compatibility with production middleware projects and integration with existing grid toolkits. The fact that the solution easily interoperates with different grid middleware platforms makes this task considerably easier. The key management service gateway can be decoupled from the policy decision point so that it can be used with several different policy management engines (using the approach described in Section 7.2).

The trusted grid architecture has been built with the ability for customisation in mind. The final prototype would be an open-source reference implementation that can be adopted and refined by the grid community to meet their specific requirements. Virtual organisations can then begin to leverage the advantages of being able to use multiple job security manager and key management service implementations on the same underlying grid infrastructure.

9.2.2 Trusted Audit Services

One of the goals of solving the middleware problem is to create a strong trusted foundation for the grid. Digital rights management has been explored in detail in this thesis. The intention is that other security components can be built on top of the trusted grid architecture. These new systems can gain the assurance benefits of being based on a secure and attested grid platform. A trusted audit service is an example of such a service.

Detection of insider attacks in the grid is complicated by the need to perform auditing across a distributed system. One of the benefits of the key management service is that it acts as a centralised point for security enforcement. The information available to the key management service provides a useful data set for security auditing. Each time a user executes a grid job the key management service can record the hashes of the application that

was executed, the host on which it was run and the user account that requested it. Much of this information is strongly protected against spoofing using trusted computing attestation. The key management service could therefore fulfil an important role in a distributed audit service.

9.2.3 Trusted Grid Services

Grid services have emerged as a standard way to deploy grid-based applications [48]. A grid service works by hosting the grid application as a web service so it can be accessed through standard XML and SOAP-based protocols [114]. The grid service can be implemented in a straightforward way by deploying web services middleware software in the virtual machine hosting the grid application.

Grid services to some extent undo the benefits of solving the middleware problem. The web services middleware is large and complex, much like the grid middleware layer. The problem stems from the fact that the web services layer runs within the trusted virtual machine that hosts the grid application. If the web services middleware is compromised the attacker can bypass the protection of the job security manager and directly attack the integrity and confidentiality of the users' data.

A similar approach to solving the grid middleware problem might be used to build trusted grid services. The web services middleware must first be placed in an untrusted virtual machine compartment. An attacker who compromises the web services layer would then have no direct access to the grid application.

Decoupling the grid application from the web services middleware is not a simple task. The current design for grid services assumes that these two components run on top of the same operating-system. A solution to this problem would be to establish a restricted communications channel to the grid application using the shared memory mechanism discussed in Section 7.4.

9.2.4 Trusted Grid Job Migration

Grid job migration allows a service provider to relocate a grid job onto a grid node that has more resources available. Migration creates problems for attestation. The secret data used by the job security manager is sealed to a particular TPM on the host grid node. Although features have been added to TPMs to support migration of sealed data, these operations must be implemented carefully to ensure they do not undermine security.

The design of the job security manager makes migration relatively straightforward. The execution state of the virtual machines hosting both the job security manager and the grid job can be suspended and transferred to another platform. Provided a virtual TPM is used this should also support seamless migration of data by transferring the virtual TPM onto the destination host [15].

Origin attestation can be implemented without modification in this architecture. When the grid job is migrated along with the secure storage service, the origin attestation value will also be transmitted as part of its protected storage. When the job restarts on the destination platform the origin attestation value will be maintained.

The difficulty in grid job migration comes in assuring that security properties are maintained. For example the user will wish to ensure that migration does not cause data to be passed outside the subgrid of trusted hosts defined in the key management service. Since migration using the virtual TPM occurs without the direct involvement of the key management service, this is a non-trivial task. The virtual TPM implementation can be modified to ensure that it respects the subgrid and other user-based policies, but this may require changes to the existing design. However this should be done while maintaining backwards compatibility if at all possible.

9.3 Conclusion

The trusted grid architecture improves on existing approaches in a number of ways. The middleware problem is solved because the service providers' infrastructure is no longer trusted to access users' data. Even administrators cannot affect the integrity or confidentiality of grid jobs belonging to users of their systems. Digital rights management policies

can be enforced to control the software and data that can be accessed throughout the grid.

Perhaps the most impressive achievement of the new architecture is maintaining backwards compatibility and interoperability. Service providers do not need to make any alterations to their existing infrastructure, and the trusted grid is fully compatible with existing grid middleware software. The job security manager supports grid job attestation independently of the grid job itself, so that it can be used with users' unmodified, legacy software. Interoperability is essential to enable the large-scale deployment of a trusted grid in the near future.

Despite these achievements, it remains to be demonstrated whether there is sufficient demand for secure systems within the grid user community. It is arguable that this situation will change as businesses and other organisations increasingly rely on grid and cloud systems to process sensitive data that is currently kept on their internal systems. Many changes still need to take place before the trusted grid can be universally adopted, including ubiquitous trusted computing support and improvements to virtual machine monitors. The vision of a global trusted grid will take time to arrive but the thesis presents the initial steps towards achieving it, and also points the way to the great gains in security that might be possible in the future.

References

[1] Darren Abramson, Jeff Jackson, Sridhar Muthrasanallur, Gil Neiger, Greg Regnier, Rajesh Sankaran, Ioannis Schoinas, Rich Uhlig, Balaji Vembu, and John Wiegert. Intel virtualization technology for directed I/O. *Intel Technology Journal*, 10(3):179–192, August 2006.

[2] Rafael Accorsi and Adolf Hohl. Delegating secure logging in pervasive computing systems. In John A. Clark, Fiona A. C. Polack, and Richard Paige, editors, *Proceedings of the 3rd International Conference on Security in Pervasive Computing*, volume 3934 of *Lecture Notes in Computer Science*, pages 58–72. Springer-Verlag, Berlin, Heidelberg, 2006.

[3] Roberto Alfieri, Roberto Cecchini, Vincenzo Ciaschini, Luca dell'Agnello, Ákos Frohner, Alberto Gianoli, Károly Lõrentey, and Fabio Spataro. VOMS, an authorization system for virtual organizations. In Francisco Fernández Rivera, Marian Bubak, Andrés Gómez Tato, and Ramón Doallo, editors, *Grid Computing: First European Across Grids Conference (Revised Papers)*, volume 2970 of *Lecture Notes in Computer Science*, pages 33–40. Springer-Verlag, Berlin, Heidelberg, March 2004.

[4] Omar Alhazmi, Yashwant Malaiya, and Indrajit Ray. Security vulnerabilities in software systems: A quantitative perspective. In Sushil Jajodia and Duminda Wijesekera, editors, *Data and Applications Security XIX*, volume 3654 of *Lecture Notes in Computer Science*, pages 281–294. Springer-Verlag, Berlin, Heidelberg, August 2005.

[5] Bill Allcock, Joe Bester, John Bresnahan, Ann L. Chervenak, Carl Kesselman, Sam Meder, Veronika Nefedova, Darcy Quesnel, Steven Tuecke, and Ian Foster. Secure, efficient data transport and replica management for high-performance data-intensive computing. In *MSS '01: Proceedings of the Eighteenth IEEE Symposium on Mass Storage Systems and Technologies*, pages 13–32, Washington, DC, USA, April 2001. IEEE Computer Society.

[6] Zachary Amsden, Daniel Arai, Daniel Hecht, and Pratap Subrahmanyam. Paravirtualization API version 2.5. Available at `http://www.vmware.com/pdf/vmi_specs.pdf`, 2006.

[7] David P. Anderson. BOINC: A system for public-resource computing and storage. In Rajkumar Buyya, editor, *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10, Los Alamitos, CA, November 2004. IEEE Computer Society.

[8] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. SETI@home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, November 2002.

[9] Ross Anderson. Cryptography and competition policy: issues with 'trusted computing'. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 3–10. ACM, July 2003.

[10] William A. Arbaugh, David J. Farber, and Jonathan M. Smith. A secure and reliable bootstrap architecture. In *1997 IEEE Symposium on Security and Privacy*, Los Alamitos, CA, May 1997. IEEE Computer Society.

[11] Boris Balacheff, Liqun Chen, Siani Pearson, David Plaquin, and Graeme Proudler. *Trusted Computing Platforms: TCPA Technology in Context*. Hewlett-Packard Professional Books. Prentice Hall PTR, Upper Saddle River, New Jersey, July 2002.

[12] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the 19th ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, October 2003. ACM.

[13] Mike Beckerle, Glenn Brunette, Lee Cooper, Wan yen Hsu, Adam Jacobs, Thomas Keefe, Richard Nicholson, Parviz Peiravi, Collin Sampson, and Bob Thome. The security architecture for open grid services. Available at `http://www.ogf.org/documents/`, July 2005.

[14] David E. Bell and Leonard J. La Padula. Secure computer systems: Mathematical foundations. Technical Report 2547, MITRE, March 1973.

[15] Stefan Berger, Ramón Cáceres, Kenneth A. Goldman, Ronald Perez, Reiner Sailer, and Leendert van Doorn. vTPM: Virtualizing the trusted platform module. In *Proceedings of the 15th USENIX Security Symposium*, pages 305–320, Berkeley, CA, July 2006. USENIX Association.

[16] C. Blanchet, R. Mollon, and G. Delèage. Building an encrypted file system on the EGEE grid: Application to protein sequence analysis. In *ARES '06: Proceedings of the First International Conference on Availability, Reliability and Security*, pages 965–973, Washington, DC, USA, April 2006. IEEE Computer Society.

[17] Daniel P. Bovet and Marco Cesati. *Understanding the Linux Kernel*. O'Reilly, Sebastopol, CA, 3rd edition, November 2005.

[18] Philippa J. Broadfoot and Gavin Lowe. Architectures for secure delegation within grids. Technical Report PRG-RR-03-19, Oxford University Computing Laboratory, September 2003.

[19] Georg A. Brox. MPEG-21 as an access control tool for the national health service care records service. *Journal of Telemedicine and Telecare*, 11(Supplement 1):23–25, July 2005.

[20] Brad Calder, Andrew A. Chien, Ju Wang, and Don Yang. The Entropia virtual machine for desktop grids. In *Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments (VEE '05)*, pages 186–196, New York, NY, USA, June 2005. ACM.

[21] David W. Chadwick and Alexander Otenko. The PERMIS X.509 role based privilege management infrastructure. In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 135–140, New York, NY, USA, June 2002. ACM.

[22] Haibo Chen, Jieyun Chen, Wenbo Mao, and Fei Yand. Daonity - grid security from two levels of virtualization. *Information Security Technical Report*, 12(3):111–186, May 2007.

[23] Haibo Chen, Fengzhe Zhang, Cheng Chen, Rong Chen, Binyu Zang, Pen chung Yew, and Wenbo Mao. Tamper-resistant execution in an untrusted operating system using a virtual machine monitor. Technical Report FDUPPITR-2007-0801, Parallel Processing Institute, Fudan University, August 2007. Available at `http://ppi.fudan.edu.cn`.

[24] Liqun Chen, Ernie Brickell, and Jan Camenisch. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick D. McDaniel, editors, *Proceedings of the 11th ACM conference on Computer and Communications security (CCS 2004)*, pages 132–145, Washington DC, USA, October 2004. ACM.

[25] Liqun Chen, Siani Pearson, and Athanasios Vamvakas. A trusted biometric system. Technical Report HPL-2002-185, HP Laboratories, July 2002. Available at `http://www.hpl.hp.com/techreports/`.

[26] Ludmila Cherkasova and Rob Gardner. Measuring CPU overhead for I/O processing in the Xen virtual machine monitor. In *ATEC '05: Proceedings of the 2005 USENIX Annual Technical Conference*, Berkeley, CA, USA, April 2005. USENIX Association.

[27] Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury, and Steven Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23(3):187–200, July 2000.

[28] Ed H. Chi. The social web: Research and opportunities. *Computer*, 41(9):88–91, September 2008.

[29] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines.

In *NSDI'05: Proceedings of the 2nd Symposium on Networked Systems Design & Implementation*, pages 273–286, Berkeley, CA, USA, May 2005. USENIX Association.

[30] Christian S. Collberg and Clark Thomborson. Watermarking, tamper-proofing, and obfuscation - tools for software protection. *IEEE Transactions on Software Engineering*, 28(8):735–746, August 2002.

[31] Common criteria evaluation and validation scheme validation report: Green Hills Software INTEGRITY-178B separation kernel. Available at `http://www.commoncriteriaportal.org`, September 2008. Report No. CCEVS-VR-10119-2008.

[32] Andrew Cooper. Vulnerability analysis of BOINC. Technical Report PRG-RR-05-01, University of Oxford, Oxford, UK, October 2005.

[33] Carlin Covey, Mark Redman, and Thomas Tkacik. An advanced trusted platform for mobile phone devices. *Information Security Technical Report*, 10(2):96–104, January 2005.

[34] Alexander W. Dent and Geraint Price. Certificate management using distributed trusted third parties. In Mitchell [110], pages 251–270.

[35] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, January 1999.

[36] Peter A. Dinda. Addressing the trust asymmetry problem in grid computing with encrypted computation. In *LCR '04: Proceedings of the 7th workshop on Workshop on languages, compilers, and run-time support for scalable systems*, pages 1–7, New York, NY, USA, October 2004. ACM.

[37] Evgueni Dodonov, Joelle Quaini Sousa, and Hèlio Crestana Guardia. Gridbox: securing hosts from malicious and greedy applications. In *Proceedings of the 2nd workshop on Middleware for grid computing*, pages 17–22, New York, NY, USA, October 2004. ACM.

[38] Suchuan Dong, George Em Karniadakis, and Nicholas T. Karonis. Cross-site computations on the TeraGrid. *Computing in Science and Engineering*, 7(5):14–23, September 2005.

[39] Stéphane Duverger. Linux 2.6 kernel exploits. *Journal in Computer Virology*, 4(1):39–60, September 2007.

[40] Dhammika Elkaduwe, Philip Derrin, and Kevin Elphinstone. Kernel design for isolation and assurance of physical memory. In Michael Engel and Olaf Spinczyk, editors, *IIES '08: Proceedings of the 1st workshop on Isolation and integration in embedded systems*, pages 35–40, New York, NY, USA, April 2008. ACM.

[41] Paul England. Practical techniques for operating system attestation. In Peter Lipp, Ahmad-Reza Sadeghi, and Klaus-Michael Koch, editors, *Trusted Computing - Challenges and Applications: First International Conference on Trusted Computing and Trust in Information Technologies*, volume 4968 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, Berlin, Heidelberg, 2008.

[42] Paul England, Butler Lampson, John Manferdelli, Marcus Peinado, and Bryan Willman. A trusted open platform. *Computer*, 36(7):55–62, July 2003.

[43] William M. Farmer, Joshua D. Guttman, and Vipin Swarup. Security for mobile agents: Authentication and state appraisal. In Elisa Bertino, Helmut Kurth, Giancarlo Martella, and Emilio Montolivo, editors, *Proceedings of the 4th European Symposium on Research in Computer Security - ESORICS 96*, volume 1146 of *Lecture Notes in Computer Science*, pages 118–130. Springer-Verlag, September 1996.

[44] Niels Ferguson. AES-CBC + Elephant diffuser: A disk encryption algorithm for Windows Vista. Unpublished, August 2006. Available at `http://www.microsoft.com`.

[45] Renato J. Figueiredo, Peter A. Dinda, and José A. B. Fortes. A case for grid computing on virtual machines. In *Proceedings of the 23rd IEEE International Conference on*

Distributed Computing Systems (ICDCS), pages 550–559. IEEE Computer Society, May 2003.

[46] Ian Foster and Adriana Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In Frans Kaashoek and Ion Stoica, editors, *Peer-to-peer Systems II: Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS 2003)*, volume 2735 of *Lecture Notes in Computer Science*, pages 118–128, Berlin, Heidelberg, October 2003. Springer.

[47] Ian Foster and Carl Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, CA, 2004.

[48] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. The physiology of the grid. In Fran Berman, Geoffrey Fox, and Tony Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*, Wiley Series in Communications Networking & Distributed Systems, chapter 8, pages 217–249. John Wiley & Sons, Chichester, England, May 2003.

[49] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organisations. *International Journal of High Performance Computing Applications*, 15(3):200–222, August 2001.

[50] Ian T. Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A security architecture for computational grids. In *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS '98)*, pages 83–92, New York, NY, USA, November 1998. ACM.

[51] James Frey, Todd Tannenbaum, Miron Livny, Ian Foster, and Steven Tuecke. Condor-G: A computation management agent for multi-institutional grids. *Cluster Computing*, 5(3):237–246, July 2002.

[52] Eimear Gallery and Allan Tomlinson. Conditional access in mobile systems: Securing the application. In *DFMA '05: Proceedings of the First International Conference on*

Distributed Frameworks for Multimedia Applications, pages 190–197, Washington, DC, USA, February 2005. IEEE Computer Society.

[53] Simson L. Garfinkel. Commodity grid computing with Amazon's S3 and EC2. *;Login: The USENIX Magazine*, 32(1), February 2007.

[54] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, and Dan Boneh. Terra: A virtual machine-based platform for trusted computing. In *Operating Systems Review: Proceedings of the 19th Symposium on Operating System Principles (SOSP 2003)*, volume 37 number 5, pages 193–206, New York, NY, October 2003. ©2003 ACM, Inc. http://dx.doi.org/10.1145/1165389.945464.

[55] Tal Garfinkel, Mendel Rosenblum, and Dan Boneh. Flexible OS support and applications for trusted computing. In *Proceeding of the 9th Workshop on Hot Topics in Operating Systems (HotOS IX)*, pages 145–150, Berkeley, CA, USA, May 2003. USENIX Association.

[56] Morrie Gasser, Andy Goldstein, Charlie Kaufman, and Butler Lampson. The digital distributed system security architecture. In *Proceedings of the 12th National Computer Security Conference*, pages 305–319, Baltimore, MD, October 1989. NIST-NCSC.

[57] Joshua Gay. *Free Software, Free Society: Selected Essays of Richard M. Stallman*, chapter 17, pages 117–120. GNU Press, Boston, MA USA, October 2002.

[58] Neil Geddes. The national grid service of the UK. In *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing (e-Science '06)*, pages 94–94, Los Alamitos, CA, December 2006. IEEE Computer Society.

[59] Wolfgang Gentzsch. Sun grid engine: Towards creating a compute power grid. In Rajkumar Buyya, George Mohay, and Paul Roe, editors, *CCGRID '01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, pages 35–36, Washington DC, USA, May 2001. IEEE Computer Society.

[60] Globus toolkit security advisories. Available at `http://www-unix.globus.org/toolkit/advisories.html`, February 2008.

[61] Robert P. Goldberg. Survey of virtual machine research. *IEEE Computer Magazine*, 7(6):34–45, June 1974.

[62] David Grawrock. *The Intel Safer Computing Initiative: Building Blocks for Trusted Computing*. Intel Press, March 2006.

[63] Vivek Haldar, Deepak Chandra, and Michael Franz. Semantic remote attestation: A virtual machine directed approach to trusted computing. In *Proceedings of the 3rd conference on Virtual Machine Research And Technology Symposium (VM '04)*, pages 29–41, Berkeley, CA, USA, May 2004. USENIX Association.

[64] Hermann Härtig, Michael Hohmuth, Norman Feske, Christian Helmuth, Adam Lackorzynski, Frank Mehnert, and Michael Peter. The NIZZA secure-system architecture. In *Proceedings of the 2005 International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Available at `http://ieeexplore.ieee.org`, December 2005.

[65] Hermann Härtig, Michael Hohmuth, Jochen Liedtke, Jean Wolter, and Sebastian Schönberg. The performance of μ-kernel-based systems. In *SOSP '97: Proceedings of the sixteenth ACM symposium on Operating systems principles*, pages 66–77, New York, NY, USA, October 1997. ACM.

[66] Jay Heiser and Mark Nicolett. Assessing the security risks of cloud computing. `http://www.gartner.com`, June 2008. Gartner RAS Core Research Note G00157782.

[67] James Hendricks and Leendert van Doorn. Secure bootstrap is not enough: Shoring up the trusted computing base. In *Proceedings of the Eleventh SIGOPS European Workshop*, pages 65–70, New York, NY, September 2004. ACM.

[68] Matt Henricksen, William Caelli, and Peter Croll. Securing grid data using mandatory access controls. In *ACSW '07: Proceedings of the fifth Australasian symposium on*

ACSW frontiers, pages 25–32, Darlinghurst, Australia, November 2007. Australian Computer Society, Inc.

[69] Jonathan Herzog. Applying protocol analysis to security device interfaces. *IEEE Security & Privacy*, 4(4):84–87, July-Aug. 2006.

[70] Michael Hohmuth and Hermann Härtig. Pragmatic nonblocking synchronization for real-time systems. In Yoonho Park, editor, *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pages 217–230, Berkeley, CA, USA, June 2001. USENIX Association.

[71] Michael Hohmuth, Michael Peter, Hermann Härtig, and Jonathan S. Shapiro. Reducing TCB size by using untrusted components: small kernels versus virtual-machine monitors. In *EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC*, New York, NY, USA, September 2004. Available at `http://www.sigmod.org`.

[72] Russell Housley, Warwick Ford, Tim Polk, and David Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459, January 1999.

[73] Michael Howard and Steve Lipner. *Writing Secure Code*. Microsoft Press, Redmond, Washington, 2nd edition, December 2002.

[74] Andrew Huang. The trusted PC: skin-deep security. *Computer*, 35(10):103–105, October 2002.

[75] Intel Core 2 Extreme processor X6800 and Intel Core 2 Duo desktop processor E6000 and E4000 sequence: Specification update. Available at `http://www.intel.com`, May 2008.

[76] Bernhard Jansen, HariGovind V. Ramasamy, and Matthias Schunter. Flexible integrity protection and verification architecture for virtual machine monitors. Presented at the Second Workshop on Advances in Trusted Computing (WATC '06 Fall), November 2006.

[77] Nicholas R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, March 2000.

[78] Jeffrey R. Jones. Windows Vista 6-month vulnerability report. Available at `http://www.csoonline.com`, June 2007.

[79] Mahesh Kallahalla, Mustafa Uysal, Ram Swaminathan, David E. Lowell, Mike Wray, Tom Christian, Nigel Edwards, Chris I. Dalton, and Frederic Gittler. SoftUDC: A software-based data center for utility computing. *IEEE Computer*, 37(11):38–46, November 2004.

[80] Paul A. Karger. Multi-level security requirements for hypervisors. In *ACSAC '05: Proceedings of the 21st Annual Computer Security Applications Conference*, pages 267–275, Washington, DC, USA, December 2005. IEEE Computer Society.

[81] Paul A. Karger, Mary Ellen Zurko, Douglas W. Bonin, Andrew H. Mason, and Clifford E. Kahn. A retrospective on the VAX VMM security kernel. *IEEE Transactions on Software Engineering*, 17(11):1147–1165, November 1991.

[82] Bernhard Kauer. OSLO: Improving the security of trusted computing. In *SS'07: Proceedings of 16th USENIX Security Symposium*, pages 229–237, Berkeley, CA, USA, August 2007. USENIX Association.

[83] Katarzyna Keahey, Karl Doering, and Ian Foster. From sandbox to playground: Dynamic virtual environments in the grid. In *GRID '04: Proceedings of the 5th International Conference on Grid Computing (Grid 2004)*, pages 34–42, Washington, DC, USA, November 2004. IEEE Computer Society.

[84] Katarzyna Keahey, Ian Foster, Timothy Freeman, and Xuehai Zhang. Virtual workspaces: Achieving quality of service and quality of life in the grid. *Scientific Programming*, 13(4):265–275, October 2005.

[85] Vishal Kher and Yongdae Kim. Securing distributed storage: challenges, techniques, and systems. In *StorageSS '05: Proceedings of the 2005 ACM workshop on Storage security and survivability*, pages 9–25, New York, NY, USA, November 2005. ACM.

[86] David N. Kleidermacher. Towards a high assurance multi-level secure PC for intelligence communities. *Technologies for Homeland Security, 2008 IEEE Conference*, pages 609–614, May 2008.

[87] Leonard Kleinrock. Time-shared systems: A theoretical treatment. *Journal of the Association for Computing Machinery*, 14(2):242–261, April 1967.

[88] Ivan Krsul, Arijit Ganguly, Jian Zhang, Jose A. B. Fortes, and Renato J. Figueiredo. VMPlants: Providing and managing virtual machine execution environments for grid computing. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, pages 7–7, Washington, DC, USA, November 2004. IEEE Computer Society.

[89] Klaus Kursawe, Dries Schellekens, and Bart Preneel. Ecrypt workshop, crash - cryptographic advances in secure hardware. ECRYPT Workshop, CRASH - CRyptographic Advances in Secure Hardware. Available at `http://www.cosic.esat.kuleuven.be`, September 2005.

[90] Massimo Lamanna. The LHC computing grid project at CERN. In *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment: Proceedings of the IXth International Workshop on Advanced Computing and Analysis Techniques in Physics Research*, volume 534 number 1-2, pages 1–6. Elsevier, November 2004.

[91] Butler Lampson, Martin Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: theory and practice. *ACM Transactions on Computer Systems (TOCS)*, 10(4):265 – 310, November 1992.

[92] Butler W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, October 1973.

[93] Hermann Lederer and Victor Alessandrini. DEISA: Enabling cooperative extreme computing in Europe. In Christian Bischof, Martin Bcker, Paul Gibbon, Gerhard R. Joubert, Thomas Lippert, Bernd Mohr, and Frans Peters, editors, *Parallel Computing: Architectures, Algorithms and Applications*, volume 15, pages 689–696, Amsterdam, Netherlands, February 2008. IOS Press.

[94] Michael Litzkow, Miron Livny, and Matthew Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, pages 104–111, Washington DC, USA, June 1988. Computer Society Press.

[95] Zhaoyu Liu, Anthony W. Joy, and Robert A. Thompson. A dynamic trust model for mobile ad hoc networks. *Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS 2004)*, pages 80–85, May 2004.

[96] Hans Löhr, HariGovind V. Ramasamy, Stefan Schulz, Matthias Schunter, and Christian Stüble. Enhancing grid security using trusted virtualization. In Bin Xiao, Laurence T. Yang, Jianhua Ma, Christian Muller-Schloer, and Yu Hua, editors, *Autonomic and Trusted Computing*, volume 4610 of *Lecture Notes in Computer Science*, pages 372–384. Springer-Verlag, Berlin, Heidelberg, August 2007. Fig 1. pp.376 reproduced with kind permission from Springer Science+Business Media. ©Springer-Verlag Berlin Heidelberg 2007.

[97] Markus Lorch, Jim Basney, and Dennis Kafura. A hardware-secured credential repository for grid PKIs. In *Proceedings of the 4th IEEE International Symposium on Cluster Computing and the Grid (CCGrid '04)*, pages 640–647, Washington, DC, USA, April 2004. IEEE Computer Society.

[98] Peter Loscocco and Stephen Smalley. Integrating flexible support for security policies into the linux operating system. In *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, pages 29–42, Berkeley, CA, USA, June 2001. USENIX Association.

[99] Peter A. Loscocco, Stephen D. Smalley, Patrick A. Muckelbauer, Ruth C. Taylor, S. Jeff Turner, and John F. Farrell. The inevitability of failure: The flawed assumption of security in modern computing environments. In *Proceedings of 21st NIST-NCSC National Information Systems Security Conference*, pages 303–314. NIST, October 1998.

[100] Umesh Maheshwari, Radek Vingralek, and William Shapiro. How to build a trusted database system on untrusted storage. In *Proceedings of the 4th Symposium on Operating System Design & Implementation (OSDI'00)*, pages 135–150, Berkeley, CA, USA, October 2000. USENIX Association.

[101] Stefan Mangard, Manfred Aigner, and Sandra Dominikus. A highly regular and scalable AES hardware architecture. *IEEE Transactions on Computers*, 52(4):483–491, April 2003.

[102] Wenbo Mao, Fei Yan, and Chunrun Chen. Daonity: grid security with behaviour conformity from trusted computing. In *Proceedings of the first ACM workshop on Scalable trusted computing (STC '06)*, pages 43–46, New York, NY, USA, November 2006. ACM.

[103] John Marchesini, Sean Smith, Omen Wild, and Rich MacDonald. Experimenting with TCPA/TCG hardware, or: How I learned to stop worrying and love the bear. Technical Report TR2003-476, Department of Computer Science, Dartmouth College, Hanover, NH, December 2003. Available at `http://www.cs.dartmouth.edu/reports/`.

[104] John C. Marchesini and Sean W. Smith. SHEMP: Secure hardware enhanced MyProxy. Technical Report TR2005-532, Department of Computer Science, Dartmouth College, Hanover, NH, February 2005. Available at `www.cs.dartmouth.edu/reports/`.

[105] Andrew Martin. The ten page introduction to trusted computing. Technical Report RR-08-11, Oxford University Computing Laboratory, December 2008. Available at `http://www.comlab.ox.ac.uk`.

[106] Andrew Martin and Po-Wah Yau. Grid security: Next steps. *Information Security Tech. Report*, 12(3):113–122, January 2007.

[107] Felipe Martins, Marcio Maia, Rossana M. de Castro Andrade, Aldri L. dos Santos, and Jose Neuman de Souza. Detecting malicious manipulation in grid environments. In *18th International Symposium on Computer Architecture and High Performance Computing*, pages 28–35, Washington, DC, USA, October 2006. IEEE Computer Society.

[108] Jonathan M. McCune, Trent Jaeger, Stefan Berger, Ramon Caceres, and Reiner Sailer. Shamon: A system for distributed mandatory access control. In *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC '06)*, pages 23–32, Los Alamitos, CA, December 2006. IEEE Computer Society.

[109] Sharon A. Mertz, Chad Eschinger, Tom Eid, and Ben Pring. Dataquest insight: SaaS demand set to outpace enterprise application software market growth. `http://www.gartner.com`, August 2007. Gartner RAS Core Research Note G00150222.

[110] Chris Mitchell, editor. *Trusted Computing*. IEE Professional Applications of Computing Series 6. The Institution of Electrical Engineers (IEE), London, UK, 2005.

[111] Hamid R Motahari-Nezhad, Bryan Stephenson, and Sharad Singhal. Outsourcing business to cloud computing services: Opportunities and challenges. Technical Report HPL-2009-23, HP Laboratories, February 2009. Available at `http://www.hpl.hp.com/techreports/`.

[112] George C. Necula and Peter Lee. Safe, untrusted agents using proof-carrying code. In Giovanni Vigna, editor, *Mobile Agents and Security*, number 1419 in Lecture Notes in Computer Science, pages 61–91. Springer-Verlag, Berlin, Heidelberg, January 1998.

[113] Clifford Neuman, Tom Yu, Sam Hartman, and Ken Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120, July 2005.

[114] Eric Newcomer. *Understanding Web Services: XML, WSDL, SOAP, and UDDI.* Independent Technology Guides. Addison-Wesley, Boston, MA, May 2002.

[115] Stephen Northcutt, Karen Frederick, Scott Winters, Lenny Zeltser, and Ronald W. Ritchey. *Inside Network Perimeter Security: The Definitive Guide to Firewalls, VPNs, Routers, and Intrusion Detection Systems.* New Riders Publishing, Indianapolis, Indiana, July 2003.

[116] Jason Novotny, Steven Tuecke, and Von Welch. An online credential repository for the grid: MyProxy. In *Proceedings of the 10th International Symposium on High Performance Distributed Computing (HPDC-10)*, pages 104–111, Washington, DC, USA, August 2001. IEEE Computer Society.

[117] Open virtualization format specification. Available at `http://www.dmtf.org`, September 2008. Document number DSP0243.

[118] Andy Ozment and Stuart E. Schechter. Milk or wine: does software security improve with age? In *USENIX-SS'06: Proceedings of the 15th USENIX Security Symposium*, pages 93–104, Berkeley, CA, USA, July 2006. USENIX Association.

[119] Andreas Pashalidis and Chris J. Mitchell. Single sign-on using TCG-conformant platforms. In Mitchell [110], pages 175–194.

[120] Laura Pearlman, Von Welch, Ian Foster, Carl Kesselman, and Steve Tuecke. A community authorization service for group collaboration. In *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks*, pages 50–59, Washington, DC, USA, November 2002. IEEE Computer Society.

[121] Siani Pearson. Trusted agents that enhance user privacy by self- profiling. Technical Report HPL-2002-196, HP Laboratories, July 2002. Available at `http://www.hpl.hp.com/techreports/`.

[122] Marcus Peinado, Yuqun Chen, Paul England, and John Manferdelli. NGSCB: A trusted open system. In *The 9th Australasian Conference on Information Security and Privacy*, pages 86–97, Berlin, Heidelberg, July 2004. Springer.

[123] Adrian Perrig, Sean W. Smith, Dawn Song, and J. Doug Tygar. SAM: A flexible and secure auction architecture using trusted hardware. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS 2001)*, pages 1764–1773, Los Alamitos, CA, April 2001. IEEE Computer Society.

[124] Birgit Pfitzmann, James Riordan, Christian Stüble, Michael Waidner, and Arnd Weber. The PERSEUS system architecture. In Dirk Fox, Marit Köhntopp, and Andreas Pfitzmann, editors, *Proceedings der GI Fachtagung für Verlässliche IT-Systeme (Dependable IT Systems) (VIS) '01*, pages 1–18, Wiesbaden, Germany, September 2001. Vieweg Verlag.

[125] Roman Pletka and Christian Cachin. Cryptographic security for a high-performance distributed file system. In *Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007)*, pages 227–232, Los Alamitos, CA, September 2007. IEEE Computer Society.

[126] Bogdan C. Popescu, Maarten van Steen, and Andrew S. Tanenbaum. A security architecture for object-based distributed systems. In *ACSAC '02: Proceedings of the 18th Annual Computer Security Applications Conference*, pages 161–171, Washington, DC, USA, December 2002. IEEE Computer Society.

[127] David J. Power, Eugenia A. Politou, Mark A. Slaymaker, and Andrew C. Simpson. Towards secure grid-enabled healthcare. *Software - Practice & Experience*, 35(9):857–871, July 2005.

[128] Omer Rana and Jeremy Hilton. Securing the virtual organization - Part 1: Requirements from grid computing. *Network Security*, 2006(4):7–10, April 2006.

[129] Shane Rau and Richard Shim. Worldwide PC interface and technologies 2008-2012 forecast, part 2. `http://www.idc.com`, December 2008. IDC Market Analysis Doc No. 215915.

[130] Dickon Reed, Ian Pratt, Paul Menage, Stephen Early, and Neil Stratford. Xenoservers: Accountable execution of untrusted programs. In *HOTOS '99: Proceedings of The Seventh Workshop on Hot Topics in Operating Systems*, pages 136–141, Washington, DC, USA, March 1999. IEEE Computer Society.

[131] Jason F. Reid and William J. Caelli. DRM, trusted computing and operating system architecture. In *Conferences in Research and Practice in Information Technology Series: Proceedings of the 2005 Australasian workshop on Grid computing and e-research*, volume 44, pages 127–136, Darlinghurst, Australia, January 2005. Australian Computer Society, Inc.

[132] Gordon Rohrmair. *Using CSP to Verify Security-Critical Applications.* PhD thesis, Oxford University, Oxford, UK, March 2005.

[133] Paul Ruth, Xuxian Jiang, Dongyan Xu, and Sebastien Goasguen. Virtual distributed environments in a shared infrastructure. *Computer*, 38(5):63–69, May 2005.

[134] Paul Ruth, Junghwan Rhee, Dongyan Xu, Rick Kennell, and Sebastien Goasguen. Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure. In *Proceedings of the 3rd International Conference on Autonomic Computing (ICAC 2006)*, pages 5–14, Washington, DC, USA, June 2006. IEEE Computer Society.

[135] Ahmad-Reza Sadeghi, Marcel Selhorst, Christian Stüble, Christian Wachsmann, and Marcel Winandy. TCG inside?: a note on TPM specification compliance. In *STC '06: Proceedings of the first ACM workshop on Scalable trusted computing*, pages 47–56, New York, NY, USA, November 2006. ACM.

[136] Ahmad-Reza Sadeghi and Christian Stüble. Property-based attestation for computing platforms: caring about properties, not mechanisms. In *NSPW '04: Proceedings of*

the 2004 workshop on New security paradigms, pages 67–77, New York, NY, USA, September 2004. ACM.

[137] Ahmad-Reza Sadeghi and Christian Stüble. Taming trusted platforms by operating system design. In Kijoon Chae and Moti Yung, editors, *Information Security Applications*, volume 2908 of *Lecture Notes in Computer Science*, pages 286–302. Springer-Verlag, February 2004.

[138] Ahmad-Reza Sadeghi, Christian Stüble, and Norbert Pohlmann. European multilateral secure computing base - open trusted computing for you and me. In *Datenschutz und Datensicherheit (DUD)*, pages 548–553, Wiesbaden, Germany, September 2004. Vieweg.

[139] Reiner Sailer, Trent Jaeger, Enriquillo Valdez, Ramón Cáceres, Ronald Perez, Stefan Berger, John Linwood Griffin, and Leendert van Doorn. Building a MAC-based security architecture for the Xen open-source hypervisor. In *ACSAC '05: Proceedings of the 21st Annual Computer Security Applications Conference*, pages 276–285, Washington, DC, USA, December 2005. IEEE Computer Society.

[140] Reiner Sailer, Trent Jaeger, Xiaolan Zhang, and Leendert van Doorn. Attestation-based policy enforcement for remote access. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 308–317, New York, NY, USA, October 2004. ACM.

[141] Reiner Sailer, Enriquillo Valdez, Trent Jaeger, Ronald Perez, Leendert van Doorn, John Linwood Griffin, and Stefan Berger. sHype: Secure hypervisor approach to trusted virtualized systems. Technical Report Research Report RC23511, IBM, February 2005. Available at `http://domino.watson.ibm.com`.

[142] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *Proceedings of the 13th USENIX Security Symposium*, pages 223–238, Berkeley, CA, USA, August 2004. USENIX Association.

[143] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 223–238, Berkeley, CA, USA, August 2004. USENIX Association.

[144] Ravi Sandhu, Kumar Ranganathan, and Xinwen Zhang. Secure information sharing enabled by trusted computing and PEI models. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 2–12, New York, NY, USA, March 2006. ACM.

[145] Ravi Sandhu and Xinwen Zhang. Peer-to-peer access control architecture using trusted computing technology. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 147–158, New York, NY, USA, June 2005. ACM.

[146] Daniel Sandler and Dan S. Wallach. Casting votes in the auditorium. Available at `http://www.usenix.org/events/evt07/tech/`, August 2007.

[147] Sriya Santhanam, Pradheep Elango, Andrea Arpaci Dusseau, and Miron Livny. Deploying virtual machines as sandboxes for the grid. In *Proceedings of the 2nd conference on Real, Large Distributed Systems (WORLDS '05)*, volume 2, pages 7–12, Berkeley, CA, USA, December 2005. USENIX Association.

[148] Luis F. G. Sarmenta, Marten van Dijk, Charles W. O'Donnell, Jonathan Rhodes, and Srinivas Devadas. Virtual monotonic counters and count-limited objects using a TPM without a trusted OS. In *Proceedings of the first ACM workshop on Scalable trusted computing (STC '06)*, pages 27–42, New York, USA, November 2006. ACM.

[149] Diego Scardaci and Giordano Scuderi. Managing confidential data in the gLite middleware. In *WETICE '07: Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 298–299, Washington, DC, USA, June 2007. IEEE Computer Society.

[150] Jonathan S. Shapiro, Jonathan M. Smith, and David J. Farber. EROS: a fast capability system. *SIGOPS Operating Systems Review*, 33(5):170–185, December 1999.

[151] Elaine Shi, Adrian Perrig, and Leendert van Doorn. BIND: A fine-grained attestation service for secure distributed systems. In *Proceedings of the 2005 IEEE Symposium on Security & Privacy*, pages 154–168, Washington, DC, USA, May 2005. IEEE Computer Society.

[152] Robert Shingledecker, John Andrews, and Christopher Negus. *The Official Damn Small Linux Book: The Tiny Adaptable Linux That Runs on Anything.* Prentice Hall PTR, Upper Saddle River, NJ, USA, August 2007.

[153] Richard O. Sinnott, Jipu Jiang, John Watt, and Oluwafemi Ajayi. Shibboleth-based access to and usage of grid resources. In *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, pages 136–143, Washington, DC, USA, September 2006. IEEE Computer Society.

[154] James E. Smith and Ravi Nair. *Virtual Machines: Versatile Platforms for Systems and Processes.* Elsevier, San Francisco, CA, June 2005.

[155] Matthew Smith, Thomas Friese, Michael Engel, and Bernd Freisleben. Countering security threats in service-oriented on-demand grid computing using sandboxing and trusted computing techniques. *Journal of Parallel and Distributed Computing*, 66(9):1189–1204, September 2006.

[156] Sean W. Smith. Outbound authentication for programmable secure coprocessors. In *Proceedings of the 7th European Symposium on Research in Computer Security*, pages 72–89, London, UK, October 2002. Springer-Verlag.

[157] Sean W. Smith. *Trusted Computing Platforms: Design and Applications.* Springer, New York, USA, December 2004.

[158] David F. Snelling, Sven van den Berghe, and Vivian Qian Li. Explicit trust delegation: Security for dynamic grids. *Fujitsu Scientific and Technical Journal*, 40(2):282–294, December 2004.

[159] David Stainforth, Jamie Kettleborough, Andrew Martin, Andrew Simpson, Richard Gillis, Ali Akkas, Richard Gault, Mat Collins, David Gavaghan, and Myles Allen. Climate*prediction*.net: design principles for public resource modelling research. In S. G. Akl and T. Gonzalez, editors, *Proceedings of the 14th IASTED conference on parallel and distributed computing systems*, pages 32–38, Calgary, Canada, November 2002. ACTA Press.

[160] David Stainforth, Andrew Martin, Andrew Simpson, Carl Christensen, Jamie Kettleborough, Tolu Aina, and Myles Allen. Security principles for public-resource modeling research. In *WETICE '04: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 319–324, Washington, DC, USA, June 2004. IEEE Computer Society.

[161] Geoff Stoker, Brian S. White, Ellen Stackpole, Thomas J. Highley, and Marty Humphrey. Toward realizable restricted delegation in computational grids. In *Proceedings of the 9th International Conference on High Performance Computing and Networking (HPCN Europe 2001)*, pages 32–41, London, UK, June 2001. Springer-Verlag.

[162] Geoffrey Strongin. Trusted computing using AMD "Pacifica" and "Presidio" secure virtual machine technology. *Information Security Technical Report*, 10(2):120–132, May 2005.

[163] Jeremy Sugerman, Ganesh Venkitachalam, and Beng-Hong Lim. Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor. In *Proceedings of the General Track: 2001 USENIX Annual Technical Conference*, pages 1–14, Berkeley, CA, USA, June 2001. USENIX Association.

[164] G. Edward Suh, Dwaine Clarke, Blaise Gassend, Marten van Dijk, and Srinivas Devadas. AEGIS: Architecture for tamper-evident and tamper-resistant processing. In *Proceedings of the 17th annual international conference on Supercomputing*, pages 160–171, New York, NY, USA, June 2003. ACM.

[165] Michael M. Swift, Brian N. Bershad, and Henry M. Levy. Improving the reliability of commodity operating systems. In *SOSP '03: Proceedings of the 19th ACM symposium on Operating systems principles*, pages 207–222, New York, NY, October 2003. ACM.

[166] *TCG PC Specific Implementation Specification Version 1.1*, August 2003. Trusted Computing Group. Available at `https://www.trustedcomputinggroup.org`.

[167] *TCG Trusted Network Connect TNC Architecture for Interoperability*, May 2005. Trusted Computing Group. Available at `https://www.trustedcomputinggroup.org`.

[168] Rodney Thayer, Naganand Doraswamy, and Rob Glenn. IP Security Document Roadmap. RFC 2411, November 1998.

[169] Robert Thibadeau. Trusted computing for disk drives and other peripherals. *IEEE Security & Privacy*, 4(5):26–33, September 2006.

[170] *TPM Main Specification Part 1: Design Principles (Revision 94)*, March 2006. Trusted Computing Group. Available at `https://www.trustedcomputinggroup.org`.

[171] Harvey Tuch, Gerwin Klein, and Gernot Heiser. OS verification: now! In *HOTOS'05: Proceedings of the 10th conference on Hot Topics in Operating Systems*, pages 7–12, Berkeley, CA, USA, June 2005. USENIX Association.

[172] Dean Turner, Marc Fossi, Eric Johnson, Trevor Mack, Joseph Blackbird, Stephen Entwisle, Mo King Low, David McKinney, and Candid Wueest. Symantec global internet security threat report: Trends for July-December 07. Volume XIII, available at `http://www.symantec.com`, April 2008.

[173] Marten van Dijk, Jonathan Rhodes, Luis F. G. Sarmenta, and Srinivas Devadas. Offline untrusted storage with immediate detection of forking and replay attacks. In *STC '07: Proceedings of the 2007 ACM workshop on Scalable trusted computing*, pages 41–48, New York, NY, USA, November 2007. ACM.

[174] Chrisi Vaughan. Xbox security issues and forensic recovery methodology (utilising Linux). *Digital Investigation*, 1(3):165–172, September 2004.

[175] Giovanni Vigna. Cryptographic traces for mobile agents. In Giovanni Vigna, editor, *Mobile Agents and Security*, number 1419 in Lecture Notes in Computer Science, pages 137–153, Berlin, Heidelberg, January 1998. Springer-Verlag.

[176] VMWare security advisories. Available at `http://www.vmware.com/security/advisories/`, February 2008.

[177] Andrew Warfield, Steven Hand, Keir Fraser, and Tim Deegan. Facilitating the development of soft devices. In *ATEC '05: Proceedings of the 2005 USENIX Annual Technical Conference*, Berkeley, CA, USA, April 2005. USENIX Association.

[178] Von Welch, Ian Foster, Carl Kesselman, Olle Mulmo, Laura Pearlman, Steven Tuecke, Jarek Gawor, Sam Meder, and Frank Siebenlist. X.509 proxy certificates for dynamic delegation. In K. I. Sankar, N. E. Hastings, and W. T. Polk, editors, *In Proceedings of the 3rd Annual PKI R&D Workshop*. NIST Technipubs technical publications, April 2004. Publication No. NIST IR7122.

[179] Paul Willmann, Scott Rixner, and Alan L. Cox. Protection strategies for direct access to virtualized I/O devices. In *Proceedings of the 2008 USENIX annual conference (USENIX '08)*, pages 15–28, Berkeley, CA, USA, June 2008. USENIX Association.

[180] Charles P Wright, Jay Dave, and Erez Zadok. Cryptographic file systems performance: What you dont know can hurt you. In *Proceedings of the Second IEEE International Security in Storage Workshop (SISW '03)*, pages 47–47, Los Alamitos, CA, USA, October 2003. IEEE Computer Society.

[181] B.S. Yee. A sanctuary for mobile agents. In J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, volume 1603 of *Lecture Notes in Computer Science*, pages 261–274. Springer-Verlag, New York, 1999.

[182] Choonhan Youn, Tim Kaiser, Cindy Santini, and Dogan Seber. Design and implementation of services for a synthetic seismogram calculation tool on the grid. In Vaidy S. Sunderam, Geert Dick van Albada, Peter M.A. Sloot, and Jack J. Dongarra, editors, *Proceedings of the 5th International Conference in Computational Science ICCS 2005*, volume 3514 of *Lecture Notes in Computer Science*, pages 469–476. Springer-Verlag, Berlin, Heidelberg, May 2005.

[183] Ming Zhao, Jian Zhang, and Renato Figueiredo. Distributed file system support for virtual machines in grid computing. In *HPDC '04: Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC'04)*, pages 202–211, Washington, DC, USA, June 2004. IEEE Computer Society.